DESARROLLO DE UNA APLICACIÓN PARA LA INTEGRACIÓN DE TÉCNICAS DE RECONOCIMIENTO DE PATRONES

Gonzalo Farías[1], Matilde Santos[2], Sebastián Dormido-Canto[1]
[1]Depto. Informática y Automática, UNED. 28040 Madrid
[2]Depto. Arquitectura de Computadores y Automática, UCM. 28040 Madrid.
gfarias@bec.uned.es, msantos@dacya.ucm.es, sebas@dia.uned.es

Resumen

El presente trabajo pretende abordar la integración de diversas técnicas de reconocimiento de patrones, en una única aplicación interactiva. El objetivo principal, consiste en proporcionar conocimiento y experiencia en las múltiples posibilidades de las técnicas mencionadas, haciendo transparente al usuario toda la programación requerida para ello. Las técnicas de reconocimiento de patrones implementadas son de dos tipos: procesamiento de señales y aprendizaje supervisado. Para el primer tipo, se han incorporado desde funciones estadísticas simples hasta técnicas avanzadas como la transformada discreta de wavelet. Respecto a las técnicas de aprendizaje supervisado se han seleccionado, las máquinas de vectores soporte y las redes neuronales debido a su uso generalizado en este campo. El desarrollo de la aplicación se ha efectuado en Matlab.

Palabras Clave: Educación en Automática, Reconocimiento de Patrones, Procesamiento de Señales, Aprendizaje Supervisado.

1 INTRODUCCIÓN

En la actualidad, la integración de múltiples funciones y la interactividad de las aplicaciones son aspectos relevantes en la enseñanza de distintas áreas de la automática [1], [7]. En el caso particular del reconocimiento de patrones, una opción, que incorpore tales características, son las herramientas comerciales, que sin embargo, por su coste o por la dificultad de incorporar nuevas funcionalidades, pueden no ser atractivas para su adquisición. Otra opción es el desarrollo de aplicaciones propias, en ambientes de programación técnica como Matlab[10], pero que exigen por parte del usuario un importante conocimiento del lenguaje de programación.

Debido a lo anterior, y a las experiencias obtenidas por los autores respecto del tema [2], [4], se ha decidido desarrollar una aplicación capaz de incorporar de forma integral diferentes técnicas de reconocimiento de patrones, que evite en lo posible tareas de programación al usuario, y proporcione mediante la interactividad, una retroalimentación respecto de la aplicación y utilidad de las funciones implementadas.

Las técnicas de reconocimiento de patrones utilizadas en este trabajo se agrupan en dos tipos: Procesamiento de Señales y Clasificación mediante Aprendizaje Supervisado [3].

El presente trabajo está estructurado de la siguiente forma: En la sección 2 se presentan las herramientas y técnicas utilizadas para la implementación de la aplicación. En la sección 3 se describen las principales características y opciones disponibles en la aplicación de forma similar a un breve manual de usuario. La sección 4 presenta dos ejemplos simples que detallan el uso de las funciones implementadas. Finalmente, en la sección 5 se exponen las conclusiones y trabajos futuros.

2 HERRAMIENTAS Y TÉCNICAS

Para alcanzar los objetivos propuestos se decidió utilizar el entorno de programación de Matlab. Las principales razones se deben a las múltiples funciones, tanto para el procesamiento de datos como para el aprendizaje supervisado, ya implementadas, así como las herramientas que posee Matlab para el diseño de una Interfaz Gráfica de Usuario (GUI) [8].

2.1 PROCESAMIENTO DE SEÑALES

El procesamiento de señales e imágenes constituye uno de los elementos fundamentales de las nuevas tecnologías de la información, pues posee aplicaciones muy diversas en diferentes ámbitos, tales como la comunicación digital, el procesamiento de la voz, el análisis de series temporales, la ingeniería biomédica, la teledetección, el control industrial, el vídeo, etc. El principal objetivo de estas técnicas es la extracción de características de las señales procesadas. Evidentemente, la utilidad de las funciones de procesamiento de señales estará dada

por una elección adecuada para el tipo de señal a analizar. Sin embargo, en muchos casos dicha elección no es trivial, por lo que la experiencia y conocimiento tanto de la señal como de la función a aplicar es un factor relevante.

2.2 APRENDIZAJE SUPERVISADO

El aprendizaje supervisado es una técnica de reconocimiento de patrones que permite estimar la salida de una función a partir de un conjunto de datos de entrenamiento.

La figura 1 presenta un diagrama de bloques del proceso de aprendizaje supervisado. En él se observan tres elementos principales: el maestro, la técnica de aprendizaje y el ajuste de parámetros adecuados, siendo éste último generado por una comparación entre lo indicado por el maestro y la técnica de aprendizaje.

La descripción del esquema se basa en que el maestro (o experto) determina con certeza la clase de una señal proveniente del entorno. Simultáneamente una técnica de aprendizaje realiza una predicción de la misma señal. La predicción es comparada con el valor indicado por el maestro, ajustándose los parámetros de la técnica de aprendizaje si existe error. Evidentemente el aprendizaje finalizará, si el error es cero o se ha llegado a un mínimo satisfactorio.

Una vez que el entrenamiento ha sido realizado es necesario evaluar la técnica de aprendizaje, para ello se introducirán señales distintas a las utilizadas en el entrenamiento, denominadas señales de prueba o test. La evaluación concluirá con el cálculo de la *tasa de éxito de clasificación* definida como el cociente entre el número de aciertos y el número total de señales.

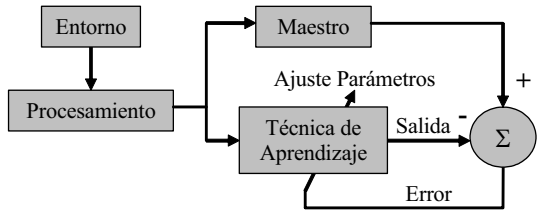

Figura 1: Proceso del aprendizaje supervisado.

3 APLICACIÓN

La aplicación desarrollada en Matlab, permite conseguir los objetivos planteados en la sección 1.

En la aplicación se han implementado técnicas tanto para el procesamiento de señales como para el aprendizaje supervisado.

En el caso del procesamiento de señales se pueden encontrar desde funciones para el análisis estadístico hasta otras más avanzadas como la transformada wavelet discreta [6]. En el caso del aprendizaje supervisado, se han implementado sólo las redes neuronales multicapa [5] y las máquinas de vectores soporte [9], y aunque existe un gran número de técnicas, el uso generalizado de las seleccionadas, proporciona un amplio campo de aplicación.

La aplicación consiste en una interfaz gráfica de usuario y un conjunto de funciones (archivos `.m`). Las funciones sirven para: gestionar las acciones efectuadas por el usuario en la GUI, realizar las llamadas a las funciones propias de Matlab, e implementar las máquinas de vectores soporte.

3.1 PROCEDIMIENTO GENERAL

Los pasos a seguir para el desarrollo de experiencias en la aplicación son expuestas en la figura 2.

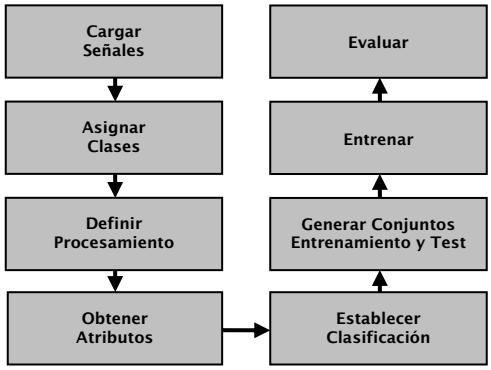

Figura 2: Procedimiento de Experimentación.

Si bien el procedimiento mostrado está definido mediante un conjunto de acciones secuenciales, no necesariamente se debe seguir el orden establecido, ya que es posible asignar nuevas clases a las señales sin tener que volver a establecer las acciones de procesamiento o los parámetros de clasificación.

3.2 INTERFAZ GRÁFICA DE USUARIO

La interfaz de la aplicación desarrollada se presenta en la figura 3.

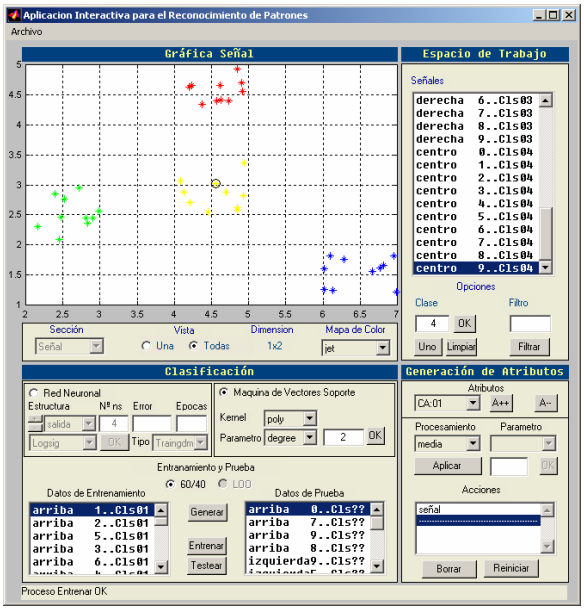

Figura 3: Interfaz de la aplicación implementada.

Es posible observar que la GUI está dividida en las siguientes secciones: *Gráfica Señal*, *Espacio de Trabajo*, *Generación de Atributos*, *Clasificación* y *Menu Archivo*.

3.2.1 Gráfica de la Señal

Esta sección permite disponer en forma visual las señales utilizadas. La señal es mostrada al hacer clic en el `ListBox` *Señales* de la sección *Espacio de Trabajo*. Dependiendo de la dimensión de los datos es posible observar las señales de tres formas:

- *Dimensión* (1,2): Cuando las señales tienen sólo dos atributos, y se representan como puntos en el plano.
- *Dimensión* (1,N): Cuando las señales son de tipo unidimensional, y se representan en forma similar a una serie temporal.
- *Dimensión* (M,N): Este caso corresponde a señales matriciales, y se representan como imágenes.

Esta sección permite además observar en forma conjunta todas las señales, cuando éstas son de dimensión (1,2) o (1,N), haciendo clic en el `RadioButton` *Todas*.

3.2.2 Espacio de Trabajo

En esta sección se asocian las señales a una clase determinada. Esto se realiza en los siguientes pasos:

- Hacer clic en la señal deseada en el `ListBox` de *Señales*.
- Escribir el número de la clase en el `EditText` *Clase*.
- Hacer clic en el `PushButton` *OK*.

Es posible deshacer la asociación de una señal a una clase, simplemente introduciendo el número de clase correcta, o un carácter distinto de un número, si se desea que la señal no esté asociada a ninguna clase.

3.2.3 Generación de Atributos

Esta sección permite el establecimiento de las acciones de procesamiento de señales que generarán los atributos a utilizar en el entrenamiento y la evaluación. Las señales procesadas están formadas por la concatenación de conjuntos de atributos denominadas *CAn*, donde *n* es el índice del conjunto. Un conjunto de atributos se genera a partir de la señal y de las funciones de procesamiento aplicadas de forma secuencial.

Las acciones a seguir para la generación de atributos se detallan a continuación:

- *Creación de Conjuntos de Atributos*: Los conjuntos de atributos se seleccionan a través del `PopupMenu` ubicado debajo del `StaticText` *Atributos*. Se agregan o eliminan conjuntos de atributos utilizando los `PushButton` *A++* y *A--* respectivamente.
- *Definir Procesamiento*: Las funciones de procesamiento de señales se seleccionan desde el `PopupMenu` *Procesamiento*. Si la función requiere de parámetros, éstos estarán disponibles en el `PopupMenu` *Parámetros*. Para indicar los valores de cada uno de los parámetros es necesario usar `EditText` y `PushButton` *OK*.
- *Guardar Procesamiento*: Para que la función de procesamiento definida sea finalmente registrada, es necesario hacer clic en el `PushButton` *Aplicar*, una vez hecho esto el nombre de la función aparecerá en el `ListBox` *Acciones*. Si se desean eliminar todas o una función en particular se debe hacer clic en los `PushButton` *Reiniciar* o *Borrar* respectivamente.

3.2.4 Clasificación

En esta sección se debe indicar la técnica de clasificación requerida. Es necesario introducir los parámetros correspondientes a la clasificación antes de comenzar el entrenamiento.

Las opciones para la *Red Neuronal* son las siguientes:

- *Estructura*: Que permite agregar o eliminar capas ocultas, número de neuronas y función de activación por capa. Esto se realiza con los `PopupMenu` y `PushButton` situados debajo de la `StaticText` *Estructura*, y con el `EditText` y `PushButton` *OK* situados debajo del

`StaticText` *Nº ns*. Es importante tener en cuenta que para que los parámetros de una capa sean guardados, es necesario presionar el `PushButton` *OK*. También es importante notar que la estructura de la red cuenta con una capa de salida predefinida y no modificable, esto se estableció para que la capa de salida tenga tantas neuronas como clases definidas existan, y la función de transferencia siempre sea del tipo *Logsig*. El objetivo de esto es estandarizar y simplificar el proceso de evaluación.

- *Parámetros de entrenamiento*: Que permiten indicar el número de épocas de entrenamiento, la función o tipo de entrenamiento, y el objetivo o error cuadrático medio (MSE). Esto se realiza introduciendo o seleccionando los parámetros en los `EditText` y `PopupMenu` correspondientes.

Las opciones de la *Máquina de Vectores Soporte* son las siguientes:

- *Kernel*: El kernel o función núcleo debe seleccionarse del `PopupMenu` a la derecha del `StaticText` *Kernel*. Dependiendo del Kernel seleccionado se deberán introducir los parámetros correspondientes.
- *Parámetros Kernel*: Los parámetros de la función núcleo deben ser introducidos uno a uno utilizando el `PopupMenu`, `EditText` y el `PushButton` *OK* a la derecha del `StaticText` *Parámetros*. Para que el valor de un parámetro quede guardado se debe hacer clic en el `PushButton` *OK*.

En la sección *Clasificación* se encuentran además las opciones generar, entrenar y evaluar o testear, descritas a continuación:

- *Generar*: Esta acción se realiza con el `PushButton` *Generar*, y ejecuta dos o una tareas dependiendo de si el procesamiento ha sido modificado o no. En el primer caso las tareas serán: aplicar las funciones de procesamiento definidas y realizar la partición de las señales en los conjuntos de entrenamiento y prueba. En el segundo caso sólo se realizará ésta última tarea. Las señales correspondientes a los conjuntos de entrenamiento y prueba serán observadas en los `ListBox` de *Datos de Entrenamiento* y *Datos de Prueba* respectivamente.
- *Entrenar*: Al presionar el `PushButton` *Entrena* se ejecutará esta acción. Si lo que se va a entrenar es una red neuronal entonces aparecerá la gráfica del error versus el número de época en la vista de la GUI. Además por la línea de comandos será posible observar, entre otros, los valores de la época y error alcanzado.
- *Testear*: Si el entrenamiento se ha realizado, entonces es posible evaluar la red neuronal o la máquina de vectores soporte obtenida. Para ejecutar esta acción se debe hacer clic en el `PushButton` *Testear*.

3.2.5 Opciones del Menu Archivo

En el menú *archivo* es posible utilizar dos opciones importantes, y que proporcionan mayor flexibilidad a la aplicación. La primera permite introducir un conjunto de datos diferente al que se este usando, la segunda se utiliza para incorporar funciones de procesamiento definidas por el usuario.

- *Introducción de nuevos datos*: Al seleccionar esta opción aparecerá un cuadro de diálogo solicitando la ubicación del archivo (con extensión `.mat`) que contiene el nuevo conjunto de datos. Para que los datos se carguen adecuadamente, el archivo debe poseer las siguientes tres variables: *datos*, donde cada fila contiene un registro o señal, *dim*, donde el primer y segundo elemento indican la dimensión de una señal, y *lista*, donde cada fila contiene una cadena de caracteres distintivos de una señal.
- *Funciones definidas por usuario*: Si se selecciona esta opción, aparece un cuadro de diálogo que solicita el nombre de la función (con extensión `.m`). La función debe tener una cabecera con el mismo nombre del archivo, dos variables de entrada (la primera para los datos, y la segunda para los parámetros). Para configurar de forma automática los `PopupMenu` asociados a la función y parámetros, el archivo debe tener cuatro líneas formateadas. Como ejemplo de lo anterior se muestran las líneas para la función *dsp_wavelet2d.m*:

  %define funcion:
  %nombre:dsp_wavelet2d:
  %parametros:madre,coeficiente,nivel:
  %descripcion:obtiene un coeficiente wavelet 2D de una imagen:

4 EXPERIMENTACIÓN

A continuación se describirán dos ejemplos con el fin de detallar las distintas opciones disponibles en la aplicación desarrollada. El primer ejemplo consiste en un conjunto de puntos distribuidos en el plano, la ubicación de éstos se establece de modo que la separación lineal no sea posible. El segundo ejemplo presenta un conjunto de imágenes de cinco clases que deben ser procesadas de forma adecuada para reducir la dimensionalidad que poseen (~220.000 píxeles).

4.1 PUNTOS EN EL PLANO

El objetivo principal de este ejemplo es mostrar las opciones disponibles en la aplicación, como por ejemplo visualizar datos, asignar clases, establecer parámetros de clasificación, entrenar y evaluar los resultados.

A continuación se describen de forma resumida los pasos establecidos en el procedimiento de experimentación descrito en la sección 3.1.

4.1.1 Cargar Señales y Asignar Clases

Para este ejemplo se dispone de un conjunto de datos donde cada registro cuenta con dos atributos. Los datos, provenientes del archivo *ejempuntos.mat*, son cargados desde la opción *Cargar Datos* del Menu *Archivo*. Para este tipo de datos la GUI representará los registros en un plano, ver figura 4.

Los puntos presentan un color debido a que están asociados a una clase determinada. El punto del centro del plano (y último en la lista de señales) no está coloreado debido a que no tiene asociada ninguna clase.

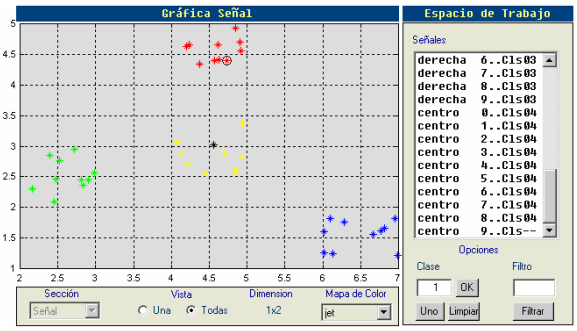

Figura 4: Gráfica de registros de *ejempuntos.mat*.

4.1.2 Definir Procesamiento y Obtener Atributos

Como el número de atributos es de sólo dos elementos, en esta ocasión no se aplicará ningún procesamiento sobre la señal, de esta forma sólo hay un conjunto de atributos, y éste está formado por los elementos de la propia señal como se aprecia en la *sección Generación de Atributos* de la figura 5.

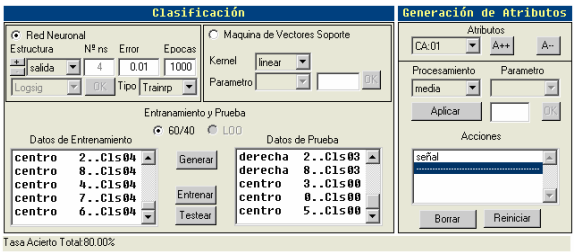

Figura 5: Secciones de *Clasificación* y *Generación de Atributos* de la GUI para *ejempuntos.mat*

4.1.3 Establecer Clasificación y Generar Conjuntos de Entrenamiento y Test

En la figura 5 se observa que la técnica de clasificación seleccionada es la *Red Neuronal*. La estructura de la red sólo consiste en la capa de salida, establecida por defecto con cuatro neuronas (número de clases) y la función de activación *Logsig*. El error exigido es de 0.01 y el número de épocas es de 1000. La función de entrenamiento es *Trainrp*.

Los conjuntos de entrenamiento y test se obtienen al presionar el `PushButton` *Generar*. Las señales agrupadas de esta forma se observan en las `ListBox` de la figura 5.

4.1.4 Entrenar, Evaluar y Modificar Clasificación

Al presionar el `PushButton` *Entrenar* se realiza el entrenamiento de la red neuronal. En la sección *Gráfica de Señal* de la aplicación se observará la evolución del entrenamiento.

Una vez finalizado el entrenamiento se realiza la evaluación presionando el `PushButton` *Testear*. La *tasa de éxito* alcanzada se observa en la barra de estado inferior de la figura 5. En este caso se obtuvo un 80% debido a que las señales *centro 3*, *0* y *5* no han sido correctamente clasificadas, lo que era de esperar, debido a que con una red neuronal de una sola capa, no es posible resolver un problema no separable linealmente. Así entonces, es necesario modificar los parámetros de clasificación de la Red Neuronal. Si se incorpora una nueva capa a la estructura anterior, con un número de neuronas igual a dos, y una función de activación *Tansig*, es posible obtener una *tasa de éxito* del 100% al realizar la clasificación de los puntos.

4.2 IMÁGENES

Este ejemplo está orientado a presentar las posibilidades de la sección *Generación de Atributos* de la GUI. Para ello se exponen dos opciones para la creación de los conjuntos de atributos.

A continuación, se describen de forma resumida los pasos establecidos en el procedimiento general descrito en la sección 3.1.

4.2.1 Cargar Señales y Asignar Clases

Las imágenes son cargadas desde el archivo *ejemimagenes.mat*, esto se realiza con la opción *Cargar Datos* del Menu *Archivo*. Para este tipo de datos la GUI representa los registros como imágenes en la sección *Gráfica de Señal*, ver figura 6. Las dimensiones de las imágenes son de 576x385 píxeles.

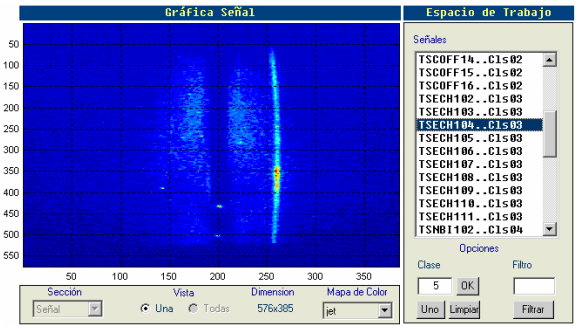

Figura 6: Sección de *Gráfica de Señal* y *Espacio de Trabajo* de la GUI para *ejemimagenes.mat*.

Las imágenes se agrupan en cinco clases. En la figura 6 se observa una imagen de clase 3. Todas las señales, salvo las de clase 1, corresponden a imágenes con al menos cuatro características importantes: una componente vertical a la izquierda, una discontinuidad o vacío al centro, otra componente vertical a la derecha, y una línea fina a la derecha. Las diferencias entre un registro y otro, corresponden a la intensidad de los componentes mencionados, elevada en el caso de la clase 4, más tenue en el caso de la clase 3 (aunque con la línea fina muy marcada), con las intensidades de las componentes concentradas inferiormente en la clase 2, y prácticamente nulas para la clase 5.

4.2.2 Definir Procesamiento y Obtener Atributos

Debido a que el número de atributos es de 221.760 píxeles, es necesario reducir la dimensionalidad antes de aplicar alguna técnica de clasificación. Para esto, se presentan a continuación dos opciones para el procesamiento de las señales.

Opción 1: Se creará sólo un conjunto de atributos a partir de la aplicación de la transformada *wavelet* a la señal.

Opción 2: Se crearán los cuatro conjuntos de atributos siguientes:

- *CA1*: A la señal se le aplicará un corte vertical con la función *franja2d* para obtener la componente izquierda de la imagen.
- *CA2*: A la señal se le aplicará un corte vertical con la función *franja2d* para obtener la componente derecha de la imagen.
- *CA3*: A la señal se le aplicará un corte vertical con la función *franja2d* para obtener la componente línea fina a la derecha de la imagen.
- *CA4*: A la señal se le aplicará un corte horizontal con la función *franja2d* para detectar la concentración inferior de la clase 2.

A todos los conjuntos de atributos de la segunda opción, se les aplicará finalmente la transformada *wavelet* y la *media* para reducir dimensionalidad en las franjas. En la figura 7, en la sección *Generación de Atributos*, se observa el procesamiento realizado para obtener el conjunto de atributos *CA4*.

Cuando se realice la generación de atributos con la *opción 1*, los registros para el entrenamiento y test tendrán 900 atributos en total, mientras que para la *opción 2* sólo serán 37.

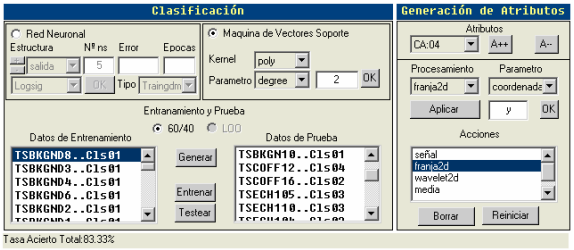

Figura 7: Vista de las secciones de *Clasificación* y *Generación de Atributos* de la GUI para *ejemimagenes.mat*.

4.2.3 Establecer Clasificación y Generar Conjuntos de Entrenamiento y Test

Para clasificar, se seleccionó la *Máquina de Vectores Soporte*, cuyo *Kernel* utilizado es *Poly* (polinomial) de *grado* 2, ver figura 7.

Los conjuntos de entrenamiento y test se obtienen al presionar el `PushButton` *Generar*. Las señales agrupadas de esta forma se observan en las `ListBox` de la figura 7.

4.2.4 Entrenar y Evaluar

El entrenamiento de la máquina de vectores soporte se realiza al presionar el `PushButton` *Entrenar*.

Debido a que el entrenamiento se realiza con la máquina de vectores soporte, el tiempo para entrenar cualquiera de las dos opciones de generación de atributos presentadas en la sección 4.2.2 es muy similar. Sin embargo, si el entrenamiento se realizara con la red neuronal, el tiempo empleado sería superior para la primera opción debido al mayor número de pesos que se necesitan calcular. Por ejemplo, si la red neuronal no cuenta con ninguna capa oculta, en la primera opción se tienen 4500 (900x5) pesos, mientras que en la segunda tan sólo 185 (37x5).

En la evaluación se obtuvieron *tasas de éxito* cercanas al 80% con ambas opciones, mientras que si el kernel cambiaba a *lineal*, el rendimiento para la primera se mantiene en el 80% y el de la segunda baja al 70%.

5 CONCLUSIONES

La aplicación o GUI presentada pretende abordar principalmente dos aspectos: el primero relacionado con la dificultad de encontrar una aplicación no comercial que incorpore en forma integral las funciones de reconocimiento de patrones, y en segundo lugar proporcionar al usuario interactividad cuando se modifican los parámetros de las funciones implementadas.

Los objetivos anteriores proporcionan las siguientes ventajas:

- Se evita programación para la integración de las funciones de reconocimiento de patrones.
- Debido a la integración, los tiempos de evaluaciones son menores, lo que implica un mayor número de pruebas.
- Incorporación de funciones de procesamiento de señal definidas por usuario.
- Flexibilidad respecto a la dimensionalidad de los datos.

El uso de la aplicación se puede destinar tanto a la enseñanza en laboratorios como a la experimentación de funciones de procesamiento definidas por el usuario o el análisis de nuevos conjuntos de datos.

Para trabajos futuros un aspecto a considerar es la independencia de la aplicación del ambiente de programación Matlab, mientras que otro factor en estudio es la posibilidad de ejecución remota de la aplicación.

Referencias

[1] Dormido S, Esquembre F, Farías G, Sánchez J, (2005), "Adding interactivity to existing Simulink models using Easy Java Simulations" CDC-ECC'05 (aceptado)

[2] Dormido-Canto S, Farías G, Dormido R, Vega J, Sánchez J, Santos M, (2004), "TJ-II wave forms analysis with wavelets and support vector machines", *Review of Scientific Instruments* 75, 10, pp. 4254-4257.

[3] Duda R, Hort P, Stork D, (2001), "Pattern Classification", 2nd Edition, A Wiley-Interscience Publication.

[4] Farias G, Dormido R, Santos M, Duro N, (2005), "Image Classifier for the TJ-II Thomson Scattering Diagnostic: Evaluation with a Feed Forward Neural Network". *Lecture Notes in Computes Science*. Springer-Verlag, 3562, pp. 604-612.

[5] Hilera J, Martínez V, (1995), "Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones". Ed. Rama.

[6] Mallat S, (2001), "A Wavelet Tour of Signal Processing". 2º Edition, Academic Press.

[7] Sánchez J, Dormido S, Esquembre F, (2005), "The learning of control concepts using interactive tools". *Computer Applications in Engineering Education*, V.13, I. 1, pp. 84-98.

[8] The MathWorks, Inc. (1984-1997), "Building GUIs with Matlab", Version 5.

[9] Vapnik V, (2000) "The Nature of Statistical Learning Theory", 2º Edition, Springer.

[10] www.mathworks.com