FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

DEPARTAMENTO DE INGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL

PERCEPCIÓN COMPUTACIONAL

IDENTIFICACIÓN DE TEXTURAS NATURALES MEDIANTE TÉCNICAS DE INTELIGENCIA ARTIFICIAL EN VISIÓN POR COMPUTADOR

GONZALO PAJARES MARTINSANZ

DPTO. IGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL

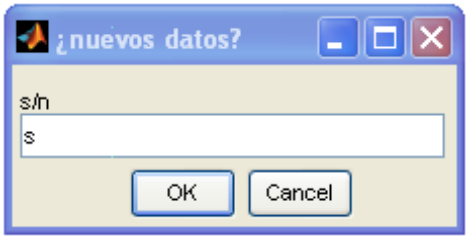
FAULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

APRENDIZAJE

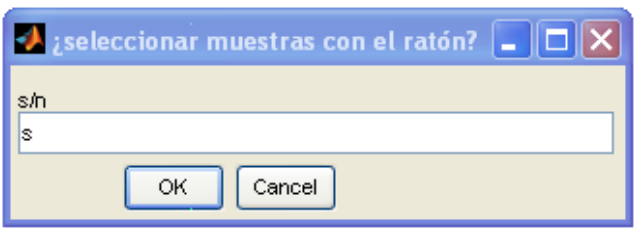
El proceso de aprendizaje se lleva a cabo mediante el script **Aprendizaje.m** cuya ejecución se realiza desde la línea de comandos. Los pasos que se describen a continuación se identifican en el código, exactamente con esta referencia.

Paso 1: cargar datos (si existen de la base de Conocimiento), estos datos se cargan sobre la variable *X* cuya dimensión es *N*x3, conteniendo valores de color de los píxeles de las muestras. Si se contesta que **n** se cargarán sólo los datos existentes en la BC



Paso 2: si en el caso anterior se selecciona **si** (s, opción por defecto), se comienza a ejecutar un bucle de forma que en cada ejecución del bucle abre una nueva imagen que ha sido previamente seleccionada y numerada como imagen de entrenamiento con la nomenclatura **IEntrenamiento_xx.bmp**, donde xx es un valor numérico que se corresponde con el número de iteración. La imagen cargada se muestra en pantalla.

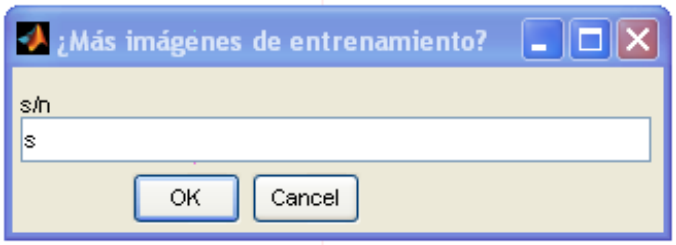
Tras la apertura de una imagen todavía se pregunta si se quiere seleccionar las muestras con el ratón de forma que se seleccionen píxeles de forma controlada o por el contrario que todos los valores de la imagen sean cargados como muestras. El diálogo es el siguiente:



En caso de contestar **si** (s, opción por defecto), se van seleccionando las muestras con el botón izquierdo, pulsando una vez, cuando se quiera dejar de capturar más muestras se pulsa el botón derecho. Sobre la imagen aparece una marca indicando las muestras que se han seleccionado.

Ante cualquiera de las dos opciones se vuelve a preguntar si se desean capturar más muestras, siendo posible de nuevo volver a seleccionar o bien con el ratón o bien toda la imagen. En una iteración se puede seleccionar con el ratón y en la siguiente toda la imagen o viceversa sin que haya ninguna restricción al respecto.

La ventana de diálogo que aparece es la siguiente con la opción **si** (s, por defecto). Cuando se opta por **no** (n), se termina el bucle de captura y por tanto la toma de muestras.

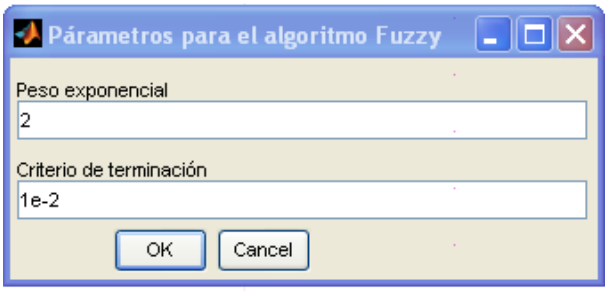


Los nuevos datos se añaden a la variable *X* de muestras previamente cargada procedente de la BC, si no hubiera muestras previas, *X* contendría las muestras seleccionadas.

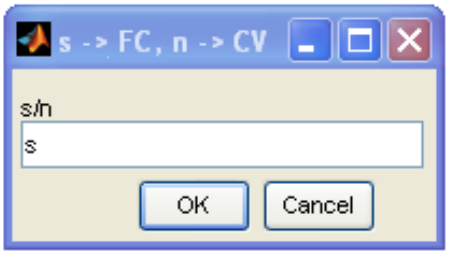
Una vez capturadas todas las muestras, éstas se almacenan en la BC, con el fin de tenerlas disponibles para futuros procesos de aprendizaje.

Las muestras obtenidas son valores de los píxeles en el rango [0,255], tras su captura se procede a su normalización al rango [0,1].

Paso 3: se inicia el proceso de partición de los datos, que como se ha mencionado previamente se basa en uno de los dos algoritmos **FC** o **CV**. El primer diálogo que se muestra a continuación relativo a dos parámetros requerido por el algoritmo **FC**, ya que son necesarios para el proceso de validación posterior independientemente del método que se elija (**FC** o **CV**), aunque el **CV** no los requiera. Estos parámetros son el peso exponencial (*m*) y el criterio de terminación (o error de tolerancia). Los valores por defecto son los que aparecen en la ventana.



Seguidamente se muestra la siguiente ventana de forma que si se elige **si** (s, por defecto) se realizará la partición con el método **FC** y si es **no** (n) será con el **CV**.

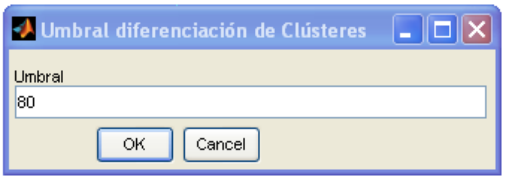


La respuesta **si** implica que es necesario proporcionar el número de clústeres para realizar la clasificación mediante **FC** apareciendo en consecuencia la siguiente ventana,

En este caso se llama a la función **fuzzy_clustering** con las muestras *X* y los parámetros *c*, *m*, error, VEstimados (que en este caso no tiene efecto ninguno) y DatosClasificados = false, porque en esta fase los datos no han sido todavía clasificados. Este último parámetro se utiliza para identificar si los centros deben inicializarse aleatoriamente o no al tener una clasificación previa (como se verá posteriormente). La función devuelve una matriz *v* conteniendo los centros de los clústeres, esta matriz es de dimensión *c* x 3 siendo *c* el número de clústeres y 3 el número de componentes de cada centro, como estamos tratando con datos de píxeles R,G, B de ahí el valor 3. Por tanto, cada fila se corresponde con el centro de cada clúster. También devuelve una matriz *U* que indica el grado de pertenencia de cada una de las muestras a cada uno de los clústeres. Esta matriz será por tanto es de dimensión *N* x *c*, ya que hay *N* muestras y *c* clústeres. La matriz *v* está normalizada al rango [0,1], tras este paso se procede a la desnormalización ya que los datos originales que van a agruparse están sin normalizar. El resto de parámetros devueltos son informativos y no se requieren de momento.

Seguidamente, se procesan todas las muestras identificando a qué clase corresponde el mayor grado de pertenencia y asignado la muestra a la clase con ese mayor grado. De esta forma se obtienen las muestras y el número de las mismas asignadas a cada clase. Esto se almacena en la variable DATOS_CLASES.

Si hubiésemos elegido la opción **no**, se ejecutaría el algoritmo **CV** para lo cual se requiere el umbral *T* de separación de los clústeres, que se solicita a través de la ventana de diñalogo siguiente,



Seguidamente, se procesan todas las muestras identificando a qué clase corresponde el mayor grado de pertenencia y asignado la muestra a la clase con ese mayor grado. De esta forma se obtienen las muestras y el número de las mismas asignadas a cada clase. Esto se almacena en la variable DATOS_CLASES.

Paso 4: validación de resultados. A partir de los datos clasificados contenidos en DATOS_CLASES se procede a validar la partición obtenida previamente. Se inicia el proceso reconstruyendo el vector de datos X, para ser pasados a la función **fuzzy_clustering** la diferencia con la anterior ejecución estriba en que ahora se inicializan los vectores representativos de cada clase tendiendo en cuenta que ya se dispone de una clasificación previa. Esto se lleva a cabo indicando que DatosClasificados = trae.

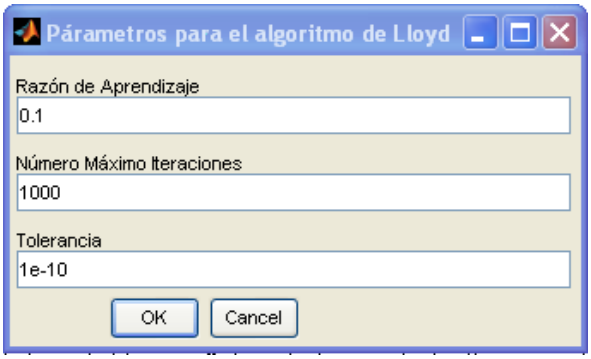
Tras la normalización al rango [0,1] tanto de las muestras *X* como de los vectores representativos de cada clase se procede a llamar a la función **fuzzy_clustering**, que devuelve los nuevos vectores representativos de cada clase a través de la matriz (*v*), la matriz de grados de pertenencia (U) y una distancia, que se utiliza en el método de validación fuzzy. Dependiendo de cual sea el valor del parámetro coeficiente, así se obtienen unos u otros coeficientes de validación fuzzy. Estos coeficientes son los proporcionados en las referencias de métodos y se obtienen en CvalidezFuzzy (en la implementación actual sólo se utiliza el coeficiente PC, en implementaciones futuras hay que combinar los restantes con los derivados del clasificador de Bayes).

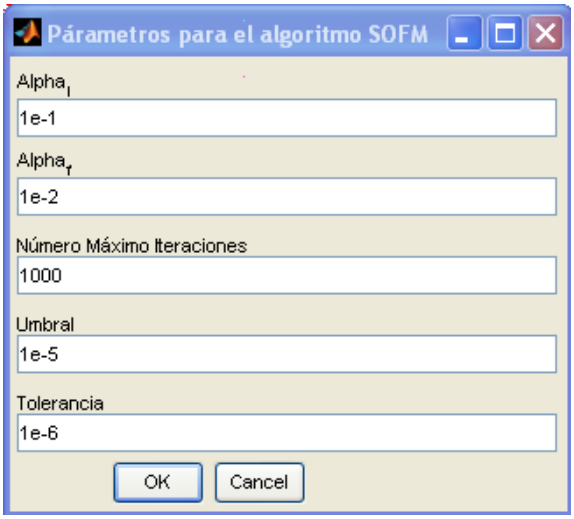
Se procede a la desnormalización de *v* para ser almacenada posteriormente en la BC.

Seguidamente se procede a realizar la clasificación mediante el clasificador **Bayesian** al que se le pasan los datos previamente clasificados como pertenecientes a las diversas clases y devuelve la media y la matriz de covarianza asociadas con cada una de las clases en la variable DATOS_CLASES.

El paso final del proceso de validación consiste en el cómputo de los valores de validación correspondientes a la agrupación Bayesiana a través de la función **validezBayesian** que devuelve los coeficientes de Divergencia, coseno y Jeffries-Matussita entre las clases (de momento estos valores no se evalúan).

Paso 5: si la validación no ha sido satisfactoria, se procede a realizar una nueva partición, no obstante si ésta ha resultado ser satisfactoria, se continúa al proceso de aprendizaje mediante los métodos **MA,CL** preguntándose sobre los parámetros requeridos a través de las siguientes ventanas de diálogo respectivamente. En ambos procedimientos los parámetros aprendidos son los centros de las clases, que serán almacenados en la BC

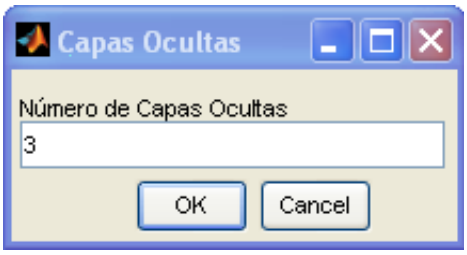




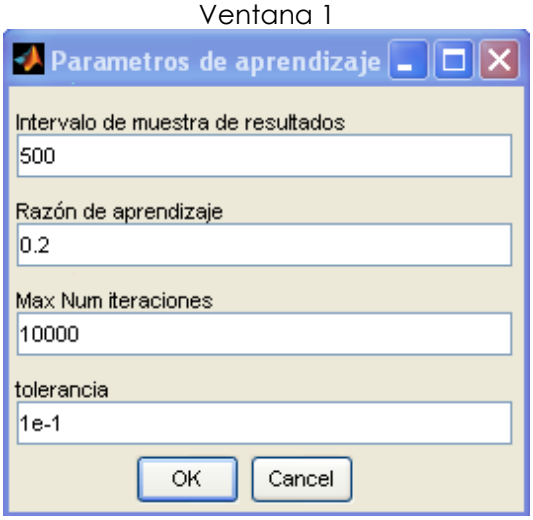
Dentro de la fase de aprendizaje se incluye un último proceso, que es el clasificador basado en la red neuronal **CR.** La primera ventana de diálogo que aparece pregunta sobre el número de capas ocultas que va a tener la red mediante el procedimiento **TopologiaRedCB** y mediante las ventanas de diálogo 1, 2 y 3.

Mediante la función **RetroPropagacion** se crea la red con la función **newff** y finalmente se procede al entrenamiento con la función **train**. Tras este proceso la red creada modifica (aprende) los pesos y se guarda en la BC exactamente la red entrenada como tal.

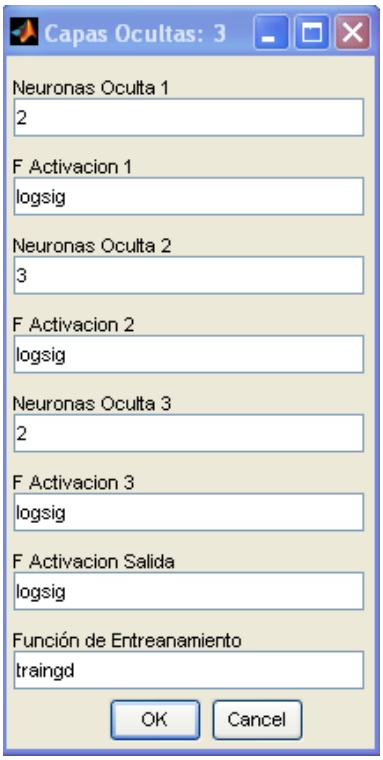
Existe un proceso final para verificar que la red ha aprendido correctamente, que consiste en que suministrando las muestras con las que ha aprendido debe generar exactamente los patrones de entrenamiento con los que se ha entrenado. Ese último proceso se lleva a cabo mediante las funciones **GenerarPatronObjetivo** y **sim.**



Ventana 1

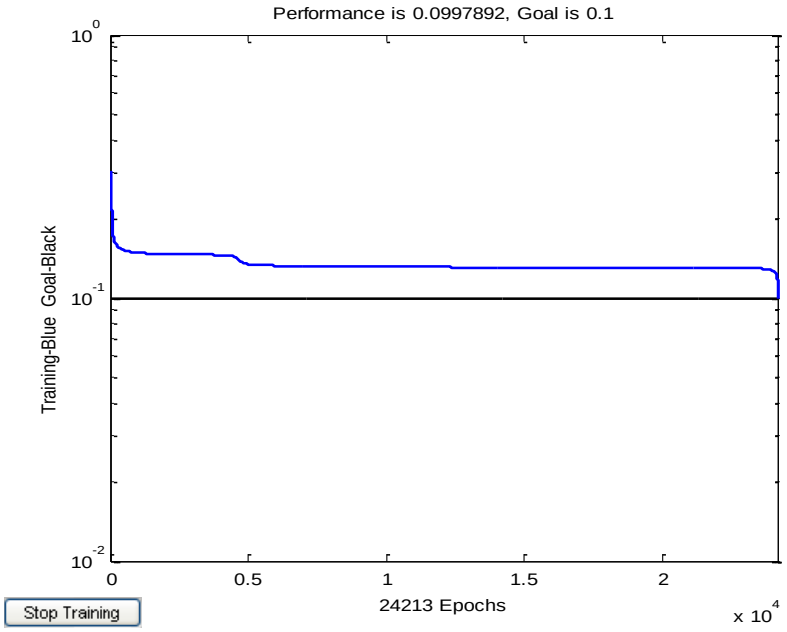


Ventana 3



Ventana 2

La red inicia su proceso de entrenamiento mostrando la siguiente información,

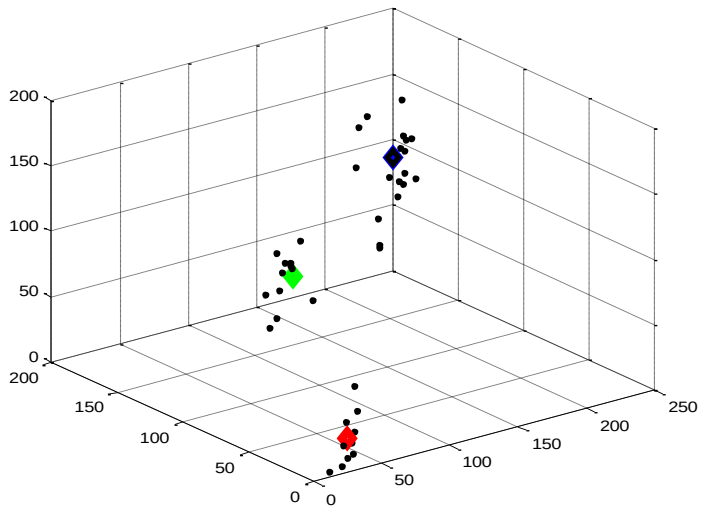


Paso 6: finalmente, se almacenan en la BC los datos aprendidos en el fichero **ParamAprendizaje**

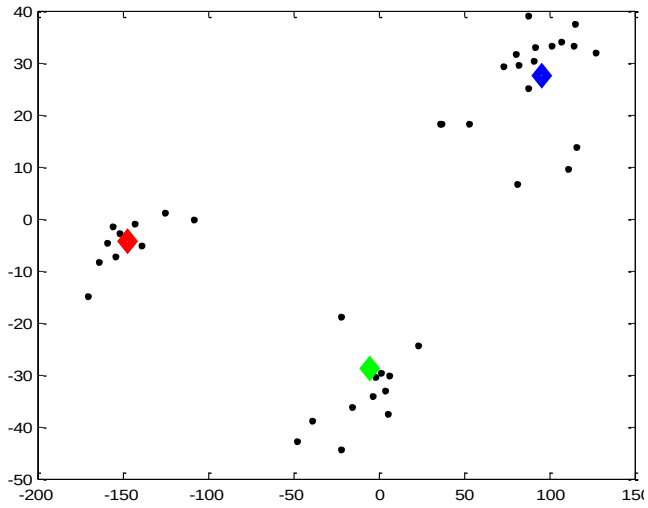
Paso 7: verificación del aprendizaje de la red

Paso 8: representación tridimensional y bidimensional de los datos originales y los centros Fuzzy. En el caso bidimensional se utiliza la proyección de **ComponentesPrincipales** para reducir en una dimensión (de tres a dos) los datos y centros.

Se muestran los resultados que aparecen en la imagen, donde los datos aparecen representados en los puntos negros y los centros como rombos coloreados.



Representación tridimensional



Representación bidimensional

CLASIFICACIÓN

El proceso de aprendizaje se lleva a cabo mediante el script **Clasificador.m** cuya ejecución se realiza desde la línea de comandos. Los pasos que se describen a continuación se identifican en el código, exactamente con esta referencia.

Paso 1: carga de parámetros desde la Base de Conocimiento (BC). Estos parámetros son los almacenados durante la fase de Aprendizaje previa. Su carga se realiza mediante el comando

load ParamAprendizaje

Se recupera toda la información almacenada en la estructura DATOS_CLASES

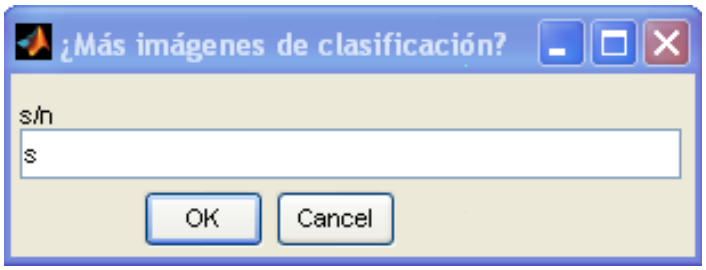
Paso 2: generación del mapa de colores para la representación gráfica de las clases identificadas. Se realiza mediante la función **MapaColores** generando una disposición de los colores a representar. Este mapa es el mostrado en la sección Resultados obtenidos

Paso 3: se pregunta por si se desea reducir la dimensionalidad de la imagen, ya que el proceso es computacionalmente costoso. El diálogo mostrado es el que aparece a continuación

Paso 4: se inicia un proceso iterativo para proceder a la carga de nuevas imágenes con la nomenclatura **IClasificacion_xx.bmp**, donde xx es un valor numérico que se corresponde con el número de iteración. La imagen cargada se muestra en pantalla.

Estas imágenes se procesan píxel a píxel, obteniendo los valores de cada píxel a través de la variable ***píxel***.

Tras el procesamiento de cada imagen se pregunta sobre si se desea realizar más clasificaciones o no mediante el siguiente diálogo, de forma que si se contesta **si** (s, por defecto) se clasifica una nueva imagen.



El resultado de este proceso es el mostrar por pantalla las imágenes clasificadas y los resultados de la clasificación