Sistemas Informáticos
Curso 2006-07

CLASIFICACIÓN DE TEXTURAS NATURALES MEDIANTE TÉCNICAS DE VISIÓN POR COMPUTADOR

Elaborado por:

Álvaro Álvarez Hernández.
Ramón Fernández Buján.
Antonio Herraiz Molina.

Dirigido por:

Prof. Gonzalo Pajares Martinsanz.
Dpto. Arquitectura de Computadores y Automática.

ÍNDICE

1. INTRODUCCIÓN

1.1 RESUMEN DEL PROYECTO.

1.1.1 RESUMEN.

El proyecto realizado consiste en la implementación de tres algoritmos óptimos para la clasificación y reconocimiento de imágenes digitales. El diseño realizado ha sido integrado en una interfaz especializada formando una aplicación completa y eficiente para el tratamiento y clasificación de imágenes. Los algoritmos que mejores resultados han ofrecido y que por lo tanto han sido escogidos para el proyecto han sido los siguientes:

1. Algoritmo Fuzzy Clustering.
2. Algoritmo LLoyd.
3. Algoritmo Bayesiano.

Estos algoritmos han sido implementados sobre una plataforma Java y su diseño se ha realizado siguiendo el patrón de diseño "Factoria". De esta forma se garantiza la fácil integración de cualquier otro algoritmo de clasificación de imágenes al diseño elaborado y una integración uniforme de este diseño a cualquier interfaz especializada.

1.1.2 ABSTRACT.

This project consists in implementation of three optimal algorithms for digital image clasification. The design has been integrated in a specialized interface forming a complete and efficient application for the treatment and classification of images. The best performance has been achived with the following three algorithms. They have been chosen due to this performance:

1. Fuzzy Clustering algorithm.
2. LLoyd algorithm.
3. Bayesian algorithm.

These algorithms have been implemented on a Java platform and its design has been made following the Factory pattern. This way is guaranteed an easy integration of any other images classification algorithm in the elaborated design and a uniform integration of this design in any specialized interface.

1.1.3 PALABRAS CLAVE

Clasificación, Lloyd, Bayessian, fuzzy, cluster, aprendizaje, entrenamiento, memoria, interfaz.

1.2 AUTORIZACIÓN.

Autorizamos a la facultad de Informática de la Universidad Complutense de Madrid, así como al resto de los centros adscritos a la Universidad Complutense de Madrid a la utilización y modificación de todos los componentes que forman este proyecto. Siempre y cuando no se derive en una utilización comercial de cualquiera de los componentes o de parte de ellos.

Firmado:	……	……	……
	Álvaro Álvarez	Ramón Fernández	Antonio Herraiz

1.3 JUSTIFICACIÓN DEL PROYECTO.

Actualmente existe un creciente interés en el desarrollo de procedimientos para el tratamiento de las texturas, siendo la clasificación un tema de especial relevancia.

Este interés tiene su fundamento en algunos aspectos tales como:

1. Control de cultivos en agricultura, propiciado por la necesidad de conocer los cultivos programados para la recepción de subvenciones u otros aspectos relacionados.

2. Cómputo y medición de parcelas agrícolas y tipo de cultivos a los que se dedica.

3. Control de riegos agrícolas.

4. Agricultura de precisión para aplicar herbicida en el tratamiento de malas hierbas de forma selectiva evitando la contaminación medioambiental y la reducción de costes de producción.

5. Evaluación de catástrofes naturales: fuegos, daños por inundaciones, heladas en cultivos agrícolas, nevadas, etc.

6. Detección de cambios en determinadas zonas, principalmente urbanas para el control de edificaciones o impactos medioambientales

7. Vigilancia en prevención de catástrofes, por ejemplo fuegos o inundaciones.

8. Control de fenómenos meteorológicos como es el retroceso de determinadas playas.

9. Vigilancia: forestal, marítima.

10. Detección de infraestructuras: carreteras, caminos forestales, cañadas reales, etc.

En este sentido, diversas empresas u organismos desarrollan o utilizan aplicaciones para abordar dicha problemática. Por citar sólo algunas podemos mencionar

1. Digital Image Processing (Dimap) (http://www.dimap.es/). Las imágenes utilizadas en el proyecto son cortesía del Servicio Territorial de Galicia (SITGA) proporcionadas por Dimap y pertenecientes a la región de Abadín (Lugo) capturadas mediante vuelos aéreos.

2. Proespacio (http://www.proespacio.org/) agrupación de empresas del sector aeroespacial donde una de las actividades destacables son aplicaciones mediante el uso de imágenes de satélite. En este consorcio destacan algunas empresas líderes del sector tanto en España como a nivel Internacional: EADS-Espacio, EADS Astrium, CRISA, GMV, Indra Espacio, Sener, Hispasat, IberEspacio, Inasmet, Insa, Mier, NTE, Tecnológica, Rymsa, Hispasat, GTD, Alcatel, CRISA, GTD.

3. Organismos oficiales y Centros de Investigación :

- Consejo Superior de Investigaciones Científicas (CSIC)
- Instituto Nacional de Técnica Aeroespacial (INTA)
- Centro de Estudios y Experimentación de Obras Públicas (CEDEX), con el que existen trabajos de colaboración previos por parte de uno de los directores del trabajo (Pajares y col. 2001, Pajares y col. 2002)

La mayoría de las empresas e instituciones anteriormente mencionadas utilizan para llevar a cabo sus aplicaciones principalmente herramientas comerciales tales como:

1 ERDAS
2 Intergraph
3 ENVI-IDL
4 ILOG
5 PCI
6 E-Cognition

Aunque bien es cierto que cada día es mayor la potencialidad de las herramientas mencionadas no es menos cierto que los retos tecnológicos derivados de las aplicaciones mencionadas hacen que en algunos casos la utilización de tales herramientas sea insuficiente para abordar las propuestas de proyectos demandados por los clientes. Este es el caso para una gran parte de las aplicaciones mencionadas en el tratamiento de las texturas naturales donde la **clasificación** de las mismas surge como una tarea fundamental.

En efecto, en general las mencionadas herramientas implementan algunos de los métodos clásicos de clasificación, siendo necesaria la intervención del usuario mediante programación para abordar algunas de las tareas que involucran aspectos de clasificación de texturas, lo cual no siempre es factible o al menos en la medida que cabría esperar debido a múltiples limitaciones.

Además, y lo que es más importante, en muchos casos no existe la posibilidad de llevar a cabo la investigación necesaria para abordar la problemática particularmente cuando los métodos clásicos no producen los resultados esperados.

Por todo lo expuesto anteriormente surge una necesidad importante en el ámbito de las aplicaciones reales para abordar el tema de la clasificación de texturas naturales en imágenes y un reto para la comunidad científica para tratar de mejorar los procedimientos existentes con la mayor flexibilidad posible.

1.4 OBJETIVOS.

Los objetivos inicialmente establecidos en este proyecto fueron:

- Investigación y localización de los tres algoritmos que mejores resultados obtengan en la clasificación de imágenes digitales en texturas naturales.

- Implementación óptima y eficiente de los algoritmos, garantizando un tiempo de ejecución reducido y un uso de memoria óptimo.

- Diseño modular y genérico de un paquete que incluya los tres algoritmos implementados, y que garantice una fácil integración a cualquier interfaz para la posible incorporación de futuros métodos.

1.5 ESTRUCTURA Y ORGANIZACIÓN DE LA MEMORIA.

Esta memoria se organiza en siete secciones, a continuación explicamos el contenido de cada una de ellas:

Sección 1: Es la sección actual hace una introducción al proyecto realizado y la justificación de este.

Sección 2: Explica la planificación del proyecto.

Sección 3: Hace un análisis de los métodos de implementación incluyendo los requisitos funcionales derivados y los riesgos previstos.

Sección 4: Plantea el diseño de la aplicación siguiendo la metodología UML.

Sección 5: Explica las diferentes pases de desarrollo a partir del diseño previo.

Sección 6: Se presentan los resultados obtenidos en el proyecto.

Sección 7: Se presentan las conclusiones más revelantes y las previsiones de futuro.

2 PLANIFICACIÓN.

La planificación para realizar una aplicación son las fases a seguir en la aplicación para conseguir los objetivos propuestos y la organización adoptada para conseguirla, por lo tanto supone una parte muy importante del proyecto. En esta sección explicaremos la organización adoptada para el desarrollo de la aplicación y el plan de fase establecido para conseguir los objetivos iniciales.

2.1 ORGANIZACIÓN.

El proyecto basado en algoritmos de tratamiento y clasificación de imágenes forma parte de una aplicación en la cual se puede distinguir dos partes muy diferenciadas: el interfaz y los algoritmos. Cada una de las partes está realizada bajo dos proyectos diferentes, debiendo complementarse correctamente para integrar ambos en la aplicación común. Por ello para describir la organización global explicaremos la organización establecida a nivel de aplicación y la organización interna del proyecto.

2.1.1 ORGANIZACIÓN DE LA APLICACIÓN.

Para facilitar una correcta integración entre las dos partes que conforman la aplicación común se tomaron las siguientes acciones y decisiones:

1. Elaboración de un plan de fase uniforme entre los dos proyectos, de forma que los objetivos y fechas establecidas en cada una de las fases de ambos proyectos estén sincronizadas para cumplir los objetivos establecidos en ambas. Con ello se trata de evitar influencias y retrasos inducidos de un proyecto sobre el otro.

2. Establecimiento de una fechas para la reunión de los componentes de cada uno de los proyectos para comentar los

objetivos y resultados efectuados hasta la fecha, los problemas surgidos por dependencias entre ambos proyectos y la solución para estas. Estas fechas se establecieron al inicio y finalización de cada una de las fases de desarrollo del proyecto, teniendo en cuenta la disponibilidad de los miembros integrantes de ambos proyectos.

3. Creación de una herramienta común para la comunicación de ambos proyectos. Para ello se creó un grupo MSN para comunicar posibles problemas surgidos de forma puntual o para informar de la aparición de algún problema.

2.1.2 ORGANIZACIÓN DEL PROYECTO.

El proyecto explicado en esta memoria ha sido realizado por un grupo de tres alumnos: Álvaro Álvarez Hernández, Antonio Herraiz Molina y Ramón Fernández Bujan. La investigación acerca del grado de eficiencia así como la integración con el interfaz ha sido realizada conjuntamente entre los tres componentes. Una vez estudiados los algoritmos cada uno de los integrantes del grupo se encargó del análisis, desarrollo e implementación de un algoritmo específico; éstas fueron las correspondencias establecidas:

Álvaro Álvarez Hernández: algoritmo Bayesiano.
Antonio Herraiz Molina: algoritmo Fuzzy Clustering.
Ramón Fernández Bujan: algoritmo Lloyd

2.2 PLAN DE FASE.

El plan de fase establecido para el desarrollo del proyecto ha sido diseñado siguiendo el modelo de proceso unificado de Rational estudiado en Ingeniería del software. Este modelo está dividido en cuatro fases: fase de inicio, fase de elaboración, fase de construcción y fase de pruebas. Cada una de estas fases estará compuesta por una o varias iteraciones establecidas previamente.

2.2.1 FASE DE INICIO.

Esta fase está compuesta por una iteración con las siguientes características:

Fecha de inicio: 10 – 10 - 2006
Fechas de conclusión: 1 – 11 - 2006

Objetivos:

- Concreción con el tutor del alcance y objetivos del proyecto.
- Análisis de riesgos.
- Captura de requisitos de acuerdo con los objetivos acordados con el profesor.
- Establecimiento del plan de fase del proyecto. La planificación fijada estará coordinada con la fijada por el otro grupo que participa en la aplicación.
- Elección del lenguaje de programación a utilizar para realizar el proyecto. Dicha elección será llevada a cabo entre los dos grupos que estamos realizando la aplicación. Optando finalmente por JAVA.
- Aprendizaje y familiarización con el lenguaje establecido así como con las bibliotecas específicas que se van a usar en la aplicación, en este caso Java Advanced Imagining particularmente.

- Investigación y documentación de los algoritmos de clasificación a implementar.
- Selección de objetivos que se deben cumplir en la siguiente fase, en función de los resultados obtenidos en ésta.

2.2.2 FASE DE ELABORACIÓN.

Esta fase está compuesta por una iteración con las siguientes características:

Fecha de inicio: 2 – 11 - 2006
Fechas de conclusión: 1 – 12 - 2006

Objetivos:

- Diseño global de la aplicación. Este diseño se realizará de forma modular y genérica, facilitando así su integración en cualquier tipo de interfaz especializada y en particular en la realizada por el otro grupo.
- Especificación de casos de uso y de los servicios prestados por la aplicación.
- Realización de un prototipo muy simple utilizando otros métodos de clasificación ya interpretados para comprobar el buen funcionamiento del diseño escogido.
- Elaboración de la documentación correspondiente a esta fase.
- Selección de objetivos que se deben cumplir en la siguiente fase, en función de los resultados obtenidos en ésta.

2.2.3 FASE DE CONSTRUCCIÓN.

Esta fase es un poco más larga que las anteriores por ello está formada por dos iteraciones.

2.2.3.1 ITERACIÓN 1

Fecha de inicio: 2 – 12 -2006

Fechas de conclusión: 20 – 1 - 2007

Objetivos:

- Reunión con el tutor y los componentes del otro grupo para explicarles el diseño elegido y las principales características del diseño previsto en este proyecto.
- Concreción de los puntos de comunicación entre el diseño escogido para su integración en el interfaz. Este apartado será realizado entre los dos grupos del proyecto definiéndose la estructura y el formato de los puntos comunes.
- Revisión de la especificación de requisitos para futuras implementaciones que puedan surgir ante nuevas necesidades.
- Revisión y seguimiento de los riesgos detectados en el proyecto hasta el momento.
- Investigación y búsqueda de soluciones para los riesgos más importantes. En nuestro caso se centra en la optimización de memoria y tiempo de ejecución en los algoritmos de clasificación y en la investigación para aumentar el tamaño de memoria dinámica disponible en tiempo de ejecución denominado 'HEAP', que usa el lenguaje de programación elegido para desarrollar los algoritmos (JAVA).
- Elaboración del primer prototipo de la aplicación final y revisión de lo realizado hasta ahora.

2.2.3.2 ITERACIÓN 2

Fecha de inicio: 23 – 2 - 2007
Fechas de conclusión: 20 – 3 - 2007

Objetivos:

- Elaboración del prototipo final de la aplicación. En este prototipo se incorporarán las soluciones encontradas para los puntos críticos de nuestra aplicación: tiempo de ejecución y memoria.
- Integración del prototipo final al último prototipo elaborado por el grupo encargado de la interfaz.
- Presentación del prototipo final de la aplicación, una vez que se han integrado ambas partes al tutor.
- Modificación de los errores o deficiencias encontradas por el tutor, al prototipo o versión final de nuestra parte de la aplicación. Esta será la versión final de nuestro proyecto.
- Elaboración de la documentación correspondiente a la fase de construcción.
- Selección de objetivos que se deben cumplir en la siguiente fase.

2.2.4 FASE DE TRANSICIÓN.

Esta fase está compuesta por una iteración, y tiene las siguientes características:

Fecha de inicio: 21 – 3 - 2007
Fechas de conclusión: 8 – 6 - 2007

Objetivos:

- Diseño de un plan de pruebas para comprobar el correcto funcionamiento de la aplicación. Estas pruebas se efectuarán con diferentes tamaños de imágenes, y verificarán si la solución obtenida es correcta y si el tiempo de ejecución es el requerido.
- Búsqueda de mejoras para el comportamiento de nuestra aplicación.
- Diseño de un plan de pruebas común para comprobar el funcionamiento de la aplicación una vez integradas las dos partes de ambos proyectos.
- Búsqueda de mejoras para el comportamiento de la aplicación conjunta.
- Elaboración de la documentación correspondiente a esta fase y de la elaboración conjunta a presentar por los dos proyectos que forman la aplicación.
- Revisión de documentos realizados hasta la fecha.
- Diseñar un esquema de la memoria principal del proyecto y presentársela al tutor para dar su aprobación e indicar sus deficiencias.
- Realización de la memoria final del proyecto acorde con las recomendaciones sugeridas por el tutor y utilizando los documentos anteriormente elaborados.
- Elaboración del CD para la entrega final con la versión definitiva de nuestro proyecto.

3 ANÁLISIS

El proyecto que se presenta se fundamenta en el estado de implementación de tres algoritmos clásicos de clasificación cuyos resultados han sido satisfactorios. En esta sección se estudian los aspectos derivados de la implementación de dichos métodos realizando una especificación de los mismos. Concluyendo en los requisitos y riesgos derivados.

Los tres algoritmos propuestos constan fundamentalmente de dos fases: entrenamiento y clasificación. En la primera se aprenden unos parámetros o estructuras según el método, que son utilizados en la segunda.

3.1 ESPECIFICACIÓN DEL PROYECTO.

3.1.1 ALGORITMO LLOYD

3.1.2.1 IDEA GENERAL

El método propuesto a continuación es una versión modificada del Algoritmo General Lloyd (AGL) original y se conoce como algoritmo de aprendizaje competitivo no supervisado en la literatura especializada sobre redes neuronales.

La red básica de aprendizaje competitivo consta de una única capa de neuronas, de forma que hay tantas neuronas como número de clases haya especificado el usuario. Cada nodo de salida o neurona representa una categoría de patrones. Cada neurona tiene asociado un centro representativo. Estos centros constituyen el objeto del aprendizaje en este método.

La capa de neuronas mencionadas va a estar implementada por un único vector de tantas componentes como neuronas tenga nuestra red. Cada componente del vector corresponde al centro asociado a la neurona que esté representado.

3.1.2.2 ESPECIFICACIÓN FORMAL

INICIALIZACIÓN DE CENTROS

Como primer paso para la clasificación, la red neuronal de aprendizaje competitivo no supervisado requiere que sus centros sean inicializados previamente. Al basar la clasificación en los centros de la red, este paso adquiere una notable importancia, ya que el buen resultado de la clasificación está directamente relacionado con la calidad de los centros iniciales. Es decir, cuanto mejor sean los valores iniciales de los centros mejores resultados se obtendrán en la fase de clasificación.

Entendemos por buenos centros aquellos que representen clases cuyos elementos (patrones) estén bien diferenciadas entre sí. De forma que a un patrón se le pueda asociar una clase sin ningún tipo de duda.

Para inicializar dichos centros se ofrecen varias técnicas las cuales describimos de mayor a menor relevancia:

- ***Fuzzy Clustering****:* Esta técnica de inicialización hace uso del método Fuzzy para obtener una agrupación de píxeles en las clases existentes. De dicha agrupación se obtiene una especie de media para cada clase obteniendo como resultado el valor inicial del centro asociado a dicha clase. *Para detalles ver sección 3.2.2.2.*
- ***Random****:* Esta técnica se basa en generar los valores iniciales de los centros de forma aleatoria. La calidad de esta técnica esta sujeta al azar, por ello no es muy recomendable, ya que no asegura el buen funcionamiento del algoritmo en su conjunto. La ventaja de esta técnica, y principal motivo de su implementación, es que la complejidad de cálculo es muy inferior al *Fuzzy Clustering* y por tanto la velocidad de cálculo es notablemente superior.

Aparte de estas técnicas hemos contemplado la posibilidad de que el usuario haga uso de valores de centros ya creados en procesos de aprendizaje previos.

ENTRENAMIENTO

Esta fase adquiere su importancia, por el hecho de tener el objetivo de mejorar los centros ya existentes con el fin de mejorar los resultados de la clasificación.

La red no supervisada adapta sus centros en base a la regla de aprendizaje competitivo, que hace que las neuronas **compitan por el derecho a responder** por ellas mismas a un determinado tipo de entrada. Esto se puede ver como un sistema muy sofisticado de clasificación, cuyo objetivo es dividir un conjunto de patrones de entrada en un número de clases tal que los patrones de la misma clase exhiben un cierto grado de similitud.

Para evaluar el grado de similitud de los patrones. Se proponen medidas de distancias, de las cuales las dos siguientes son ampliamente utilizadas:

1. Producto interno:

$$\vec{A} \cdot \vec{B} = |\vec{A}||\vec{B}|cos\theta$$

2. Distancia Euclídea con Centros:

$$D(\mathbf{x}(k), \boldsymbol{c}_j(k)) = \left\|\mathbf{x}(k) - \boldsymbol{c}_j(k)\right\|^2$$

En este caso hemos optado por el más común, que es la distancia Euclídea entre el patrón de entrada y los centros existentes en la red. Los patrones de entrada representan cada uno de los píxeles que forman la imagen, por consiguiente, los patrones de entrada y los centros van a ser vectores de tres componentes (R, G, B). Cada píxel se describe mediante un sector

de la forma X =(R, G, B). Los centros de los clústeres al aprender tienen esta misma estructura.

El algoritmo de aprendizaje es básicamente como se detalla a continuación y pudiéndose realizar un proceso iterativo tantas veces como el usuario indique.

1) Inicio: dados los puntos de datos $\mathbf{x}(k)$, $k = 1,2,...,$ y centros de salida iniciales $\boldsymbol{c}_j(0)$, $j = 1,\ldots,m$, siendo m el número de clases.

2) Determinar el centro $\boldsymbol{c}_j(k)$ más próximo al punto $\mathbf{x}(k)$

$$j = \arg\min_j \left\| \mathbf{x}(k) - \boldsymbol{c}_j(k) \right\|^2$$

3) Actualizar el centro de salida utilizando las ecuaciones,

$$\boldsymbol{c}_j(k_j + 1) = \boldsymbol{c}_j(k_j) - \gamma(k_j)\,\mathrm{grad}\left(\left\| \mathbf{x}(k) - \boldsymbol{c}_j(k_j) \right\|^2 \right)$$
$$k_j = k_j + 1$$

Obsérvese que cada centro podría tener su propia razón de aprendizaje, lo que se indica con k_j en $\gamma(k_j)$, con $j = 1,\ldots,m$. El gradiente se calcularía como,

$$\frac{\partial}{\partial \boldsymbol{c}_j} \left\| \mathbf{x} - \boldsymbol{c}_j \right\|^2 = 2(\mathbf{x} - \boldsymbol{c}_j)$$

La actualización del centro, sólo se lleva acabo en el caso de que la diferencia entre el nuevo centro y el antiguo, supera la tolerancia establecida por el usuario (por defecto su valor es de 1e-10).

No obstante en la implementación actual hemos optado por una razón de aprendizaje común, la cual es introducida por el

usuario, por defecto su valor es de 0.1.Los centros de salida se actualizan como sigue.

$$\boldsymbol{c}_j(k_j+1) = \boldsymbol{c}_j(k_j) + \gamma(k_j)\left[\mathbf{x}(k) - \boldsymbol{c}_j(k_j)\right]$$
$$k_j = k_j + 1$$

CLASIFICACIÓN

Los procedimientos de clasificación no supervisados se basan a menudo en algunas técnicas de clasificación, que forman grupos de patrones parecidos. Para realizar la clasificación es necesario definir, de nuevo, una distancia o medida de similitud. Nosotros seguimos basándonos en la distancia Euclídea.

1) Inicio: dados los puntos de datos $\mathbf{x}(k)$, $k = 1,2,...,$ y centros de salida ya actualizados $\boldsymbol{c}_j$, $j = 1,\ldots,m$, siendo m el número de clases.

2) Determinar el centro $\boldsymbol{c}_j$ más próximo al punto $\mathbf{x}(k)$

$$j = \arg\min_j \left\|\mathbf{x}(k) - \boldsymbol{c}_j(k)\right\|^2$$

Una vez conocido el centro más próximo la entrada pasa a ser clasificada como perteneciente a la asociada con j.

3.1.2.3 ANÁLISIS DE PARÁMETROS

Estos son los parámetros utilizados en el algoritmo Lloyd. Estos datos los define cada usuario que ejecute dicho algoritmo con unos valores apropiados para su aplicación.

1. Número de clústeres o clases ´c´.

Parámetro que establece el número de particiones o clases en que se desea clasificar el conjunto de datos iniciales o muestras. El valor de este parámetro depende del número de clases en que el usuario quiera clasificar la imagen o imágenes.

2. Numero de iteraciones 'NUMITER'

Este parámetro se corresponde con el número de veces que el usuario quiera repetir la fase de aprendizaje. Cuanto mayor sea este número los centros obtendrán mejores valores finales, y por tanto una mejor clasificación posterior. Por el contrario al aumentar el número de iteraciones se aumenta la complejidad del algoritmo, y por tanto disminuirá la velocidad de cálculo de este.

3. Error permitido o tolerancia ´ε´

Parámetro utilizado en el entrenamiento, se encarga de definir el grado de precisión de los centros obtenidos por el entrenamiento. Por tanto cuanto mayor sea su valor menor será la precisión de la clasificación. Este parámetro establece un umbral al número de iteraciones, ya que llegado al punto en el cual la diferencia entre el nuevo centro y el actual no supere el valor de tolerancia, la actualización de dicho centro no se llevará acabo. Su valor por defecto es 1e-10.

4. Razón de aprendizaje $'\gamma'$:

La "cantidad"' de la corrección de cada centro va a estar determinada por este parámetro. Cualquier función de corrección utiliza la llamada ***función de ganancia*** o ***razón de aprendizaje***, definida como γ: $N \rightarrow R$. La única restricción sobre esta función es que debe proporcionar una secuencia monótona que cumpla $0 < \gamma(t) < 1$. Valores altos de $\gamma(t)$ hacen que el prototipo se acerque mucho al patrón y valores bajos hacen que el acercamiento sea más moderado. En nuestro caso hemos optado por fijar un valor constante. Su valor por defecto es 0.1.

3.1.2 ALGORITMO C-MEDIAS FUZZY

3.1.2.1 IDEA GENERAL

El algoritmo C-medias fuzzy es un algoritmo iterativo de clasificación basado en técnicas de lógica fuzzy o borrosa. Este algoritmo consta de dos fases: un entrenamiento (no supervisado) y una de clasificación.

En el entrenamiento inicialmente se señalan arbitrariamente los centros de las clases, de acuerdo a un número de clases indicadas por el usuario. Los píxeles se asignan al centro más cercano, y se vuelven a calcular los nuevos centros. Este proceso se repite hasta alcanzar un número máximo de iteraciones, o hasta conseguir que la modificación entre los valores de pertenencia asignados entre dos iteraciones diferentes sea menor que una determinada tolerancia especificada como parámetro de entrada. Este método utiliza la distancia espectral mínima para asignar cada píxel a un centro candidato. Una vez concluido el proceso anterior el valor de los centros resultantes va a ser el utilizado para realizar la clasificación de las imágenes deseadas.

El proceso de clasificación consiste en la obtención del denominado valor de pertenencia de cada uno de los píxeles que componen la imagen con respecto a cada una de las clases definidas, para ello se calculará la distancia de este píxel al centro correspondiente a cada clase. El píxel será asignado a la clase que mayor valor de pertenencia ha obtenido anteriormente.

3.1.2.2 ESPECIFICACIÓN FORMAL

FUZZY CLUSTERING

El objetivo de la técnica de agrupamiento conocida como *Fuzzy Clustering* consiste en dividir *n* patrones $x \in X$ caracterizados por *p* propiedades en *c* "clusters" o grupos. Supongamos el conjunto de datos $X = \{x_1, x_2, ..., x_n\} \in \Re^p$ un subconjunto del espacio real *p*-dimensional $\Re^p$. Cada patrón $x_k = \{x_{k_1}, x_{k_2}, ..., x_{k_p}\} \in \Re^p$ se denomina vector de características, x_{k_j} es la *j*-ésima característica de la observación x_k. En nuestro caso concreto *X* va a ser el conjunto de todos los píxeles de cada imagen y los x_i van a pertenecer a $\Re^3$, ya que cada píxel va estar caracterizado por las propiedades R,G,B por tanto ´*p*´ es igual a tres. El número de clusters ´c´ será especificado por el usuario, siendo variable dependiendo del tipo de imagen a clasificar.

Puesto que los elementos de un cluster deben ser tan similares entre sí como sea posible y a la vez deben ser tan diferentes a los elementos de otros clústeres como también sea posible, el proceso de clustering se controla por el uso de medidas de similitud basadas en distancias. Así la similitud o la diferencia entre dos puntos x_k y x_l puede interpretarse como la distancia entre esos puntos.

Una distancia entre dos objetos x_k y x_l es una función que toma valores reales $d : XxX \to R^+$ cumpliendo:

$d(x_k, x_l) = d_{kl} \geq 0$; $d_{kl} = 0 \Leftrightarrow x_k = x_l$; $d_{kl} = d_{lk}$ y $d_{kl} \leq d_{kj} + d_{jl}$

Si se desea realizar una partición del conjunto *X* en *c* clústeres tendremos $S_i\{i=1,...,c\}$ subconjuntos. A partir de esta consideración se define lo que se conoce como grado de pertenencia μ_{ik} de cada objeto x_k al subconjunto S_i. Cada uno de estos S_i será un conjunto fuzzy por lo cual un objeto puede pertenecer a diferentes subconjuntos. Por lo tanto se puede decir que x_k pertenece a un conjunto S_i con grado de pertenencia μ_{ik} y a S_j con grado de pertenencia μ_{ij}. Todos los valores de referencia tomados para cada x_k están dentro del intervalo continuo [0,1].

Dado $X=\{x_1,x_2,...,x_n\}$ y el conjunto V_{cn} de todas las matrices reales de dimensión *c x n*, con $2 \le c < n$. Se puede obtener una matriz representando la partición de la siguiente manera $U=\{\mu_{ik}\} \in V_{cn}$. Esta matriz cumple las siguientes condiciones:

1) $\mu_{ik} \in \{0,1\}$ *crisp* o $\mu_{ik} \in [0,1]$ *fuzzy* $\quad 1 \le i \le c;\ 1 \le k \le n$

2) $\sum_{i=1}^{c} \mu_{ik} = 1 \qquad 1 \le k \le n$

3) $0 < \sum_{k=1}^{n} \mu_{ik} < n \qquad 1 \le i \le c$

La localización de un clúster S_i se representa por su centro $v_i = \{v_{i_1}, v_{i_2},...,v_{i_p}\} \in \Re^p$ con *i* = 1,…,*c*, alrededor del cual se concentran los objetos. La definición básica de llevar a cabo el problema de la partición fuzzy para *m* > 1 consiste en minimizar la siguiente función objetivo según la siguiente expresión:

$$\min z_m(U;v) = \sum_{k=1}^{n}\sum_{i=1}^{c} \mu_{ik}^m \|x_k - v_i\|_G^2$$

donde G es una matriz de dimensión p x p que es simétrica y definida positiva. Así se puede definir una norma general de este tipo:

$$\|x_k - v_i\|_G^2 = (x_k - v_i)^t G(x_k - v_i)$$

Diferenciando la función objetivo para v_i (suponiendo constante U) y μ_{ik} (suponiendo constante v) y aplicando la condición de que $\sum_{i=1} \mu_{ik} = 1$, se obtiene:

$$v_i = \frac{1}{\sum_{k=1}^{n} (\mu_{ik})^m} \sum_{k=1}^{n} (\mu_{ik})^m x_k \qquad i = 1,...,c \tag{1}$$

$$\mu_{ik} = \frac{\left(\frac{1}{\|x_k - v_i\|_G^2} \right)^{2/m-1}}{\sum_{j=1}^{c} \left(\frac{1}{\|x_k - v_i\|_G^2} \right)^{2/m-1}} \qquad i = 1,...,c; k = 1,...,n \tag{2}$$

donde el exponente m es un parámetro que se conoce como peso exponencial.

ENTRENAMIENTO

El algoritmo utilizado para el entrenamiento es el C-Medias Fuzzy. Este algoritmo de agrupamiento no supervisado

está basado en el Fuzzy Clustering explicado anteriormente. A partir de unas muestras iniciales, que en nuestro caso serán los píxeles para el entrenamiento, obtiene los centros correspondientes de los clústeres especificados.

Estos son los pasos establecidos en el algoritmo para obtener los centros requeridos:

1) Elegir c ($2 \le c \le n$), m ($1 < m < \infty$) y la matriz G de dimensión p x p siendo simétrica y definida positiva. Inicializar $U^{(0)}$ y poner $t = 0$. En nuestro algoritmo G es la matriz identidad debido a que la norma matricial elegida va a ser la norma euclídea, en caso de cambiar la norma el valor de G sería distinto.

2) Inicialización pseudo-aleatoria de cada uno de los centros de las clases v_i con $0 < i \le c$.

3) Calcular los c centros fuzzy de los clústeres a partir de (1) $\{v_i^{(t)}\}$ utilizando $U^{(t)}$.

4) Calcular los nuevos grados de pertenencia de la matriz $U^{(t+1)}$ utilizando $\{v_i^{(t)}\}$ a partir de la condición (2) si $x_k \neq v_i^{(t)}$. De lo contrario

$$\mu_{jk} = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

5) Elegir una norma matricial, que en nuestro algoritmo ha sido la norma euclídea, y calcular $\Delta = \left\| U^{(t+1)} - U^t \right\|_G$. Si $\Delta > \varepsilon$ poner $t = t + 1$ y regresar al paso 2), de lo contrario detener el proceso.

Para limitar el tiempo de ejecución del algoritmo se pondrá un límite de iteraciones de forma que aunque se cumpla $\Delta > \varepsilon$ se detendrá el proceso.

El valor de los centros obtenido tras la finalización del proceso será posteriormente utilizado para realizar la clasificación.

CLASIFICACIÓN

El algoritmo de clasificación fuzzy utilizado en este proyecto está basado en la norma euclídea.

Los pasos realizados para cada una de las muestras a clasificar, que en nuestro caso serán píxeles son las siguientes:

1. Calculo de la distancia de cada muestra a cada centro de los clústeres. Para calcular esa distancia utilizaremos la distancia euclídea:

$$\|x_k - v_1\|_G^2$$

2. Calculo del grado de pertenencia de la muestra a cada uno de los clústers o clases, para ello sustituiremos el valor dado en el paso 1, en la ecuación (1) , dando el valor del grado de pertenencia para cada una de las clases.

3. Clasificación de la muestra con aquel clúster que obtenga un mayor valor de la función de pertenencia.

3.1.2.3 ANÁLISIS DE PARÁMETROS

Estos son los parámetros utilizados en el algoritmo C-Medias Fuzzy. Estos datos los define cada usuario que ejecute dicho algoritmo con unos valores apropiados para su aplicación.

3. Número de clústeres o clases ´*c*´.

Parámetro que establece el número de particiones o clases en que se desea clasificar el conjunto de datos iniciales o muestras. El valor de este parámetro depende del número de clases en que el usuario quiera clasificar la imagen o imágenes.

3. Peso exponencial ´*m*´

Este parámetro reduce la influencia del ruido cuando se obtienen los centros de los clústeres, reduce la influencia de los valores pequeños de μ_{ik} (puntos lejos de v_i) frente a valores altos de μ_{ik} (puntos cerca de v_i). Cuanto mayor sea $m > 1$ mayor es dicha influencia. El valor típico para este parámetro es 2.

3. Error permitido o tolerancia ´ε´

Parámetro utilizado en el entrenamiento, este parámetro se encarga de definir el grado de precisión de los centros obtenidos por el entrenamiento. Por lo tanto a valores muy grandes de este parámetro la precisión será muy poca, en caso de establecer un número muy bajo el tiempo de ejecución del entrenamiento puede ser muy elevado al tener que ejecutar un gran número de veces el bucle principal del algoritmo del entrenamiento, para controlar este número de iteraciones utilizaremos el parámetro 'NumMax' que indica el número máximo de iteraciones posibles. El valor típico para el error permitido suele ser en torno a 0.02.

5 Número máximo de iteraciones ‘NumMax’.

Parámetro utilizado en el entrenamiento para establecer un número máximo de iteraciones del bucle principal utilizado en este algoritmo. Con esto conseguimos establecer un tiempo máximo de ejecución, evitando ejecuciones muy largas debido a una tolerancia extremadamente pequeña o debido a la dificultad de convergencia del algoritmo provocada por una mala elección de los datos de muestra. El valor por defecto establecido para este parámetro es 10.

3.1.3 ALGORITMO BAYESIAN0.

3.1.2.1 IDEA GENERAL

La teoría de la decisión de Bayes es un método estadístico clásico en clasificación de patrones. Se basa en el supuesto de que el problema de la decisión se enfoca en términos probabilísticos y que todas las probabilidades relevantes resultan conocidas.

El entrenamiento para este algoritmo consistirá en calcular medias y matrices de covarianza de cada clase de las muestras proporcionadas por el entrenamiento C-Medias Fuzzy.

Por tanto la clasificación de cada píxel de una imagen se enfoca según la probabilidad de pertenecer a cada una de ellas. Esa probabilidad vendrá dada en nuestro caso por la distancia de Mahalanobis que a su vez se calcula con los datos obtenidos en el entrenamiento.

3.1.2.2 ESPECIFICACIÓN FORMAL

El algoritmo de clasificación Bayesiano en el ámbito de la clasificación de imágenes nos será de utilidad para analizar una imagen píxel a píxel y determinar a qué clase pertenece cada uno según una probabilidad calculada.

Supongamos que no tenemos conocimiento sobre los valores R, G y B del píxel que estamos analizando. Utilizando la terminología de la teoría de la decisión, podemos decir que la probabilidad a priori de que un píxel pertenezca a cualquiera de las clases es la misma y cuya suma es la unidad.

Sea ***y*** el estado que designa la pertenencia de un píxel a una de las clases, ***c*** el número de clases en que hay que dividir la imagen y considerando a ***y*** una variable aleatoria, ya que la pertenencia a uno de los dos estados es impredecible, podemos decir de forma más precisa que suponemos que existe alguna *probabilidad a priori* $P(y=c_i)$ de que el píxel pertenezca a la clase c_i y alguna *probabilidad a priori* de que pertenezca a la clase c_j $P(y=c_j)$ y así con todas las clases que queramos hasta ***c***.

A la hora de decidir acerca de la pertenencia de un píxel a una clase concreta, la única información son esas probabilidades a priori y parece razonable utilizar la siguiente *regla de decisión*: decidir la que tenga mayor probabilidad.

Pero en nuestro caso conocemos los valores para R, G y B del píxel a clasificar. A estos tres valores los llamaremos $\mathbf{x} = \{x_R, x_G, x_B\}$**.** Así, diferentes píxeles darán diferentes valores de **x**, resulta natural por tanto expresar esta variabilidad en términos probabilísticos. En este sentido, consideremos a **x** una variable aleatoria continua cuya distribución depende de la clase.

Sean $p(\mathbf{x}/y=c_j)$ las funciones de *densidad de probabilidad condicionales* para **x** dado que el píxel pertenezca a la clase c_i. Supongamos que conocemos tanto las probabilidades a priori como estas últimas funciones de densidad de probabilidad. Suponer además que medimos las componentes R, G y B de un píxel y descubrimos que vale **x**, la pregunta será ¿cómo influye esta medida sobre nuestra actitud con relación a la clase de que se trata? La respuesta nos la proporciona la *regla de Bayes*, considerando que tanto las probabilidades a priori $P(y=c_j)$ como las densidades condicionales para cada clase $p(\mathbf{x}/y=c_j)$ son conocidas o se pueden estimar, es posible determinar para una observación dada **x** la probabilidad de que esa observación

pertenezca a una determinada clase. Estas probabilidades, llamadas *probabilidades a posteriori* pueden usarse para construir una regla discriminante:

$$p(y = c_j \ / \mathbf{x}) = \frac{p(\mathbf{x} / y = c_j) P(y = c_j)}{p(\mathbf{x})}$$

donde

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x} / y = c_j) P(y = c_j)$$

La regla de Bayes muestra cómo la observación del valor **x** cambia las probabilidades a priori a las *probabilidades a posteriori*
$p(y = c_j / \mathbf{x})$.

Una vez que se determinan esas probabilidades a posteriori, la siguiente regla de decisión se utiliza para clasificar **x**.

$\mathbf{x} \in c_i \ \ sii \ \ P(y = c_i/\mathbf{x}) > P(y = c_j/\mathbf{x}) \ \ \forall\, i \neq j, \quad i, j = 1,2,..c$

Ahora bien, si nos fijamos en el segundo término de la expresión que obtiene las *probabilidades a posteriori* del teorema de Bayes y eliminado el término no discriminante $p(\mathbf{x})$(no aporta nada en la decisión), se tiene una forma alternativa de clasificar el vector de atributos **x**:

$\mathbf{x} \in c_i \ \ sii \ \ P(\mathbf{x} / y = c_i) P(y = c_i) > P(\mathbf{x} / y = c_j) P(y = c_j) \ \ \forall i \neq j, \ i, j = 1,2,..,c$

Generalmente las distribuciones de densidad de probabilidad se eligen Normales o Gaussianas. Un caso especial surge cuando las probabilidades a priori son iguales para todas las

clases, ya que en esta situación la distancia de Mahalanobis se puede utilizar como función discriminante mediante la siguiente regla de decisión a partir de la regla anterior y teniendo en cuenta el signo negativo en el término exponencial de la función de densidad de probabilidad Normal, así

$$\mathbf{x} \in c_i \;\; sii \;\; d_M^2(\mathbf{x},\mathbf{m}_i) < d_M^2(\mathbf{x},\mathbf{m}_j) \;\; \forall i \neq j, \; i,j = 1,2,..,c$$

donde $\mathbf{m}_i$, $\mathbf{m}_j$ son los vectores media de las clases c_i y c_j respectivamente.

Sin pérdida de generalidad, la distancia de un vector $\mathbf{x}_k$ a la clase c_i resulta ser:

$$d_M^2(\mathbf{x}_k,\mathbf{m}_i) = (\mathbf{x}_k - \mathbf{m}_i)^{\mathrm{t}} C_i^{-1} (\mathbf{x}_k - \mathbf{m}_i)$$

En el supuesto de que las matrices de covarianza sean la identidad, la distancia de Mahalanobis al cuadrado resulta ser la distancia Euclídea al cuadrado, en cuyo caso tendríamos,

$$d_E^2(\mathbf{x},\mathbf{m}_i) = (\mathbf{x} - \mathbf{m}_i)^t (\mathbf{x} - \mathbf{m}_i) = \mathbf{x}^t\mathbf{x} - 2\mathbf{x}^t\mathbf{m}_i + \mathbf{m}_i^t\mathbf{m}_i$$

En la expresión anterior el término $\mathbf{x}^t\mathbf{x}$ no discrimina, ya que se repite en todas las clases, de forma que puede despreciarse. Ahora, si se cambia de signo y se divide por 2 en la ecuación anterior se obtiene la siguiente función discriminante, donde se deduce que la distancia Euclídea al cuadrado mínima hace la expresión siguiente máxima:

$$fd_i(\mathbf{x}) = \mathbf{x}^{\mathrm{t}}\mathbf{m}_{\mathrm{i}} - \frac{1}{2}\mathbf{m}_{\mathrm{i}}^{\mathrm{t}}\mathbf{m}_{\mathrm{i}}$$

ENTRENAMIENTO

El entrenamiento, como paso previo a la clasificación Bayesiana consistirá en obtener las medias y matrices de covarianza de las muestras.

Una muestra consiste en conocer un píxel y la clase a la que pertenece.

En este caso las muestras nos las proporcionará el algoritmo de entrenamiento C-Medias Fuzzy, que normalmente nos dará la información necesaria, suficiente y fiable a partir de una imagen de entrenamiento, dependiendo de los parámetros escogidos para dicho algoritmo.

En el caso de que se quisiera clasificar varias imágenes también es posible obtener más muestras una vez hayamos finalizado la clasificación de una imagen, añadiendo a la base de muestras la salida obtenida y refinando por tanto la calidad de las muestras para utilizarlas en posteriores clasificaciones.

Una vez obtenidas las muestas se procede a calcular para cada clase la media y la covarianza como se muestra a continuación:

$$\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \qquad C = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^{\mathrm{t}}$$

El valor de las medias y covarianzas obtenidas serán posteriormente utilizadas para realizar la clasificación.

CLASIFICACIÓN

Como hemos visto antes, el clasificador Bayesiano utilizará la distancia de Mahalanobis para discriminar la pertenencia de un píxel a una clase, ya que las probabilidades a priori son iguales para todas las clases.

La clasificación se reduce a calcular para cada píxel la distancia que existe entre éste y cada una de las clases y elegir la menor distancia

De manera más precisa

$$\mathbf{x} \in c_i \;\; sii \;\; d_M^2(\mathbf{x},\mathbf{m}_i) < d_M^2(\mathbf{x},\mathbf{m}_j) \;\; \forall i \neq j,\; i,j = 1,2,..,c$$

Donde

- Distancia de Mahalanobis:

$$d_M^2(\mathbf{x}_k,\mathbf{m}_i) = (\mathbf{x}_k - \mathbf{m}_i)^{\mathrm{t}} C_i^{-1}(\mathbf{x}_k - \mathbf{m}_i)$$

- $\mathbf{m_i}$ es la matriz que contiene las medias aritméticas de los valores R, G y B de las muestras de la clase c_i.
- $\mathbf{C_i}$ es la matriz de covarianza de la clase c_i.

Diremos por tanto que un píxel pertenece a la clase cuya distancia a ella sea menor.

El proceso se completa una vez se conoce la pertenencia de cada píxel de la imagen a clasificar y se muestra una nueva imagen con los valores de los píxeles originales con los valores de las clases a las que pertenecen.

3.1.2.3 ANÁLISIS DE PARÁMETROS

Los parámetros necesarios para el algoritmo Bayesiano son los que se muestran a continuación:

1 Número de clases ´*c*´

Se utiliza en el entrenamiento y en la clasificación. Este parámetro indica el número de clases en que se desea clasificar los píxeles de una imagen.

2 Muestras clasificadas

Este parámetro corresponde con las muestras del entrenamiento sobre las que se calculan medias y matrices de covarianzas.

3 Medias ‘mi‘y matrices de Covarianza ‘Ci’

Si se han guardado resultados de previas clasificaciones puede ser de gran utilidad usar las medias y matrices de covarianzas obtenidas.

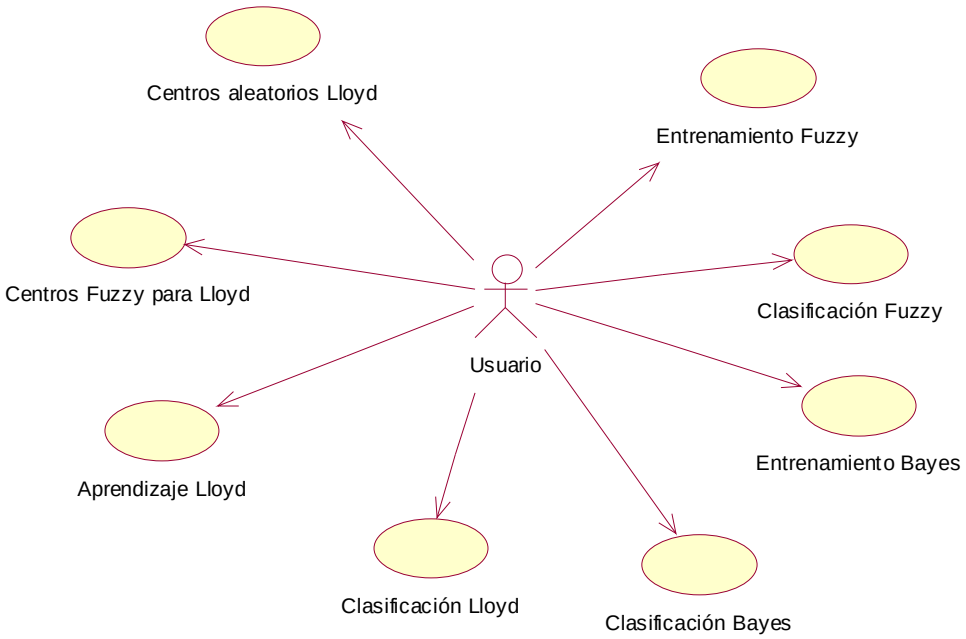
4 Parámetros del algoritmo C-Medias Fuzzy

Puesto que para obtener las muestras iniciales clasificadas para nuestro entrenamiento utilizamos el algoritmo C-Medias Fuzzy, serán necesarios unos valores apropiados para su aplicación, según se explica en dicho algoritmo.

3.2 REQUISITOS FUNCIONALES.

1. Los algoritmos deberán realizar las funciones de entrenamiento y clasificación.
2. Parámetros a elegir por el usuario (como mínimo) serán:
    - Nº de clases.
    - Imágenes con las que se realizará el entrenamiento.
    - Imágenes a clasificar.
3. El entrenamiento se llevará acabo con tantas imágenes como el usuario desee.
4. Los algoritmos serán capaces de clasificar hasta en 8 clases diferentes.

A continuación mostramos en un diagrama los casos de uso:



3.3 REQUISITOS NO FUNCIONALES.

1. La implementación ha de realizarse en el lenguaje de programación orientado a objetos de Java, haciendo uso de la plataforma J2SE™ Development Kit 5.0 y como entorno de desarrollo Eclipse.

2. Los algoritmos deberán ser organizados en base a algún patrón de diseño.

3. La "*Heap"* de la maquina virtual de Java de la máquina donde se ejecute nuestra aplicación ha de tener el espacio suficiente, para que el tamaño de cada imagen no sea un problema. Estimamos que con 5Mb es suficiente.

3.3.1 REQUISITOS FUNCIONALES OPCIONALES

1. El usuario ha de tener un mínimo de conocimientos acerca de los algoritmos implementados para poder introducir parámetros adecuados y obtener resultados satisfactorios.

3.4 RIESGOS.

3.4.1 IDENTIFICACIÓN.

3.4.1.1 TECNOLÓGICOS

Riesgo	Tipo de Riesgo	Descripción
PÉRDIDA DE DATOS	**tecnológico**	Posibilidad de pérdida de información, código o documentación debido a un fallo de la tecnología usada para este fin.
INSUFICIENCIA DE RECURSOS	**tecnológico**	Puede que la aplicación requiera de alguna tecnología de la que no dispongamos.
INCOMPATIBILIDAD DE LAS PARTES	**tecnológico**	El proyecto desarrollado tal vez requiera un determinado software o hardware Que el equipo desconoce
COMPLEJIDAD DE CÓDIGO	**tecnológico**	El código realizado por algún miembro del grupo, puede ser poco legible, poco comentado o incluso muy complicado
TIEMPO DE EJECUCIÓN	**tecnológico**	El tiempo de respuesta del algoritmo debido a una mala algoritmia puede ser excesivo y desagradable a la vista del usuario

3.4.1.2 PERSONALES

Riesgo	Tipo de Riesgo	Descripción
ABANDONO DE ALGÚN COMPONENTE DEL GRUPO	**personales**	Algún componente del grupo por cualquier tipo de razón decida abandonar el proyecto
MAL ENTENDIMIENTO ENTRE LOS COMPONENTES DE AMBOS GRUPOS	**personales**	Tanto dentro del grupo como con el grupo encargado de la interfaz exista un mal ambiente de trabajo.
PERSONAL POCO CUALIFICADO	**personales**	La falta de conocimientos necesarios, tanto en programación como en Ingeniería del Software por parte de algún componente del grupo.

3.4.1.3 ORGANIZACIÓN

Riesgo	Tipo de Riesgo	Descripción
PERDIDA DE DATOS	**Organización**	Posibilidad de pérdida de información, código o documentación debido a una mala gestión de los documentos.
FALTA DE COMUNICACIÓN	**Organización**	Trabajo defectuoso porque los componentes no se comunican entre ellos.

3.4.2 ANÁLISIS.

3.4.2.1 TECNOLÓGICOS

Riesgo	Probabilidad	Efectos y seguimiento
PÉRDIDA DE DATOS	**moderada**	catastróficas
INSUFICIENCIA DE RECURSOS	**moderada**	catastrófica
INCOMPATIBILIDAD DE LAS PARTES	**baja**	catastrófica
COMPLEJIDAD DE CÓDIGO	**moderada**	tolerable
TIEMPO DE EJECUCIÓN	**alta**	grave

3.4.2.2 PERSONALES

Riesgo	Probabilidad	Efectos y seguimiento
ABANDONO DE ALGÚN COMPONENTE DEL GRUPO	**Baja**	seria
MAL ENTENDIMIENTO ENTRE LOS COMPONENTES DE AMBOS GRUPOS	**Baja**	seria
PERSONAL POCO CUALIFICADO	**Baja**	Minúscula

3.4.2.3 ORGANIZACIÓN

Riesgo	Probabilidad	Efectos y seguimiento
PERDIDA DE DATOS	**Moderado**	catastrófica
FALTA DE COMUNICACIÓN	**Baja**	catastrófica

3.4.3 PLANIFICACIÓN.

3.4.3.1 TECNOLÓGICOS

Riesgo	Estrategia a seguir
PÉRDIDA DE DATOS	Llevar copias de seguridad de todos aquellos trabajos realizados, tanto físicamente, como en diferentes dispositivos de almacenamiento (discos diferentes, disquetes, etc.)
INSUFICIENCIA DE RECURSOS	Hacer un reconocimiento de en qué recursos puede fallar e investigar en cómo solventar el posible fallo.
INCOMPATIBILIDAD DE LAS PARTES	Antes de empezar a codificar establecer un entorno que satisfaga las necesidades de todas las partes.
COMPLEJIDAD DE CÓDIGO	Uno o varios miembros del grupo elaborarán información y manuales sobre la nueva tecnología, poniéndose al día previamente sobre ella mediante la compra de libros, explicaciones de alguien documentado, etc.
TIEMPO DE EJECUCIÓN	Investigación de cómo optimizar al máximo el tiempo de ejecución de cada algoritmo.

3.4.3.2 PERSONALES

Riesgo	Estrategia a seguir
ABANDONO DE ALGÚN COMPONENTE DEL GRUPO	Impedir que alguien se haga indispensable haciendo que exista más de una persona especializada en cada parte.
MAL ENTENDIMIENTO ENTRE LOS COMPONENTES DE AMBOS GRUPOS	Entre todos fomentar un buen ambiente de trabajo y compañerismo.
PERSONAL POCO CUALIFICADO	Asegurarse, antes de iniciar el proyecto, de que todos los componentes an cursado las correspondientes asignaturas.

3.4.3.3 ORGANIZACIÓN

Riesgo	Estrategia a seguir
PERDIDA DE DATOS	Llevar copias de seguridad de todos aquellos trabajos realizados, tanto físicamente, como en diferentes dispositivos de almacenamiento (discos diferentes, disquetes, etc.)
FALTA DE COMUNICACIÓN	Establecer un medio de comunicación común en el que periódicamente los componentes del grupo informan de sus avances.

3.4.4 SEGUIMIENTO.

3.4.4.1 TECNOLÓGICOS

Riesgo	Seguimiento
PÉRDIDA DE DATOS	Todo componente tiene una copia almacenada del trabajo que se va realizando.
INSUFICIENCIA DE RECURSOS	Investigación de cómo solventar el riesgo y evitar que el proyecto dependa de ello.
INCOMPATIBILIDAD DE LAS PARTES	Intentar ir integrando las partes de forma gradual, partiendo de que se trabaja en un entorno común
COMPLEJIDAD DE CÓDIGO	Búsqueda de documentación necesaria y estudio de dicha documentación
TIEMPO DE EJECUCIÓN	Aplicación de un sistema de mapeo en la fase de entrenamiento que reduce con notables resultados el tiempo de ejecución

3.4.4.2 PERSONALES

Riesgo	Seguimiento
ABANDONO DE ALGÚN COMPONENTE DEL GRUPO	Manteniendo el contacto periódicamente y poniéndose al día del trabajo de cada compañero.
MAL ENTENDIMIENTO ENTRE LOS COMPONENTES DE AMBOS GRUPOS	En las situaciones comprometidas cediendo todos, y haciendo todo lo posible por crear un buen ambiente.
PERSONAL POCO CUALIFICADO	Todos los componentes del grupo hemos cursado las correspondientes asignaturas para realizar el proyecto.

3.4.4.3 ORGANIZACIÓN

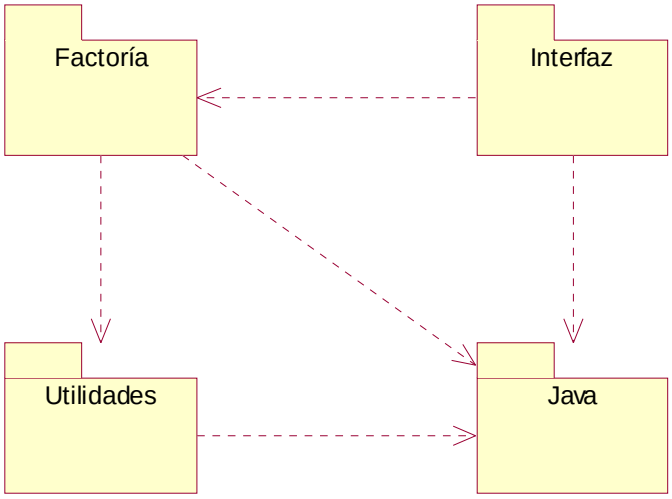
Riesgo	Seguimiento
PERDIDA DE DATOS	Todos los componentes del grupo reciben una copia del trabajo realizado.
FALTA DE COMUNICACIÓN	Quedar periódicamente para poner al día al resto de compañeros. Creación de un grupo de mensajería que se ha convertido en la principal vía de comunicación junto con el profesor de sector.

4 DISEÑO

4.1 ESPECIFICACIÓN DEL DISEÑO.

Para una mayor modularización de la aplicación hemos introducido un diseño basado en paquetes, de manera que podamos diferenciar de manera clara la implementación de los algoritmos y la interfaz del usuario.

El siguiente diagrama muestra esta organización en paquetes:

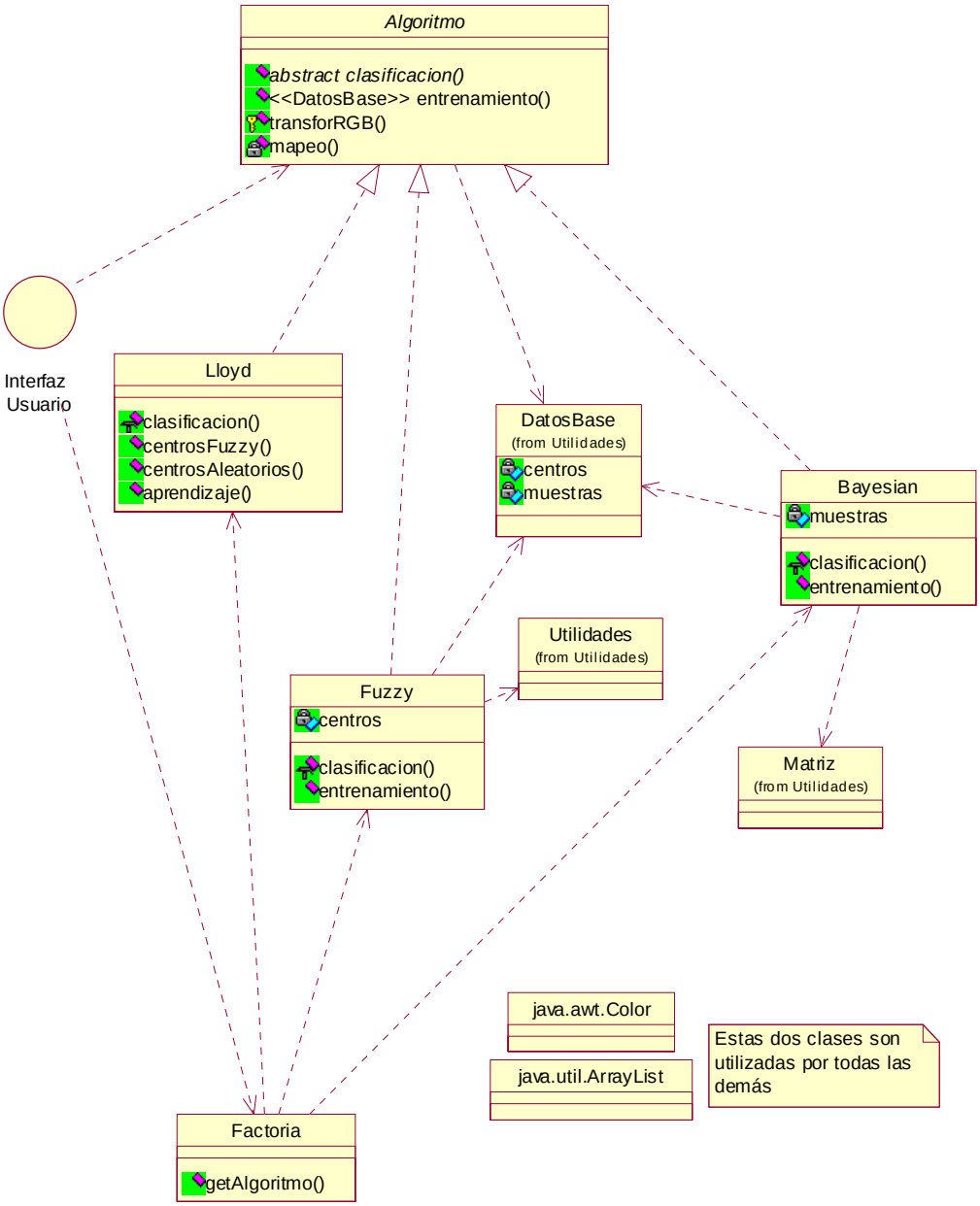


La implementación de los algoritmos se ha llevado a cabo dentro del paquete “Factoría”, llamado de esta manera porque se diseñó siguiendo el patrón Factoría.

El patrón Factoría es una clase que crea objetos de otras clases. La clase Factoria es única. No delega en una subclase la creación de instancias y sus métodos pueden ser estáticos. Esta factoría es muy sencilla: en función del argumento crea una instancia de algoritmo u otro:

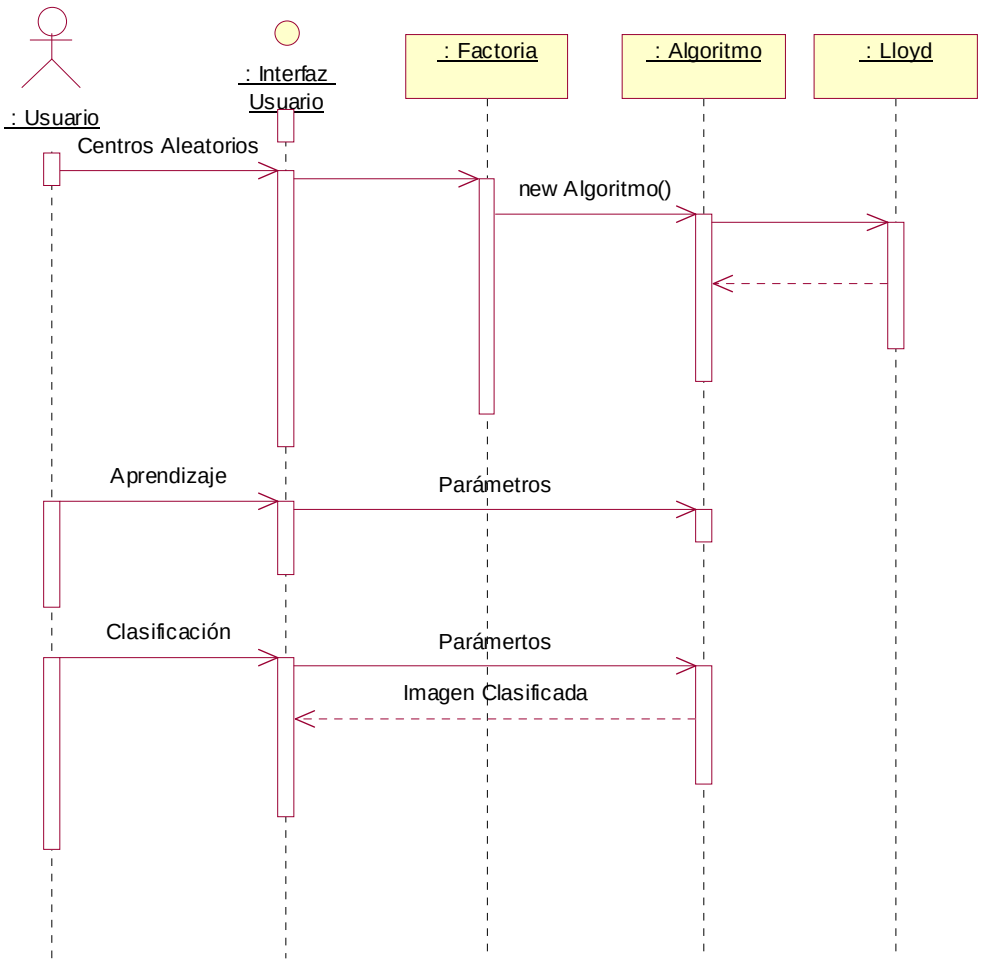
Lo esencial de la clase Factoría es que oculta la complejidad de crear un objeto. Encapsula la creación de la instancia.

La interfaz de usuario será la que se comunique con las clases Factoria y Algoritmo. Y no actuará directamente sobre las clases Lloyd, Fuzzy o Bayesian. Lo explicado anteriormente se puede ver en el diagrama de clases de la aplicación:



A continuación se muestra un diagrama de secuencia que recoge varios casos de uso mencionados anteriormente. Muestra la secuencia completa del proceso de clasificación de una imagen. En este caso, nos hemos centrado en el algoritmo de Lloyd. A través del entrenamiento del algoritmo Fuzzy, se

inicializan los centros de manera aleatoria, aprendiendo con los parámetros seleccionados por el usuario y finalmente procediendo a la clasificación, obteniendo de manera visible a través de la interfaz la imagen clasificada.



5 DESARROLLO

5.1 FASES DE DESARROLLO

5.1.1 FASE DE INICIO

Como se ha mencionado en el plan de fase, esta fase está compuesta por una iteración.

Las fechas planificadas eran:

Fecha de inicio: 10 – 10 - 2006
Fechas de conclusión: 1 – 11 – 2006

En las fechas mencionadas hemos logrado los siguientes objetivos:

- Hemos concretado con el tutor el alcance y objetivos del proyecto.
- Analizamos los riesgos que podrían ocurrir durante el desarrollo del proyecto.
- Analizamos requisitos de acuerdo con los objetivos acordados con el profesor.
- Establecimiento del plan de fase del proyecto. La planificación fijada estuvo coordinada con la fijada por el otro grupo que participa en la aplicación.
- La elección del lenguaje de programación a utilizar para realizar el proyecto (JAVA), se decidió entre los dos grupos que realizamos la aplicación.
- Aprendizaje y familiarización con el lenguaje establecido así como con las bibliotecas específicas que se van a usar en la aplicación (Librería JAI).
- Investigación y documentación de los algoritmos de clasificación a implementar.
- Selección de objetivos que se deben cumplir en la siguiente fase, en función a los resultados obtenidos en esta.

5.1.2 FASE DE ELABORACIÓN

Esta fase está compuesta por una iteración, y tiene las siguientes características:

Fecha de inicio: 2 – 11 - 2006
Fechas de conclusión: 1 – 12 - 2006

Objetivos logrados:

- Diseño global de la aplicación. Este diseño se ha realizado de forma modular y genérica, facilitando así su integración en cualquier tipo de interfaz especializada y en particular respecto de la realizada por el otro grupo.
- Especificación de casos de uso y de los servicios prestados por la aplicación.
- Realización de un prototipo muy simple utilizando otros métodos de clasificación ya interpretados para comprobar el buen funcionamiento del diseño escogido.
- Elaboración de la documentación correspondiente a esta fase.
- Selección de objetivos que se deben cumplir en la siguiente fase, en función de los resultados obtenidos en ésta.

5.1.3 FASE DE CONSTRUCCIÓN

Esta fase es un poco más larga que las anteriores por ello está formada por dos iteraciones.

5.1.3.1. ITERACIÓN 1

Fecha de inicio: 2 – 12 -2006
Fechas de conclusión: 20 – 1 - 2007

Objetivos logrados:

- Reuniones semanales con el tutor y los componentes del otro grupo para explicarles el diseño elegido y las principales características de este.
- Concreción de los puntos de comunicación entre el diseño escogido para su integración en el interfaz. Este apartado fue realizado entre los dos grupos del proyecto, acordando la estructura y el formato de los puntos comunes.
- Revisión de la especificación de requisitos para una posible ampliación.
- Revisión y seguimiento de los riesgos detectados en el proyecto.
- Investigación y búsqueda de soluciones para los riesgos más importantes para la correcta realización del proyecto. En nuestro caso se centra en la optimización de memoria y tiempo de ejecución en los algoritmos de clasificación y en la investigación para aumentar el tamaño de memoria del HEAP que usa el lenguaje de programación elegido para desarrollar los algoritmos (JAVA).
- Elaboración del primer prototipo de la aplicación final.

5.1.3.2. ITERACIÓN 2

Fecha de inicio: 23 – 2 - 2007

Fechas de conclusión: 20 – 3 - 2007

Objetivos logrados:

- Elaboración del prototipo final de la aplicación. En este prototipo se incorporan las soluciones encontradas para los puntos críticos de nuestra aplicación: tiempo de ejecución y memoria.
- Integración del prototipo final al último prototipo elaborado por el grupo encargado de la interfaz.
- Presentación del prototipo final de la aplicación, una vez que se han integrado ambas partes, al tutor.
- Modificación de los errores o deficiencias encontradas por el tutor, al prototipo o versión final de nuestra parte de la aplicación. Este será la versión final de nuestro proyecto.
- Elaboración de la documentación correspondiente a esta fase.
- Selección de objetivos que se deben cumplir en la siguiente fase.

5.1.4 FASE DE TRANSICIÓN.

Esta fase está compuesta por una iteración, y tiene las siguientes características:

Fecha de inicio: 21 – 3 - 2007
Fechas de conclusión: 8 – 6 - 2007

Objetivos logrados:

- El plan de pruebas que habíamos diseñado se llevó a cabo con diferentes cargas, y verificamos que la solución obtenida es correcta y el tiempo de ejecución aceptable.
- Hemos mejorado el tiempo de ejecución gracias al submuestreo de imágenes de entrenamiento grandes, ya que no son necesarias tantas muestras.
- Inicialmente comprobamos los resultados de nuestros algoritmos a través de la consola. Luego los comprobamos en una aplicación que nos proporcionó el otro grupo, teniendo que introducir el algoritmo cada vez que queríamos simular uno distinto. Finalmente integramos los tres algoritmos en una interfaz que permitía ejecutar cualquiera de ellos, seleccionando el deseado por el usuario, mediante el patrón Factoría, comprobado que los resultados obtenidos eran los esperados.
- Búsqueda de mejoras para el comportamiento de la aplicación conjunta.
- Elaboración de la documentación correspondiente a esta fase y de la elaboración conjunta a presentar por los dos proyectos que forman la aplicación.
- Revisión de documentos realizados hasta la fecha.
- Diseñamos un esquema de la memoria principal del proyecto y fue aprobada por el tutor, sugiriendo ciertas modificaciones.
- Realizamos la memoria final del proyecto acorde con las recomendaciones sugeridas por el tutor y utilizando los documentos anteriormente elaborados.
- Hemos elaborado un CD para la entrega definitiva con la versión final de nuestro proyecto.

5.2 PLAN DE PRUEBAS.

En este apartado trataremos el plan de pruebas que se ha llevado a cabo para verificar el comportamiento de la aplicación, centrándonos sobre todo en la parte que corresponde al proyecto.

El plan de pruebas se divide básicamente en tres Fases:

- Pruebas independientes de los proyectos
- Pruebas de integración
- Pruebas finales de la aplicación al completo.

Previamente a cada fase se establecieron unos requisitos mínimos a cumplir por parte de las dos aplicaciones que conforman el proyecto global.

5.2.1 PRUEBAS UNITARIAS.

La primera fase de pruebas corresponde a las realizadas por cada componente del grupo, de forma independiente, sobre sus respectivos algoritmos. Los factores que se evaluaron en dichas pruebas fueron:

1. Entrenamiento del algoritmo en cuestión con una única imagen.
2. Clasificación correcta de una secuencia de píxeles para diferentes números de clases.

5.2.1.1 PRUEBAS REALIZADAS

Cada algoritmo se sometió a una batería de problemas propuestos por el tutor del proyecto, de forma que permitió comprobar con seguridad la bondad de los resultados obtenidos por cada algoritmo.

5.2.1.2. EVALUACIÓN DE PRUEBAS PROYECTO.

Al finalizar la primera fase de pruebas se realizó una segunda evaluación de la cual se destacan los siguientes puntos:

1. Los requisitos principales impuestos al plan se han conseguido satisfactoriamente.
2. Requisitos a tener en cuenta en pruebas futuras:
   2.1. El tiempo de ejecución de los algoritmos ha de disminuir lo máximo posible.
   2.2. El tamaño de la imagen no debe estar condicionado por los recursos de la aplicación.
3. Los algoritmos han de seguir algún tipo de patrón de diseño.

5.2.2. PRUEBAS DE INTEGRACIÓN.

La segunda fase de pruebas se lleva acabo en conjunto entre el grupo encargado de la aplicación y el grupo encargado de la algoritmia. En definitiva la finalidad de esta fase es que exista una perfecta comunicación entre ambas partes una vez se haya llevado a cabo la integración.

Antes de llevar a cabo la integración de ambas partes, el grupo encargado de la algoritmia aplicó el patrón de diseño “Factoría”, encapsulando así la forma en que se crean los objetos correspondientes a cada algoritmo.

Por consiguiente los requisitos que se establecieron en esta fase fueron los siguientes:

1. Los algoritmos han de responder adecuadamente al nuevo patrón de creación.

2. La interfaz ha de ser capaz de pasarle a cada algoritmo, de forma correcta, los parámetros necesarios para su ejecución.
3. Los algoritmos han de devolver los píxeles de la imagen seleccionada por el usuario, clasificados y de forma correcta para que ésta sea capaz de mostrar la nueva imagen.

5.2.2.1 PRUEBAS REALIZADAS

La aplicación fue probada con diversas imágenes tanto en la fase de clasificación como en la fase de entrenamiento. Las imágenes usadas se probaban de menor a mayor tamaño con el fin de evaluar el comportamiento de los algoritmos en cuanto al tiempo de ejecución y en cuanto a la capacidad de memoria a la hora de procesar dichas imágenes.

5.2.2.2 EVALUACIÓN DE PRUEBAS DE INTEGRACIÓN.

En la evaluación que se realizó al finalizar esta fase de pruebas se obtuvieron como puntos o conclusiones a destacar:

1. Los requisitos principales impuestos a esta fase se han conseguido de forma satisfactoria.
   1.1. Ambas partes se comunican correctamente.
   1.2. Los algoritmos se han ajustado correctamente al plan de creación “Factoría”.

2. La eficiencia en cuanto a la velocidad de procesamiento no es la deseada a partir de un determinado tamaño, estimado este en 500x500 píxeles (dependiendo también del número de clases).

3. La memoria de la Maquina Virtual de Java limita el tamaño de la imagen a un valor concreto, estimado en 500x500 píxeles (dependiendo igualmente del número de clases).

4. Requisitos a tener en cuenta en pruebas futuras:
   1.3. El usuario ha de ser capaz de seleccionar varias muestras para realizar el entrenamiento.

5.2.3 PRUEBAS FINALES

En esta última fase se llevaron a cabo acabo las pruebas finales, con el objetivo de cumplir los requisitos pendientes del resto de planes aplicados al proyecto.

Para ello se establecieron los siguientes objetivos:

1. La ampliación llevada a cabo en la Máquina Virtual de Java permite clasificar imágenes sin restricción por el tamaño.
2. El método de mapeo aplicado al entrenamiento no perjudica al buen funcionamiento de los algoritmos y reduce el tiempo de ejecución.
3. El usuario puede seleccionar varia muestras para realizar un entrenamiento.

5.2.3.1 PRUEBAS REALIZADAS

La aplicación fue probada con diversas imágenes tanto en la fase de clasificación como en la fase de entrenamiento. Las imágenes usadas se probaban de menor a mayor tamaño con el fin

de evaluar el comportamiento de los algoritmos en cuanto al tiempo de ejecución y en cuanto a la capacidad de memoria a la hora de procesar dichas imágenes.

5.2.3.2. EVALUCION DE LAS PRUEBAS FINALES

En la evaluación realizada al finalizar esta fase del plan pruebas se obtiene como conclusión que la aplicación reacciona con el comportamiento deseado, cumpliendo en su totalidad los requisitos que se le han ido imponiendo a lo largo de su implementación.

6 RESULTADOS PROYECTO

6.1 EJEMPLOS

A continuación mostramos un ejemplo del resultado obtenido al clasificar una imagen con todos los algoritmos. Los valores de los parámetros para cada caso los hemos fijados por defecto y como número de clases a identificar en la imagen 8.

Para realizar el entrenamiento hemos seleccionado una serie de 4 imágenes que han sido previamente almacenadas y procesadas por cada algoritmo.

IMAGEN ORIGINAL

6.1.2 ALGORITMO C-MEDIAS FUZZY

El algoritmo c-media Fuzzy ha realizado la el entrenamiento y clasificación con los siguientes parámetros:

1 C o número de “Clusters”=8;

2 M o peso exponencial = 2;

3 ε o error permitido = 0,02;

4 NUMMAX número de iteraciones = 10;

IMAGEN RESULTADO FUZZY

6.1.3 ALGORITMO BAYESIAN

El algoritmo Bayesian ha realizado el entrenamiento y clasificación con los siguientes parámetros:

1. C o número de “Clusters”=8;
2. ε o error permitido = 0,02;
3. NUMMAX número de iteraciones = 10;

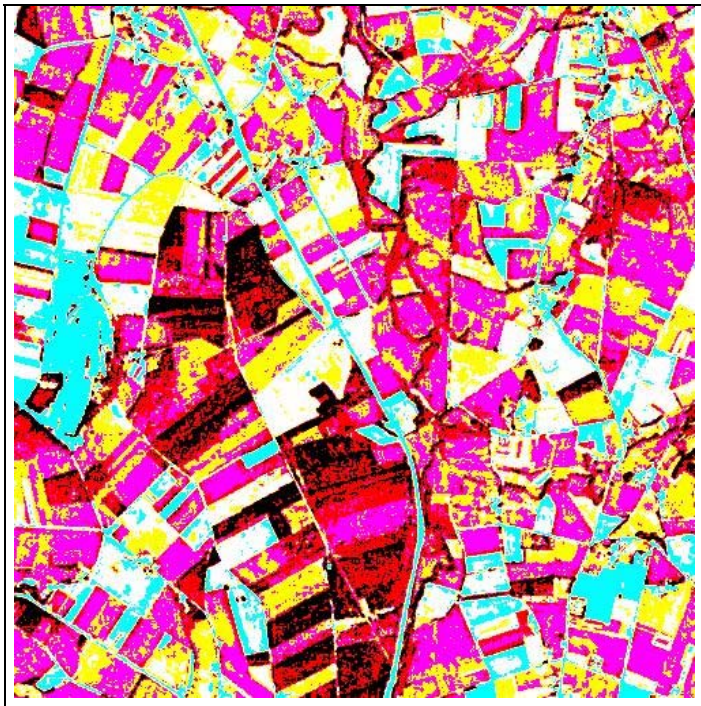


IMAGEN RESULTADO BAYESIAN

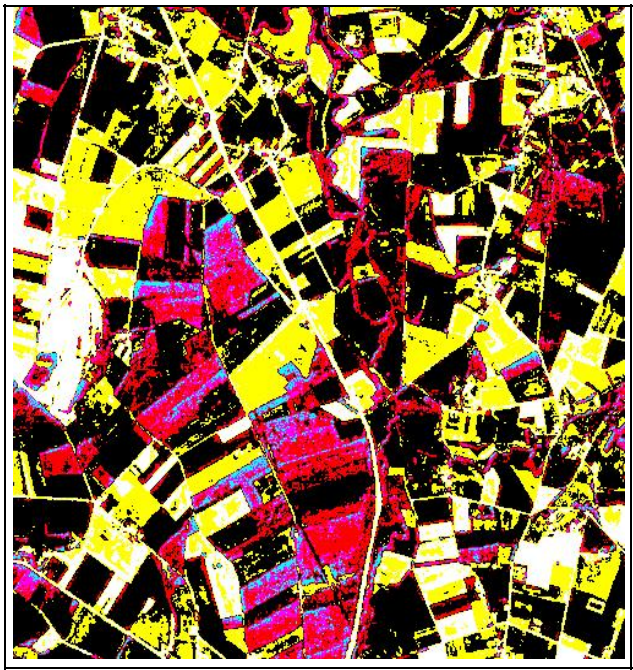
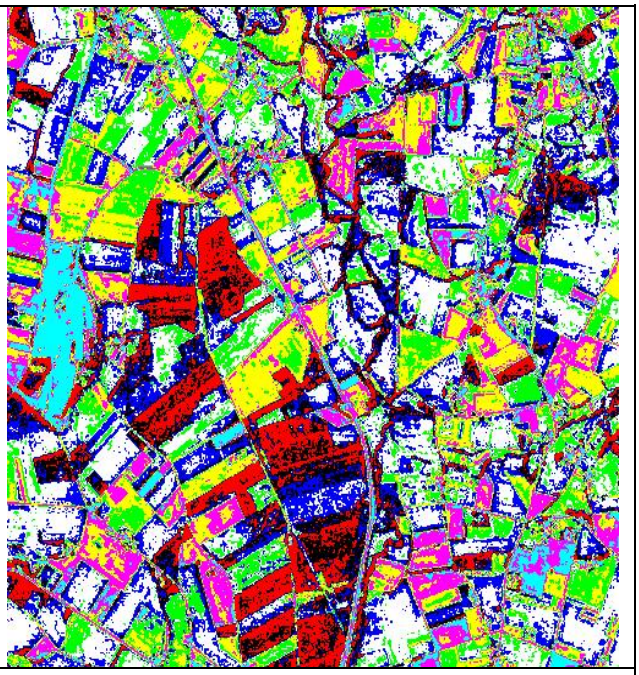
6.1.4 ALGORITMO LLOYD

El algoritmo Lloyd nos permite realizar varios tipos de operaciones. Mostramos los resultados en el siguiente orden

1. Imagen clasificada con centros aleatorios.
2. Imagen clasificado con centros Fuzzy.

El algoritmo de Lloyd ha realizado la clasificación con los siguientes parámetros:

1. C o número de “Clusters”=8;
2. γ o Razón de aprendizaje = 0.1;
3. ε o Tolerancia = 1e-10;
4. NUMITER número de iteraciones = 10;

	
CENTROS ALEATORIOS	CENTROS FUZZY

Se puede observar que cada algoritmo ha clasificado satisfactoriamente la imagen propuesta como ejemplo. Como observaciones a destacar:

- El algoritmo Fuzzy es el que mejores resultados ha obtenido, tanto en su aplicación como algoritmo en sí, como en su utilización como herramienta para calcular los centros iniciales del algoritmo Lloyd.

- La utilización de centros Fuzzy para la aplicación del algoritmo Lloyd ha mejorado notablemente la clasificación con respecto a la obtenida haciendo uso de centros aleatorios.

- El algoritmo Bayesian ha obtenido una buena clasificación pero no tan precisa con la obtenida por el algoritmo Fuzzy y el algoritmo Lloyd con centros Fuzzy.

6.2 APLICACIÓN OBTENIDA.

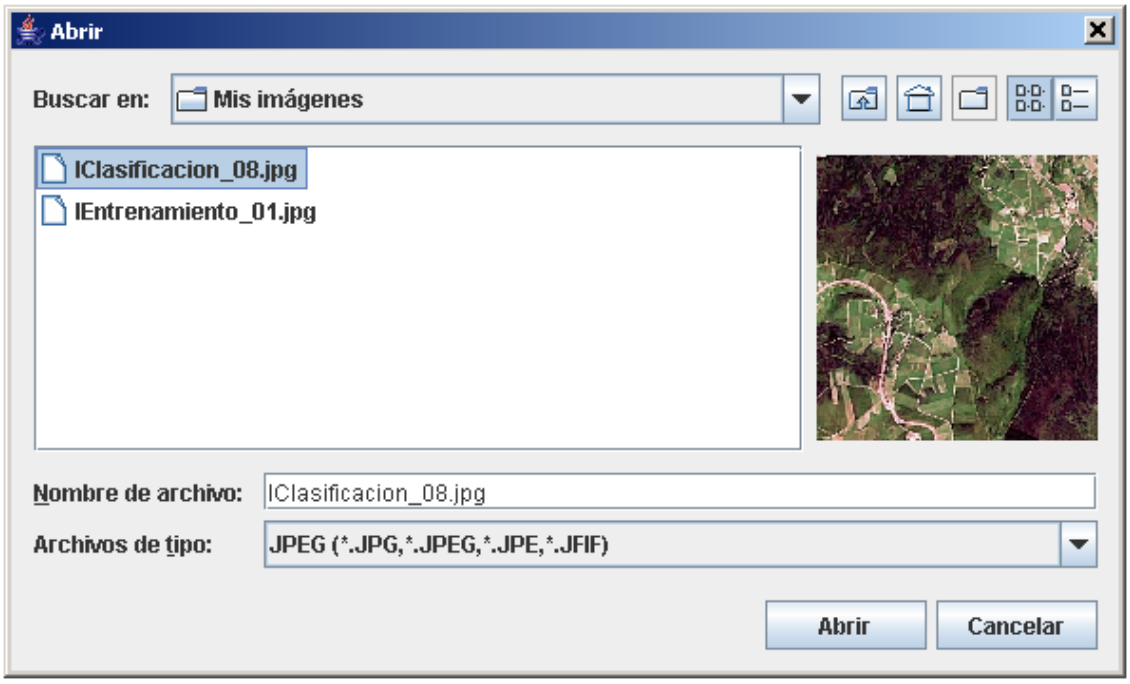
En este apartado se muestra un ejemplo de la aplicación tras integrar los algoritmos implementados por nuestro grupo, con la interfaz realizada por el otro grupo participante de la aplicación.

A continuación mostraremos los pasos necesarios para realizar una clasificación Fuzzy de una imagen determinada utilizando la aplicación resultante tras la integración. La clasificación utilizando los algoritmos de Lloyd y Bayesiano, serían de forma similar cambiando los parámetros requeridos específicos de cada algoritmo. En el caso de Lloyd la aplicación ofrece una opción adicional pudiendo cargar los centros aleatoriamente.

EJEMPLO DE CLASIFICACIÓN FUZZY

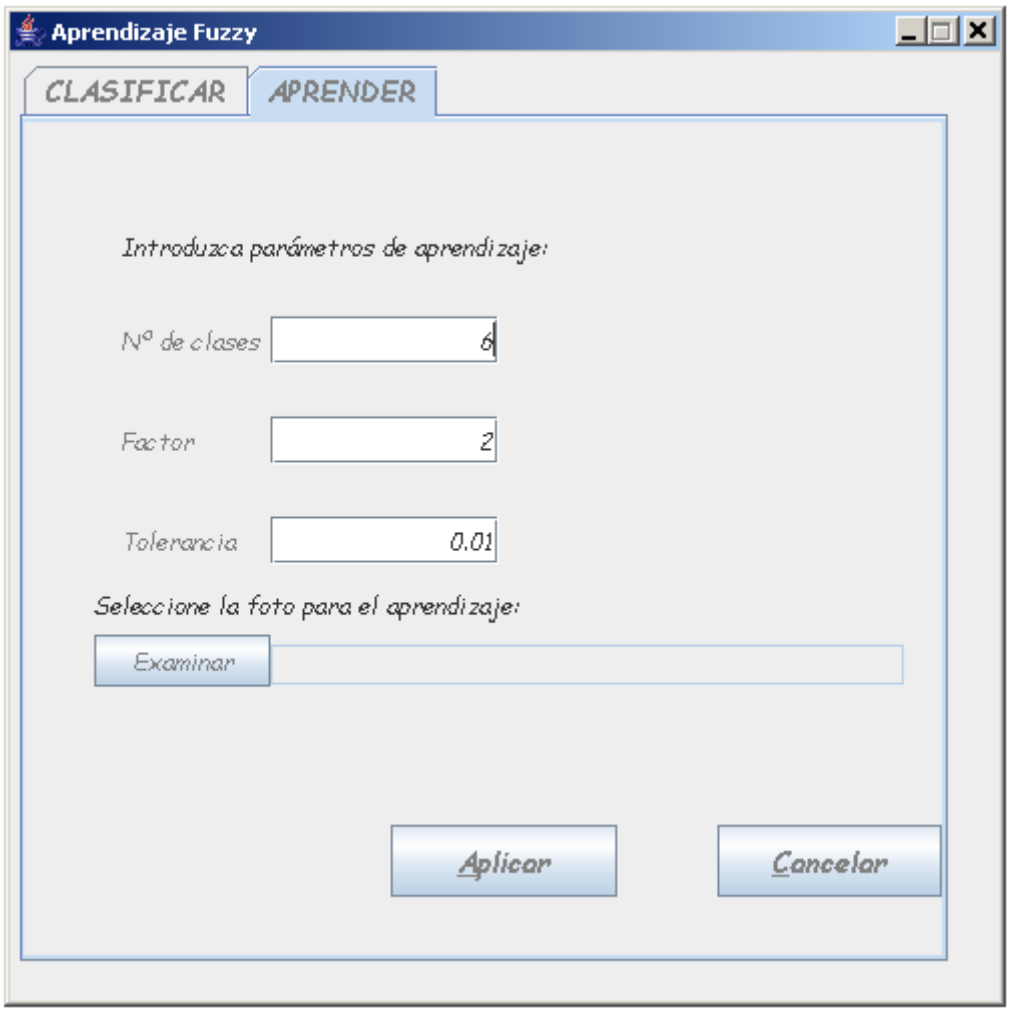
1. Inicio de la aplicación.

2. Elección de la imagen a clasificar.

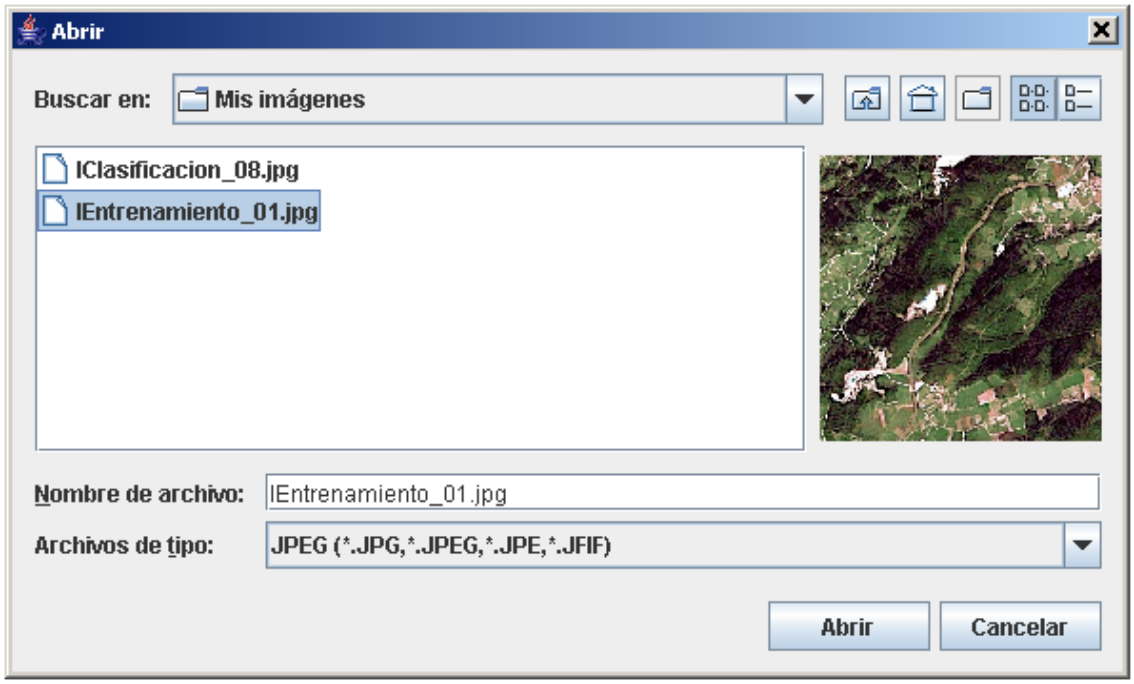


3. Selección del algoritmo de clasificación, en nuestro ejemplo tipo Fuzzy.

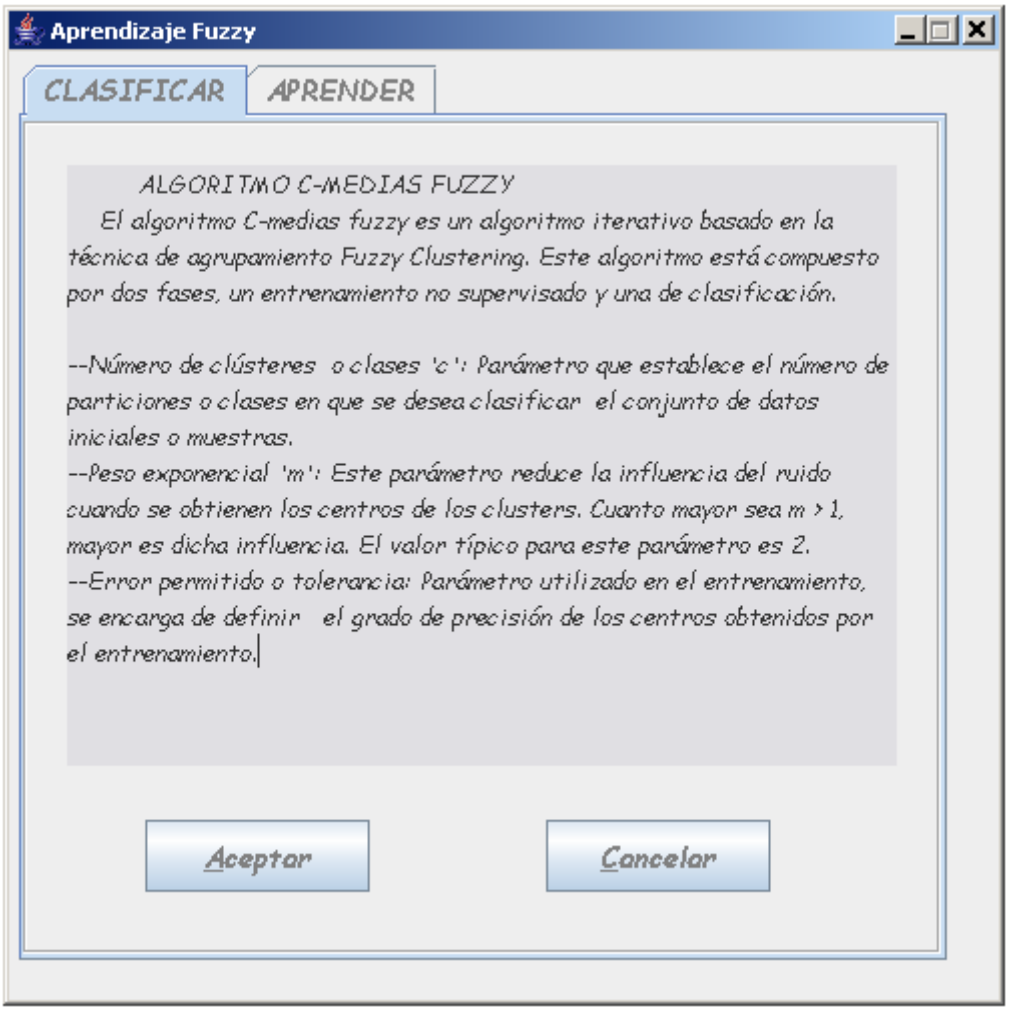
4. Selección de la pestaña de aprender, en la cual se efectuara el entrenamiento Fuzzy e introducción de los parámetros específicos del entrenamiento.

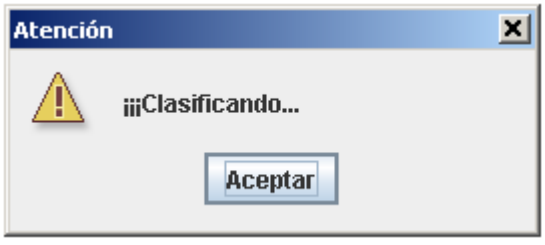


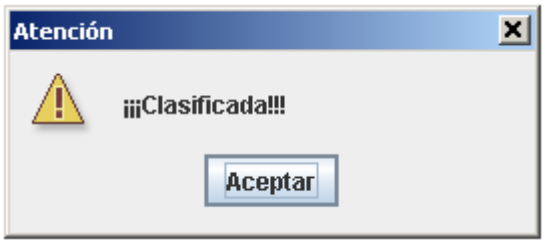
5. Selección de la imagen para realizar el entrenamiento.



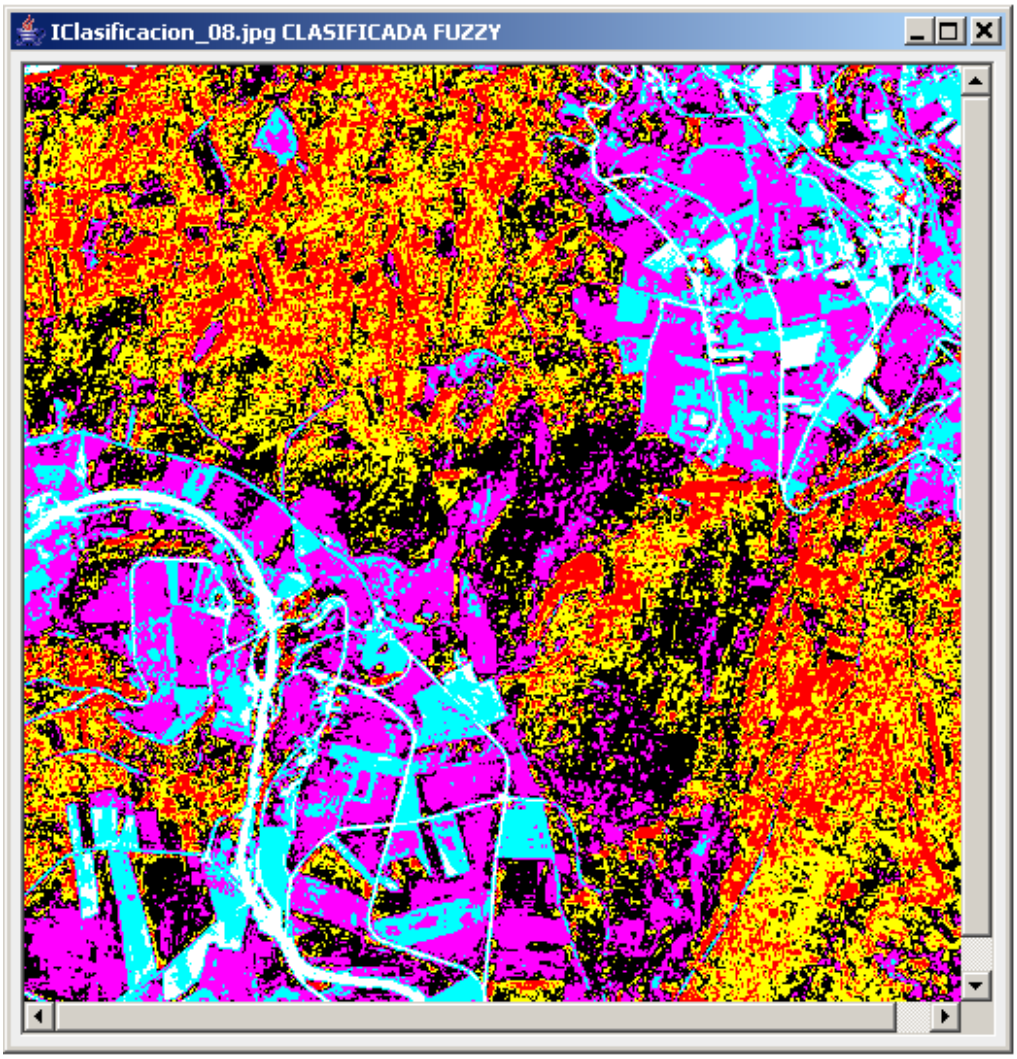
6. Una vez realizado el entrenamiento, selección de la pestaña de clasificación, en la cual hay una breve explicación del algoritmo seleccionado.

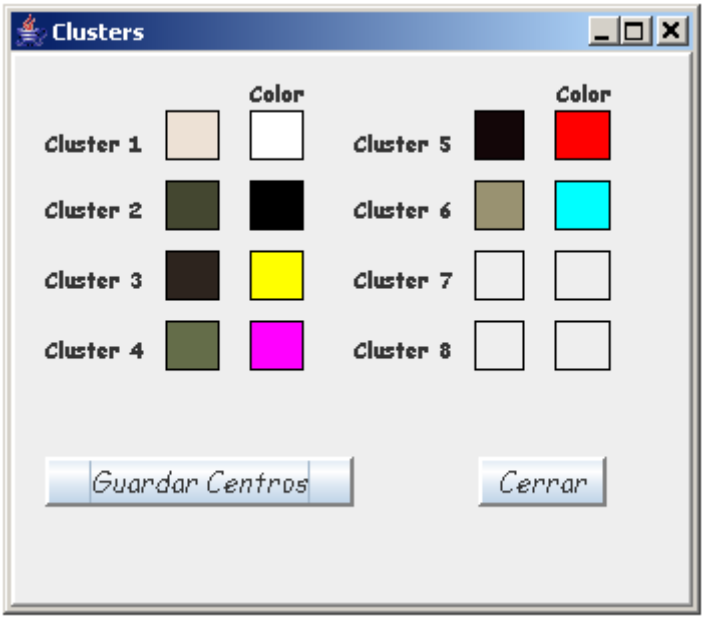






7. Exposición por pantalla de la imagen obtenida una vez realizada la clasificación.





7 CONCLUSIONES Y TRABAJO FUTURO.

El proyecto se presenta bajo la cobertura de la asignatura de Sistemas Informáticos. Ha sido elaborado bajo la metodología de ingeniería del Software tratando de emular las fases de cualquier proyecto profesional.

Como conclusiones más relevantes cabe destacar las siguientes:

1. Se trata de un proyecto demandado por la industria del sector dedicado al tratamiento de imágenes para cubrir en el futuro las deficiencias de los métodos actualmente implementados en herramientas comerciales. Algunas de estas empresas y herramientas son las mencionadas en la sección 1.3.

2. El planteamiento anterior ha permitido desarrollar el proyecto bajo las diferentes fases del ciclo de vida de un proyecto SW: definición de requisitos, arquitectura del sistema, diseño, implementación, integración y pruebas (unitarias y de integración).

3. El trabajo en equipo, primeramente como equipo aislado y posteriormente como trabajo integrado por dos equipos. Se trata este de un aspecto muy importante de cara a nuestra futura dedicación profesional poniendo de manifiesto la capacidad de trabajo en equipo, donde han imperado aspectos tales como: coordinación, comunicación, planificación o corrección de errores.

4. Se han implementado tres métodos de clasificación de texturas que suponen la aplicación de técnicas de inteligencia artificial en tratamiento de imágenes, un área emergente como aplicación de nuevas tecnologías informáticas.

5. Se han solventado las dificultades surgidas así como la propuesta de nuevas soluciones a una problemática derivada del tamaño de las imágenes, lo que nos sitúa ante una perspectiva con capacidad de resolver problemas de cara al futuro profesional.

6. El diseño modular nos permite un fácil mantenimiento.

La elaboración y el diseño tanto de la aplicación en su conjunto, como de nuestro proyecto individualmente han sido realizados pensando en el futuro.

La realización de una aplicación con una perspectiva de aplicación a cliente, requiere un diseño claro y estructurado ya que el manteniendo y modificación de la aplicación puede ser efectuado en un futuro más o menos próximo por personas ajenas a los desarrolladores de la aplicación. Además en muchos casos es necesario la modificación de la aplicación para adaptarse a nuevos requisitos y la ampliación para definir nuevas funcionalidades no incluidas en la aplicación inicial.

Al tratarse de una aplicación derivada de las necesidades de potenciales clientes reales se ha diseñado la misma con una proyección de futuro con las siguientes características y propiedades:

1. El diseño modular y estructurado (patrón Factoría) permite la incorporación de nuevos módulos y la eliminación de los ya existentes, pudiendo realizarse por personas ajenas al proyecto en el momento actual tras su incorporación en el futuro como continuidad del mismo.

2. Ampliación del paquete incluyendo un nuevo método de clasificación o entrenamiento que mejore los resultados obtenidos por los existentes. Esta inclusión sería trivial, debiendo añadir

únicamente un método con la implementación al paquete diseñado.

3. Eliminación de un algoritmo de clasificación. Esta modificación no tiene ninguna complicación para el paquete principal, dado el diseño modular de este.

4. Integración del paquete elaborado en una interfaz distinta a la utilizada en la aplicación. El único requisito imprescindible para realizar la integración es que la interfaz este desarrollada en Java o en un lenguaje de programación que permita hacer llamar a métodos implementados en Java. Cumplido este requisito, la integración es sencilla y rápida.

5. Modificación de los algoritmos existentes, añadiendo parámetros que indiquen mayor información sobre el algoritmo, como puede ser el cálculo del tiempo de ejecución del algoritmo o alguno similar.

6. Ampliación o modificación de las propiedades de la clasificación. Los algoritmos utilizados clasifican las imágenes teniendo en cuenta las propiedades R,G,B de cada píxel. Si se desea realizar una clasificación mediante con estos algoritmos, teniendo en cuenta otras propiedades o para otros objetos, únicamente se deberá introducir una clase que contenga estas prioridades y hacer una modificación mínima en los métodos de clasificación y entrenamiento. Esto es debido a que los métodos implementados de manera genérica, para poder ser utilizados para otros tipos de clasificaciones.

Las modificaciones y ampliaciones para los que han sido diseñado el proyecto han sido enunciadas anteriormente, no obstante un cliente

puede realizar alguna más especifica dependiendo de su objetivo propuesto.

AGRADECIMIENTOS.

Gracias al director del proyecto y profesor de la Facultad de Informática de la Universidad Complutense de Madrid, Don Gonzalo Pajares Martinsanz, por aceptarnos en su proyecto para la asignatura de Sistemas Informáticos y por la atención y ayuda ofrecida para la realización del proyecto.

Agradecimiento a Javier Vizcaíno Lamas, Álvaro Toledo y a Héctor Marco, integrantes del proyecto encargados de desarrollar la interfaz de la aplicación, por su ayuda en la realización del proyecto y su constante colaboración y comunicación para la realización de las partes conjuntas de la aplicación.

Un especial agradecimiento a la empresa Dimap (Digital Image Processing) que nos ha facilitado las imágenes aéreas utilizadas en el proyecto, a su vez proporcionadas por el Servicio Territorial de Galicia (SITGA).

8 BIBLIOGRAFÍA.

[1] G. Pajares y J.M. de la Cruz, Visión por Computador: Imágenes digitales y Aplicaciones, RA-MA, 2001.

[2] Revistas internacionales: " Fuzzy Sets and Systems" y " IEEE Trans. On Neural Networks.

[3] Artículo " Clasificación de texturas naturales mediante k-means". Gonzalo Pajares y J.M de la Cruz. Revista REVC.

[4] Zimmerman H.J 1991. Fuzzy Set Theory And its Applications, Kluwer Academic Publishers, Norwell.