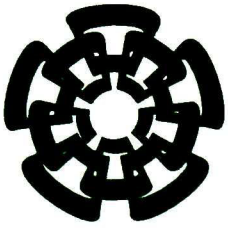
CT-747
Don-2013



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Uso de los Procesos de Dirichlet y Optimización por Colonia de Hormigas para el Aprendizaje de Sistemas Difusos

Tesis que presenta:
Arturo García García

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Andrés Méndez Vázquez

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Enero de 2013.

210421.1

Uso de los Procesos de Dirichlet y Optimización por Colonia de Hormigas para el Aprendizaje de Sistemas Difusos

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Arturo García García
Ingeniero en Sistemas Computacionales
Universidad de Guanajuato 2005-2010

Becario de CONACYT, expediente no. 243193

Director de Tesis
Dr. Andrés Méndez Vázquez

CINVESTAV del IPN Unidad Guadalajara, Enero de 2013.

Agradecimientos

Gracias a Dios por permitirme llegar hasta esta etapa de mi vida y por haberme permitido vivir buenas y malas experiencias a lo largo de ésta, las cuales han formado y definido al ser humano que ahora soy y por haberme permitido conocer a tanta gente en mi camino.

Gracias a mis padres, Ma. Guadalupe García Ramírez y Arturo García Rodríguez, que creyeron en mí y que sin su apoyo y consejos no hubiera llegado a estas instancias de mi vida. A mi hermano, José María, por su apoyo y paciencia.

Este trabajo de tesis, no pudo ser realizado sin el apoyo y guía de mi asesor de tesis Dr. Andrés Méndez Vázquez y que me compartió sus experiencias y conocimientos para la elaboración del trabajo, así como su contribución para mi formación personal.

Gracias a los compañeros y amigos que forman parte del laboratorio de Computación por compartir su tiempo y conocimiento para el mejoramiento, no sólo de este trabajo, sino de mi persona.

A CONACYT por proporcionarme el apoyo financiero para poder realizar los estudios de maestría.

Resumen

El aprendizaje máquina es muy importante y es una área de estudio muy extensa. La importancia de tener sistemas de aprendizaje deriva de la necesidad de que las computadoras sean capaces de aprender diversos tipos de tareas para que las realicen de forma autónoma o con poca intervención del ser humano, por razones como lo peligroso, lo repetitivo o lo complicado de la tarea entre otras.

Dentro de las Ciencias Computacionales existe una gran variedad de áreas donde el aprendizaje máquina puede ayudar, algunas de ellas son: Visión por Computadora, Minería de Datos, Aprendizaje Bayesiano, entre o tras. De estas áreas se desprenden las técnicas utilizadas en este trabajo para el aprendizaje de Sistemas Difusos.

En este trabajo se utilizan técnicas de *clustering* basados en procesos de Dirichlet (método probabilístico), algoritmos de optimización (Métodos de Colonias de Hormigas para optimización) y los Sistemas Difusos (*Soft Computing* para el manejo de incertidumbre) para diseñar un Sistema de Aprendizaje de Sistemas Difusos. Esto nos permitirá a la computadora aprender a realizar una tarea dada de forma autónoma o semi-autónoma.

Abstract

Machine Learning is a very important an extensive area of study. The importance of learning systems arises from the necessity of computers to be able to learn different types of tasks in order to perform them autonomously or with a few human interventions. The main reasons are that the tasks can be dangerous, repetitive or complicated, among others.

In Computer Science, there are several areas where machine learning can help, such as: Computer Vision, Data Mining, Bayesian Learning, among others. From these areas, several techniques are used in this work for Fuzzy Systems Learning.

In this work, we use clustering techniques based on Dirichlet processes (probabilistic approaches from computer vision), optimization algorithms (Ant Colony methods for optimization) and Fuzzy Systems (Soft computing method to handle uncertainty). All the above is used to design a learning system for Fuzzy Systems. This will allow the computer to learn to perform a given task in an autonomous or semi-autonomous way.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los Sistemas Difusos se usan para la solución de una gran variedad de problemas como lo son: control, toma de decisiones y en segmentación de imágenes, entre otros. Por lo que es importante el estudio de las características que debe tener el Sistema Difuso para la correcta solución de estos problemas.

En muchas, sino es que en todas de las ocasiones es necesario un análisis muy detallado del problema que se desea resolver con un Sistema Difuso y la asesoría de un experto en el problema para lograr el diseño de un sistema que lo solucione, lo cual es tardado y costoso.

El Aprendizaje No Supervisado a sido un tema muy estudiado para diversos problemas, en éste se propone el uso de herramientas estadísticas en conjunto, basadas en la naturaleza como lo son los Algoritmos Genéticos (AG), las Estrategias Evolutivas (EE), los Algoritmos de Optimización, entre otras, para diseñar sistemas que sean capaces de aprender ciertas tareas "sin la necesidad de la intervención humana". Con esto en mente propondremos una solución basada en el uso de algunas de estas técnicas en este trabajo.

1.1. Definición del Problema: Aprendizaje de sistemas Difusos.

En este trabajo se aborda el problema del aprendizaje de sistemas difusos, los cuales pueden ser utilizados en la solución de diversos problemas como en control y en segmentación de imágenes, entre otros.

El problema se puede dividir en dos grandes sub-problemas:

1. Obtención de los Conjuntos Difusos de cada una de las entradas.

2. Obtención y minimización de la Base de Reglas Difusas.

El primer de ellos se puede ver como un problema de *Clustering*, donde se desea obtener las clases o *clusters* que nos separen "perfectamente" a los datos sin que estas clases se traslapen unas con otras (*overlapping*) o en su defecto que lo minimice.

Para dar solución a este primer problema se propone el uso de los Procesos de Dirichlet (DP, por sus siglas en inglés) los cuales son usados para el *clustering* no supervisado de los datos.

El segundo de ellos puede ser visto como un problema de optimización donde se desea obtener un menor número de elementos teniendo una buena aproximación al resultado deseado. Este problema en particular se puede ver como un problema de asignación, en el cual tenemos un cierto número de antecedentes y uno de consecuentes (todos diferentes entre sí) y se desea formar un mínimo conjunto de reglas (formados por un antecedente y un consecuente) el cual nos arroje una buena aproximación a la salida deseada.

Para la solución de este problema se propone el uso de la técnica de Optimización por Colonia de Hormigas (ACO, por sus siglas en inglés) que es una técnica de *Machine Learning*. Esta técnica nos ayudará a reducir el número de reglas difusas del sistema, este conjunto debe cumplir con la minimización del error entre la salida arrojada por este sistema y la salida esperada dada por los datos de entrenamiento.

En resumen, se usará la combinación de los DP y del ACO para resolver el problema del Aprendizaje de Sistemas Difusos.

1.2. Motivación.

El poder contar con sistemas capaces de aprender tareas sin la intervención humana o una mínima es muy importante, dado que ésto nos lleva al ahorro de recursos económicos, materiales y humanos. En muchas de las ocasiones al carecer de recursos económicos y por ende humanos, se lleva a la implementación de sistemas que no cumplen con las características básicas necesarias lo cuál deriva en problemas que pueden ser más costosos.

El poco desarrollo en sistemas de aprendizaje de sistemas difusos aunado al gran desarrollo de los sistemas difusos en varios problemas, nos lleva a considerar esta como una área de oportunidad y desarrollo en la cual se pueden hacer grandes aportaciones.

El desconocimiento del comportamiento de los datos y la falta de expertos en el área nos da pie a la utilización de los procesos de Dirichlet en el desarrollo del trabajo. A su vez la gran cantidad de datos nos lleva al uso de algoritmos de optimización como parte del sistema.

El desarrollo de los procesadores con capacidad de cómputo en paralelo ha crecido en los últimos años y de la mano la programación en paralelo. El alto grado de paralelización de muchos de los algoritmos de optimización, en especial del algoritmo ACO utilizado en este trabajo, aunado a la gran cantidad de datos a procesar, nos da pie a la implementación de estos algoritmos haciendo uso de las tarjetas NVIDIA, con lo cuál se busca la reducción del tiempo de cómputo.

1.3. Propuesta.

Proponemos un sistema de aprendizaje de sistemas difuso el cual hace uso de los procesos de Dirichlet y el algoritmo ACO. La combinación de estas dos técnicas nos dará un sistema capaz de determinar los componentes básicos de un sistema difuso para la tarea en cuestión que se quiera resolver.

Para validar y observar los resultados del sistema propuesto se aplicará a la segmentación de imágenes por color.

1.3.1. Restricciones.

Que sí va a resolver del problema:

Obtención del número óptimo de reglas difusas y evitar la explosión de las mismas.

Evitar el *overlapping* de conjuntos difusos.

Aprendizaje No supervisado.

Que no va a resolver del problema:

El problema de pre-procesar la imagen.

Buscar el mejoramiento de la segmentación de imágenes por color.

Robusto a cambios de iluminación.

La optimización de las reglas difusas.

1.3.2. Evaluación.

Los resultados se compararán con dos sencillas técnicas utilizadas en la segmentación de imágenes: el algoritmo K-Means y el algoritmo Fuzzy C-Means. Los cuales serán inicializados haciendo uso del número de conjuntos difusos obtenidos por el DP y los resultados se compararán usando una matriz de confusión.

Del mismo modo los resultados arrojados por el sistema difuso aprendido serán comparados con una matriz de confusión con los resultados arrojados por los DP y así se poderá observar el porcentaje de aprendizaje obtenido por el ACO.

La matriz de confusión hará una relación del número de píxeles de una clase de color encontrada por los DP vs el número de píxeles de esa misma clase catalógados por el sistema difuso que se aprendió. A parte de la matriz de confusión se utilizarán algunas de las principales métricas estadísticas para ver la similaridad entre el cluster base y el cluster dado por el algoritmo como lo son, la métrica F [22] [26], el índice Rand [25], el índice de Jaccard [15], el índice Fowlkes–Mallows [9], entre otros. Con el fin de dar un valor cuantitativo de los resultados arrojados por el sistema.

1.4. Objetivo.

El objetivo de este trabajo es el desarrollo e implementación de un Sistema de Aprendizaje de Sistemas Difusos, el cuál requiera un mínimo de intervención por parte del usuario y que sea capaz de determinar tanto el número de conjuntos difusos como el número de reglas difusas necesarias.

Este sistema será aplicado a la Segmentación de Imágenes por Color para observar su comportamiento.

Un segundo objetivo es el aprovechamiento de la alta paralelización de las técnicas y algoritmos utilizadas para el aprovechamiento del cómputo en paralelo de las tarjetas de vídeo NVIDIA con lo cual se busca disminuir el tiempo de ejecución.

1.5. Estructura de la Tesis.

En el primer capítulo damos una introducción al problema que queremos resolver, las restricciones y evaluaciones que realizaremos a los resultados. En el segundo capítulo abordamos los temas base para poder resolver el problema, así como la descripción de los trabajos realizados anteriormente para la solución del mismo. En el tercer capítulo se tiene la solución propuesta así como el diseño del sistema y su implementación. En el cuarto capítulo se muestran los resultados arrojados por el sistema y los análisis de éstos. En el capítulo final se muestran las conclusiones finales de este trabajo así como el trabajo a futuro.

En las últimas secciones se tiene la bibliografía, glosario y anexos, que sirve como base de este trabajo.

Capítulo 2

Estado del Arte: Aprendizaje de Sistemas Difusos y Procesos de Dirichlet.

En este capítulo se presentan las técnicas basicas para la solución del problema del Aprendizaje de Sistemas Difusos.

Se presenta una recopilación de trabajos relacionados con el Aprendizaje de Sistemas Difusos para la Segmentación de Imágenes por Color. Algunos de estos trabajos hacen uso de técnicas de optimización, entre ellas destacan el ACO y la Optimización por Enjambre de Partículas (PSO, por sus siglas en inglés). En otros se hace uso de las Redes Neuronales (NN, por sus siglas en inglés) para la Segmentación de Imagénes por Color.

Otros trabajos relacionados tratan sobre los Procesos de Dirichlet y sus aplicaciones en la Segmentación de Imagénes y el *clustering* de datos.

2.1. Background

2.1.1. Lógica Difusa

La lógica difusa es una forma de lógica multi-valuada, trata con razonamiento aproximado más que exacto. En contraste con la teoría tradicional de lógica, donde las variables solo pueden tener dos valores de verdad (lógica binaria). Las variables de la lógica difusa tienen un valor de verdad que varia entre 0 y 1.

La lógica difusa se desarrolló para manejar el concepto de verdad parcial, esto es, se

mueve entre lo completamente verdadero y completamente falso. Cuando se usan variables lingüísticas estos valores de verdad pueden ser dados por una función especifica. Al uso de esta función para el manejo de los valores de verdad se le llama conjunto difuso.

La lógica difusa llego con la propuesta de la teoría de conjuntos difusos hecha por Lotfi Zadeh [38] en 1965. La lógica difusa se ha aplicado en muchas áreas que abarcan desde la teoría de control hasta la inteligencia artificial, entre otras.

2.1.2. Sistemas Difusos

Los sistemas difusos fueron introducidos por L.A. Zadeh [38] en 1965. Diseñados para representar matemáticamente la incertidumbre y vaguedad y proporcionar herramientas formalizadas para trabajar con la imprecisión intrínseca en muchos problemas. El uso de este tipo de sistemas en muchas tareas de control y de toma de decisiones se ha vuelto muy popular dadas las características de estos sistemas.

Los sistemas difusos están compuestos por:

$InLV$, número de variables lingüísticas de entrada.

$OutLV$, número de variables lingüísticas de salida.

$InFS$, conjunto de conjuntos difusos de cada una de las variables lingüísticas de entrada.

$OutFS$, conjunto de conjuntos difusos de cada una de las variables lingüísticas de salida.

FR, conjunto de reglas difusas.

Cada uno de las variables lingüísticas y cada uno de los conjuntos difusos están asociados con una etiqueta formada por palabras u oraciones utilizadas cotidianamente.

El comportamiento de los sistemas difusos en general es el siguiente:

1. **Fuzificación:** Dado un vector de entrada, $\vec{x}$ de dimensión d, activa conjuntos difusos si:

$$C_{j,k} \text{ is activated iff } \mu_{C_{i,k}}(x_k) > \epsilon \tag{2.1}$$

Donde $C_{j,k}$ es el k-ésimo conjunto difuso de la j-ésima variable lingüística de entrada, $\mu_{C_{i,k}}(\cdot)$ es la función de membresia correspondiente al conjunto difuso $C_{j,k}$ y x_k es el k-ésimo elemento del vector de entrada $\vec{x}$.

2. **El núcleo de un sistema difuso:**

   Reglas Difusas: Al evaluar todos los elementos del vector de entrada y al realizar la combinación de todos los conjuntos difusos activados se generá un conjunto de antecedentes, los cuáles son comparados con la Base de Reglas difusas y se obtienen un conjunto de r reglas de inferencia activadas de la siguiente forma:

$$\begin{aligned} R_1&\text{: if } V_I^1 = C_{1,k} \ \wedge \ \cdots \ \wedge \text{ if } V_I^d \Rightarrow C_{d,k'}\text{, then } V_O^1 = B_{1,k} \\ &\vdots \\ R_r&\text{: if } V_I^1 = C_{1,k} \ \wedge \ \cdots \ \wedge \text{ if } V_I^d \Rightarrow C_{d,k'}\text{, then } V_O^{N_O} = B_{N_O,k'} \\ &\vdots \end{aligned} \tag{2.2}$$

   Donde V_I^j es la j-ésima variable lingüística de entrada, V_O^i es la i-ésima variable lingüística de salida y $B_{N_O,k}$ es el conjunto k-ésimo de la i-ésima variable lingüística de salida.

   Motor de Inferencia: El valor de membresía de cada uno de los conjuntos difusos de salida es obtenido mediante la combinación de t-norma y s-norma, un ejemplo es:

$$\mu_{V_O^i} = \max_{R_l, l=1,\cdots,r} \Big(\min\left(\mu_{V_I^1}(x_1), \cdots, \mu_{V_I^d}(x_d)\right) \Big) \tag{2.3}$$

   En esta ecuación solo se toman en cuenta las reglas, R_l, que tengan como consecuente al conjunto difuso $B_{i,k}$ asignado a V_O^i para obtener el valor de membresía $\mu_{V_O^i}$. Cabe mencionar que las funciones mín y máx son la generalización de los operadores lógicos $\wedge$ y $\vee$, respectivamente. Los sistemas difusos que utilizan un motor de inferencia como el de la ecuación 2.3 son conocidos como sistemas difusos del tipo Mamdani [21].

3. **Desfuzificación:** Para obtener un valor "real" en las variables lingüísticas de salida se hace uso de los conjuntos difusos de salida, activados por las reglas difusas en

conjunto con los valores de membresía calculados por las t-norma y s-norma, a lo cual se le llama el proceso de desfuzificación; una forma de obtener el valor es mediante el método de centroide. Llamemos VR_q al valor real obtenido de la desfuzificación obtenida para la q-ésima variable de salida.

$$VR_q = \frac{\sum_{i=1}^{N_{FSOVq}} A_i \cdot R_i}{A_T} \tag{2.4}$$

En la ecuación 2.4 se tiene el método de desfuzificación por centroide, donde A_i y R_i son el área y el centroide, respectivamente, del i-ésimo conjunto difuso de la variable de salida q-ésima. A_T es la suma del área de cada uno de los conjuntos difusos de la variable de salida q-ésima, esto es, $A_T = \sum_i^{NOFSq} A_i$. Si el conjunto no es activado por alguna de las reglas, los valores A_i e R_i son 0. N_{FSOVq} es el número de conjuntos difusos de la variable de salida q-ésima.

Todo lo anterior se simplifica en la imagen de la Figura 2.1.

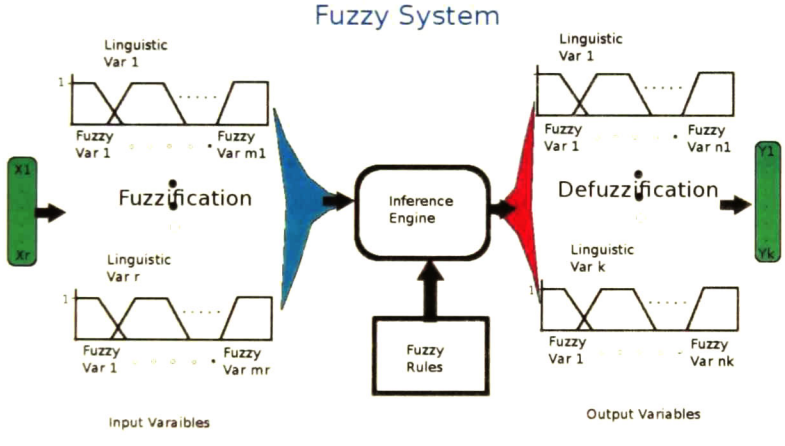


Figura 2.1: Composición de un Sistema Difuso.

2.1.3. Procesos de Dirichlet.

Un Proceso de Dircichlet (DP, por sus siglas en inglés) según Yee Whye Teh [29], introducidos por S. Ferguson [8] 1973, es un proceso estocástico usado en modelos Bayesianos no paramétricos de datos. En otras palabras, es una distribución sobre distribuciones, esto

es. cada muestra de un proceso de Dirichlet es en sí una distribución.

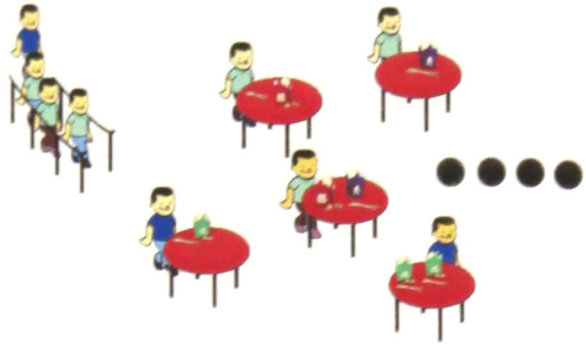


Figura 2.2: Proceso del Restaurante Chino (CRP, por sus siglas en inglés).

EL CRP (*Chinese Restaurant Process*) puede verse como un DP, por lo cual explicaremos los DP con el ejemplo del CRP. Como se observa en la Figura 2.2 el CRP es el proceso de determinar en que mesa se sentará el cliente que acaba de entrar al restaurante. En el restaurante se tiene un número de mesas infinito, donde se van sentado los clientes mientras van ingresando al restaurante. Dependiendo del número de mesas ocupadas, un nuevo cliente tiene una cierta probabilidad de sentarse en alguna de las mesas ocupadas y otra cierta posibilidad de sentarse en una mesa desocupada.

Los elementos del CRP se pueden ver de la siguiente manera: Los clientes representan los datos de la población de muestreo que queremos clasificar (sean univaribles o multivariables) y las mesas representán las clases a las que corresponden cada uno de los datos (clientes). De esta manera cada que llega un nuevo dato se tienen $k + 1$ posibilidades de ser clasificado, en alguna de las k clases ya formadas o creando una nueva clase con ese dato.

Matemáticamente se ve de las siguiente manera:

Definición 1 *Una distribución de Dirichlet [28] es una distribución sobre la probabilidad del simplex k-dimensional. Representación matemática del simplex k-dimensional:*

$$\Delta_\kappa = \{(\pi_1, \cdots, \pi_\kappa) : \forall \pi_k \geq 0, \sum_\kappa \pi_k = 1\} \tag{2.5}$$

Definición 2 *La variable aleatoria* $\vec{\pi} = (\pi_1, \cdots, \pi_\kappa)$ *es Dirichlet distribuido [28],* $(\pi_1, \cdots, \pi_\kappa)$ *Dirichlet*$(\alpha_1, \cdots, \alpha_\kappa)$ *con parámetros* $\vec{\alpha} = (\alpha_1, \cdots, \alpha_\kappa)$*, sí:*

$$P(\pi_1, \cdots . \pi_\kappa) = \frac{\Gamma(\sum_\kappa \alpha_k)}{\prod_\kappa \Gamma(\alpha_k)} \prod_\kappa \pi_k^{\alpha_k - 1} \tag{2.6}$$

Esta definición puede ser relacionada con el ejemplo del CRP de la siguiente manera: la variable aleatoria $\vec{\pi}$ representa a los clientes que van ingresando al restaurante, el vector de parámetros $\vec{\alpha}$ representa a una mesa y la Ecuación 2.6 es la probabilidad de que el cliente se siente en esa mesa.

Definición 3 *Un DP es una distribución sobre medidas de probabilidad [28], el cual tiene dos parámetros:*

- G_0 *es la distribución base, que es como la media del DP.*
- α *es el parámetro de fuerza, que es como la varianza inversa del DP.*

Definición 4 *Sea* $\mathbf{X}$ *un espacio medible, Sea* G_0 *una medida de probabilidad sobre el espacio y* α *un número real positivo. Escribimos* $G \sim DP(\alpha, G_0)$ *[28] si para cualquier partición* $(A_1, \cdots, A_r)$ *de* $\mathbf{X}$*:*

$$(G(A_1), \cdots, G(A_r)) \sim Dirichlet(\alpha G_0(A_1), \cdots, \alpha G_0(A_r)) \tag{2.7}$$

2.1.4. Segmentación de Imágenes por Color.

Para nuestro trabajo la segmentación en Visión por Computadora es la división de la imagen en regiones o conjuntos de píxeles. Este proceso en muchas de las ocasiones nos ayuda a realizar un análisis más fácil de la imagen.

Por lo cual, la segmentación por color es aquella división de la imagen en regiones (o conjuntos de píxeles), donde la diferencia del valor arrojado por la métrica para un miembro de la región con los valores de las métricas de los demás miembros de la misma región es mínima en comparación con la diferencia de la métrica de los demás miembros de las otras regiones.

Digamos que cada elemento de la imagen I se define como $I_{i,j}$ (que es el píxel del i-ésimo renglón de la j-ésima columna). Supongamos que existen κ regiones en la imagen.

Denotemos como A_k al k-ésimo conjunto (o partición) de píxeles que define a la región k-ésima e $I^k_{i',j'}$ como el píxel que pertenece a ese conjunto ($I^k_{i',j'} \in A_k$), se tiene una partición:

$$P_A = (A_1, \cdots, A_\kappa) \tag{2.8}$$

Donde A esta definido como:

$$A = \bigcup_{k=1} A_k \tag{2.9}$$

El conjunto de resultante A es el conjunto formado por todos los píxeles de la imagen I:

$$A = \bigcup_{i=0}^{N-1} \bigcup_{j=0}^{M-1} \{I_{i,j}\} \tag{2.10}$$

Para cualesquiera dos regiones k y l se tiene que

$$A_l \cap A_k = \emptyset \tag{2.11}$$

Por último, un píxel $I_{i,j}$ pertenece a una región k en el instante t, dada la métrica $\mathfrak{M}$, si

$\forall I^k_{i',j'} \in A^t_k$ y $\forall I^{k'}_{i'',j''} \in A^t_{k'} = A^t - A^t_k$ se cumple que:

$$|\mathfrak{M}^t_k(I_{i,j}) - \mathfrak{M}^t_k(I^k_{i',j'})| < |\mathfrak{M}^t_{k'}(I_{i,j}) - \mathfrak{M}^t_{k'}(I^{k'}_{i'',j''})| \tag{2.12}$$

Donde el conjunto $A^t \in A$ representa a los elementos ya clasificados en cada una de las A^t_k regiones en el tiempo t.

Esto es, la diferencia entre los valores arrojados por la métrica del nuevo píxel con los píxeles de la región a la que pertenece es menor que la diferencia del valor de la métrica de ese mismo píxel con los valores arrojados por la métrica de los píxeles pertenecientes a las otras regiones.

$\mathfrak{M} = d(\cdot,\cdot)$, es la función distancia entre dos elementos. En este caso $\mathfrak{M}^t_k(\cdot) = d(\cdot, C^t_k)$ es la distancia entre un elemento de la imagen y el elemento que representa a la región A^t_k. La distancia y la forma de representar la región depende de las características del problema que se desee resolver.

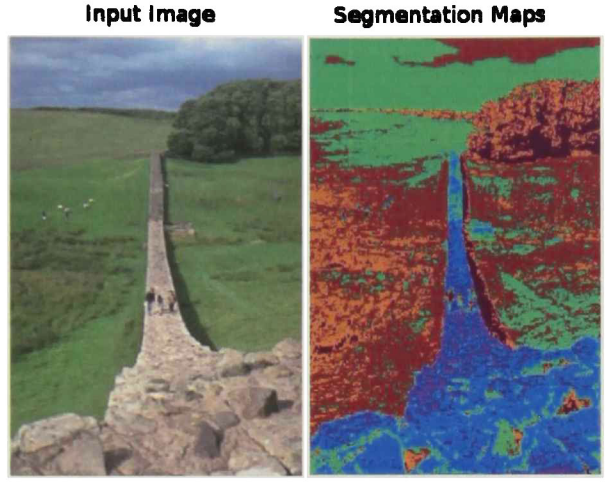


Figura 2.3: Ejemplo de Segmentación de una imagen por color.

2.1.5. Algoritmo de Optimización Metaheurístico: ACO.

El algoritmo ACO se basa en el comportamiento de las hormigas [5]; donde cada hormiga deja un rastro de feromonas sobre el camino que recorrieron, de esta forma se "guarda" el camino recorrido para que pueda ser usado posteriormente por otras hormigas o por la misma hormiga que recorrió el camino y poco a poco el camino más corto para llegar a la comida se llena de una cantidad de feromonas considerablemente mayor a cualquier otro camino que se haya hecho por las hormigas y así, la mayoría de las hormigas termina recorriendo este camino.

Los diferentes recorridos de las hormigas se ilustra en la Figura 2.4. El camino de color rojo es el camino más corto y el que es recorrido por más hormigas. Este problema puede ser trasladado a Teoría de Grafos al problema del camino más corto (*the shortest path*).

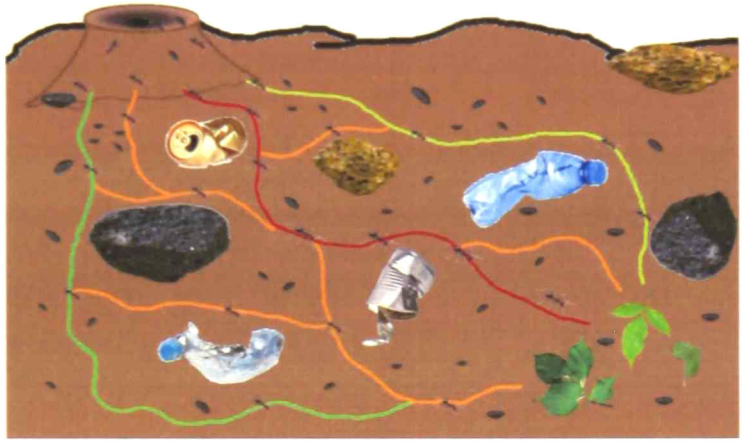


Figura 2.4: Representación del ACO [33].

El algoritmo utiliza las siguientes variables:

Una lista ***tabu***, asociada a cada una de las hormigas, la cual guarda los vértices recorridos por la hormiga.

- La matriz τ almacena la cantidad de feromonas que hay entre los vértices r y s del grafo.

  La matriz $\Delta\tau$ almacena la cantidad de feromonas de la i-ésima iteración.

  ρ es el coeficiente de evaporación para las feromonas.

- $b(i)$ es el número de hormigas en el i-ésimo vértice.

  V es el conjunto de vértices.

- E es el conjunto de aristas.

  δ es la matriz de costos, depende del problema.

  $\eta(r,s) = 1/\delta(r,s)$ la visibilidad.

  Q_1 y Q_2 constantes previamente definidas.

El algoritmo del ACO es el siguiente:

1. **ACO-Algorithm()**
2. $\rho \leftarrow initEvapCoefficinet()$ // Initialize the evaporation coefficient
3. $t \leftarrow 0$
4. $\tau^t \leftarrow initTrailIntensity()$
5. $b^t \leftarrow initAntPlace()$
6. $\forall r, s \in V$ do $\Delta\tau^t(r,s) \leftarrow 0$
7. // Run the algorithm
8. Repeat until tabu list is full:
9. For $i \leftarrow 1$ to n do // for every vertex
10. For $k \leftarrow 1$ to $b^t(i)$ do
11. Choose the vertex to move to with $aggTransProb\_rs(r,s)$ // r is the actual vertex and s is the possible next vertex to move on.
12. Move the k-th ant to the choosen vertex (in $b^{t+1}(s)$)
13. $\Delta\tau_k^{t,t+1}(r,s) \leftarrow aggDeltaTau\_ith\_rs(r,s,k,MODEL\_TYPE)$
14. $\Delta\tau(r,s)^{t,t+1} \leftarrow \Delta\tau_k^{t,t+1}(r,s) + \Delta\tau(r,s)^{t,t+1}$
15. For $\forall r, s \in V$ do
16. $\tau^{t+1}(r,s) \leftarrow \rho\tau^t(r,s) + \Delta\tau^{t,t+1}(r,s)$
17. $P^{t+1}(r,s) \leftarrow aggTransPob_rs(r,s)$
18. Memorize the shortest path (or the path with the lowest cost) and empty all tabu list
19. If(not end test) then
20. $t \leftarrow t+1$
21. $\forall r, s \in V$ do $\Delta\tau^t(r,s) \leftarrow 0$
22. go to line 7
23. else
24. return the shortest path

La descripción de las funciones utilizadas en el algoritmo son las siguientes:

$initEvapCoefficinet()$: Esta función inicializa el valor del coeficiente de evaporación ρ.

$initTrailIntensity()$: Esta función inicializa la cantidad de feromonas que debe de haber inicialmente, se basa en el problema y lo cual puede ayudar a que la solución sea encontrada en un menor número de iteraciones.

$initAntPlace()$: Esta función inicializa aleatoriamente en que vértice se encontrará cada una de las hormigas.

$aggTransProb\_rs(r, s)$: Esta función calcula el valor de la probabilidad de transición entre dos vértices, r y s.

$aggDeltaTau\_ith\_rs(r, s, k, MODEL\_TYPE)$: Esta función regresa el rastro de feromona que debe ser aplicado al borde por el cual a pasdo la hormiga haciendo uso de los pesos y las constantes Q_1 y Q_2.

El comportamiento del algoritmo es la siguiente:

Los pasos del 2 al 6 del algoritmo son para inicializar la posición de las hormigas en el grafo, así como la inicialización de las feromonas y el coeficiente de evaporación que se usará.

Los ciclos de las lineas 9 y 10 indican que se deben buscar a todas las hormigas y hacer que se muevan en cada iteración, en otras palabras, que la hormiga se mueva al siguiente vértice. Estos ciclos se detendrán hasta que todas las listas *tabu* de cada una de las hormigas estén llenas. Las líneas que van dentro de estos ciclos son de la 11 a la 14 que contienen la parte donde las hormigas se mueven entre los vértices para formar la solución.

- La hormiga tiene un subconjuntos de vértices a los cuales puede moverse en el tiempo t, este conjunto de posibles vértices esta dado por la "conexión" que existe del vértice en el que se encuentra la hormiga con los demás (dado por la definición del grafo).
- Para seleccionar el vértice al cual se moverá, se usa una función de probabilidad de transición que conjunta la cantidad de feromonas y el costo asignado a ese enlace. Al seleccionar el vértice la hormiga deja una cierta cantidad de feromonas en el borde, que por lo regular es predeterminada.

  En las líneas de la 15 a la 17 una vez que están llenas todas las listas *tabu* se actualizan las feromonas de cada borde así como el valor de la función de probabilidad de transición se guarda el camino más corto. Si no cumple con las restricciones se vuelve a realizar todo el procedimiento.

  En las líneas de la 18 a la 24 de otra manera el algoritmo regresa el resultado.

Cabe mencionar que el camino más corto depende del problema; puede referirse al camino de menor o de mayor costo y no al que tenga menos vértices.

Este tipo de algoritmos es muy usado en problemas de combinatoria, los cuales pueden ser trasladados a una representación en grafos donde es difícil y costoso encontrar la mejor solución al revisar cada una de las combinaciones y por lo cual el algoritmo busca obtener el resultado óptimo.

2.2. Segmentación de Imágenes por Color

Las tres características utilizadas en las técnicas de segmentación de imágenes son: color, textura y forma. En muchas de las ocasiones se utiliza una combinación de estas para lograr mejores resultados, ya sea en la detección de objetos, en detección de bordes, en detección puntos de interés , entre otros problemas. En esta sección se presentan algunos trabajos que tienen como principal característica el color para la segmentación de la imagen.

En trabajo sobre segmentación de imágenes, Wang y Suter [32], proponen un método que utiliza información local y global de la homogeneidad de la imagen. Este algoritmo hace uso del *mean shift algorithm* [10] en las componentes de matiz de intensidad del subespacio de color HSV y toma en cuenta la propiedad cíclica de la componente de matiz.

El *mean shift algorithm*, presentado por Fukunaga y Hostetler[10] en 1975, es un algoritmo iterativo no paramétrico el cual es usado para detectar la moda estadística, para el clustering, entre otras aplicaciones. Este algoritmo considera al espacio de características como una función de densidad de probabilidad empírica.

F. kurogullo et. al. [14] proponen un método para segmentación de imágenes multibanda, el cual se basa sobre la segmentación de subconjuntos de bandas utilizando multiumbrales seguido por la fusión del resultado de la segmentación de "canales". En este trabajo utiliza el espacio de color más comúnmente usado en sistemas de vídeo y de imagen, el RGB. Para ello se utilizan histogramas de dos dimensiones dados por las combinaciones de los componentes RB, RG y BG; los cuales son procesados por un algoritmo *peak-picking* para efectuar la multi-umbralización.

En el trabajo propuesto por Gholamiparvar et. al [23] se utiliza un PSO para optimizar un sistema difuso para clasificación del color y segmentación de imágenes. Cada partícula codifica un conjunto de reglas difusas. El espacio de color utilizado es el HSL. Este trabajo se utilizo muestras obtenidas de la *Robocup* al igual que el trabajo de James et. al [3].

2.2.1. Aprendizaje de Sistemas para Segmentación de Imágenes

Nosotros exploraremos la segmentación vía clustering por lo cual en esta sección trataremos algunos de los trabajos que abordan el tema del aprendizaje de sistemas para segmentación de imágenes los cuales hacen uso de técnicas de clustering.

Don y Xie [6] proponen un sistema para segmentación de imágenes, en su trabajo utilizan algoritmos de *clustering* para poder separar los colores para ser segmentados, y basado en redes neuronales para realizar el aprendizaje del sistema. Los colores son representados en el espacio de color L*u*v (CIELuv [30]) modificado. El sistema propuesta comprende segmentación no supervisada y segmentación supervisada.

La segmentación no supervisada se logra mediante una aproximación de dos niveles: reducción del color y clustering por color. En la reducción del color, los colores de la imagen son proyectados dentro de un pequeño grupo de prototipos haciendo uso del aprendizaje por *self-organizing map* [18] [12] (SOM). En el clustering por color la técnica de *simulated annealing* [17] [40] (SA) busca los clusters óptimos de los prototipos de SOM. La segmentación no supervisada involucra el aprendizaje de color y la clasificación de píxeles. En el aprendizaje del color se define un color prototipo para representar una región esférica en el espacio de color, se hace uso de *hierarchical prototype learning* [6] (HPL) para generar los diferentes tamaños de los prototipos de color. Los píxeles de la imagen son clasificados mediante el *matching* de los prototipos de colores.

Este problema se a abordado bastante en la *Robocup Soccer* para lidiar con el problema de identificar tanto al balón, como a sus compañeros y contrincantes. En este tipo de problemas es necesario que el procesamiento sea rápido por lo cual James et. al [3] proponen un sistema con un nuevo umbral clasificador, un fusionador de regiones para el cálculo de componentes conectadas y por último un sistema de separación y clasificación para reunir varias características de región y ordenarlos por tamaño. Este sistema tiene la capacidad de clasificar un píxel en más de uno de los 32 colores de base diferentes que tienen al hacer uso de operaciones AND. En este trabajo utilizan el espacio de color YUV.

2.3. Sistemas Difusos

En el trabajo de Kosko [19] demuestran que un sistema difuso aditivo puede apróximar uniformemente cualquier función real continua en un dominio compacto a cualquier grado de preicisión, mediante la covertura de su grafo con parches difusos y promediando parches que se sobreponen. Concluyen que el poder de aproximación de los sistemas difusos recae mayormente en la libertad del modelo más que en su interpretación difusa.

2.3.1. Aprendizaje de Sistemas Difusos

Jorge Casillas et. al [4] proponen un sistema de aprendizaje basado en ACO para el aprendizaje de las reglas difusas. Los conjuntos difusos de cada una de las variables de entrada y de salida están ya determinados de antemano. El problema en si es determinar los pares antecedente-consecuente que genere la base de reglas difusas necesarias para el sistema. En otras palabras al dar una entrada del conjunto de datos de entrenamiento se obtienen un cierto número de antecedentes, dado que se activo un cierto número de conjuntos difusos de las entradas; a su vez se tienen los datos de salida para esa entrada en especifico lo cual nos genera un cierto número de consecuentes. Ahora el problema puede ser mapeado a un problema de asignación; necesitamos asignar un número de consecuentes a cada antecedente que nos permita minimizar el número de reglas pero as u vez obteniendo un comportamiento óptimo del sistema difuso. Para lograrlo Jorge Casillas et. al proponen la solución del problema de asignación cuadrática (QAP, por sus siglas en inglés) mediante el uso del ACO.

En el trabajo de Van Der Lee et. al [31] se propone un algoritmo generalizado para el refinamiento automático para el desarrollo de MPC (*Model Predictive Control*), caso principalmente utilizado para sistemas MIMO (múltiples entradas múltiples salidas) en la parte de MOFDM (*multi-objective fuzzy decision-making*). Los métodos o técnicas utilizados son: combinación de un GA con *multi-objective fuzzy decision-making.*

En el trabajo de Chia-Feng y Chi-Yen [16] se propone un sistema difuso auto generado con habilidades de aprendizaje. Los métodos o técnicas que utilizan son: El algoritmo OSAC (*on-line self-aligning clustering*) para la creación de los conjuntos difusos el cual trata de evitar o minimizar el *overlping* y un APSCO (optimización por enjambre de partículas cooperativas) para la obtención de las reglas difusas. Esta combinación nos da un sistema de aprendizaje de sistemas difusos capaz, no solo de aprender las reglas

difusas sino también, de poder construir los conjuntos difusos sin la necesidad de algún conocimiento previo sobre estos.

2.4. Procesos de Dirchlet: Teoría y Aplicaciones

En los últimas décadas los Procesos de Dirichlet han ganado fuerza en el área de Ciencias Computacionales al ser capaces de modelar datos sin la necesidad de tener conocimiento *a priori* de ellos, por ejemplo en clustering donde nos permite obtener el número de *clusters* o particiones de los datos.

En minería de datos Tianbing Xu et. al [35] proponen dos modelos para la solución al problema del *clustering* evolutivo basados en los Procesos de Dirirchlet. Ellos prueban sus resultados con datos sintéticos y con una base de datos de mensajes de diferentes tópicos de los "*20 Newsgroups*" [1] obteniendo resultados mejores que con las otras técnicas conocidas.

Sabri Boutemedjet et. al [2] presenta una aproximación no supervisada para el problema de selección y extracción de características haciendo uso de mezcla de distribuciones generalizadas de Dirichlet (GD, por sus siglas en inglés). Con su método definen un nuevo modelo de mezcla que puede extraer características no Gausianas independientes sin perdida de exactitud. Esta técnica muestra buenos resultados en el problema de la categorización de imágenes de objetos.

Zare A. y Gader P. D. [39] hacen uso de los procesos de Dirichlet en imágenes hiperespectrales para encontrar el número de *endmembers*. Este algoritmo provee un estimado de espectros de *endmembers*, mapas de proporciones y el número de *endmembers* necesarios para un escenario.

N. Bouguila et. al proponen en [1] un algoritmo no supervisado para el aprendizaje de un modelo de mezcla finita de datos multivariados. Esta basado en la distribución de Dirichlet y hace uso del *Maximum Likelihood* (ML) y de los métodos de puntuación de Fisher para la extimación de parametros de una mezcla de Dirichlet. Las aplicaciones a este trabajo son para la estimación de histogramas artificiales, "resumen" de la base de datos de imágenes para la extracción eficiente, modelado de piel humana y su aplicación para la detección de piel en base de datos multimedia.

[1] http://kdd.ics.uci.edu/databases/20newsgroups/

2.4.1. Aplicaciones de los Procesos de Dirichlet

Existen dos aplicaciones principales de los DP: la clasificación de documentos y la segmentación de imágenes.

En la aplicación de clasificación de documentos se encuentra el trabajo de Shafiei, M.M. y Milios, E.E. [20] en el cual presentan un modelo generativo para el *clustering* simultaneo de documentos y términos. El modelo que presentan es un modelo Bayesiano jerárquico de cuatro niveles, en donde cada documento es modelado como una mezcla aleatoria de tópicos/temas del documento, donde cada tópico/tema es una distribución sobre algunos segmentos del texto. A su vez cada uno de estos segmentos en el documento puede ser modelado como una mezcla de tópicos de palabras donde cada tópico es una distribución sobre palabras.

Ruizhang Huang et. al [13] proponen una nueva aproximación, llamada *Dirichlet Process Mixture Feature Partition* (DPMFP), para descubrir la estructura latente de *cluster* basado en el modelo DPM (*Dirichlet Process Mixture*) sin requerir el número de *clusters* como entrada, una mejora de su trabajo del 2010 [37] en el cual proponían la aproximación llamada *Dirichlet Process Mixture Feature Selection* (DPMFS). Esto es, proponen el DPMFP para encontrar el número apropiado de *clusters* en los cuales los documentos deben ser particionados.

En las aplicaciones de segmentación de documentos existen trabajos como el de K.A.D.N.K. Wimalawarne proponen en [34] un método de procesos variacionales de Dirichlet para la segmentación de imágenes. Aplica un *kd-tree* para particionar imágenes y procesos de Dirichlet para agrupar los valores de los colores de píxel en esas particiones.

Otra aplicación para la segmentación de imágenes es la de Lu Yi-su y Chen Wu-fan proponen en [36] el uso del modelo no-parametrico de mezcla de procesos de Dirichlet (MDP, por sus siglas en inglés) para la segemntación de imágenes, el cual obtiene el número de clases automáticamente sin ninguna inicialización.

2.5. Propuesta: Uso de Procesos de Dirichlet y ACO para el Aprendizaje de Sistemas Difusos

Hasta el momento, en nuestra revisión del estado del arte, no nos hemos encontrado con algún trabajo que use los DP para el aprendizaje de Sistemas Difusos. Por lo cual es un área de oportunidad para investigación.

Los trabajos más relevantes para nuestro trabajo se condensan en la siguiente tabla:

Autores	Técnicas / Algoritmos	Supervisado	No Supervisado	Descripción
J. Casillas et. al [4] 2000	ACO y FS	x		Aprendizaje de de FS
Chia-Feng y Chi-Yen [16] 2009	OSAC, APSCO y FS	x		Aprendizaje de FS
Gholamiparvar et. al [23]	PSO y FS	x		Segmentación de Imágenes
Zare A. y Gader P. D. [39] 2008	DP	x		Detección de *endmembers*
N. Bouguila et. al [1] 2004	DP, ML y *Fisher scoring methods*		x	Detección de piel
K.A.D.N.K. Wimalawarne [34] 2008	VDP		x	Segmentación de Imágenes

MR: *Magenetic Resonance*	**MDP:** *Dirichlet Process Mixture*
VDP: *Variational Dirichlet Processes*	**APSCO:** *Ant and Particle Swarm Cooperative Optimization*
FS: *Fuzzy System*	**ACO:** *Ant Colony Optimization*
PSO: *Particle Swarm Optimization*	**OSAC:** *Online Self-Aligning Clustering*

Capítulo 3

Uso de los Procesos de Dirichlet y Optimización por Colonia de Hormigas para el Aprendizaje de Sistemas Difusos

Nuestra hipótesis es que el uso de los DP nos ayudará a tener un mejor modelado de los conjuntos difusos que los arrojados por las técnicas mencionadas en el capítulo anterior. Esto se supone ya que los conjuntos difusos se pueden ver como una partición del espacio de las variables lingüísticas dados por la frecuencia de los datos, aunado a la relación que existe entre la teoría de probabilidad y la lógica difusa.

3.1. Definición de la Propuesta:

El objetivo de este trabajo es desarrollar e implementar un sistema para el aprendizaje de sistemas difusos, el cual se aplicará en la segmentación de imágenes por color.

En otras palabras, nuestro sistema será capaz de obtener los conjuntos difusos de cada una de las variables lingüísticas, tanto de entrada como de salida, así como el conjunto de reglas difusas (o base de conocimientos); en nuestro caso el sistema tomará como entrada una imagen y ciertos parámetros iniciales, como el número inicial de clusters entre otros.

Como ya se mencionó en el Capítulo 1 nos centraremos en resolver: la obtención del número mínimo de reglas difusas, evitar el *overlapping* de conjuntos difusos y que sea un aprendizaje No supervisado.

Se puede ver a los conjuntos difusos como un clustering de los datos de la variable lingüística, ya sea de entrada o de salida y así modelarlos usando modelos probabilísticos. Por lo cual se propone el uso de DP para la creación de los conjuntos difusos.

El aprendizaje de las reglas difusas puede ser visto como un problema de combinatoria y de asignación por lo cual usaremos la técnica de ACO para resolver esta parte del problema.

3.1.1. Aprendizaje de los Conjuntos Difusos usando DP

Para el aprendizaje de los conjuntos difusos haremos uso del algoritmo propuesto por N. Bouguila et. al [1], a continuación describiremos el desarrollo que presento en su trabajo.

Si el vector $\vec{X} = (X_1, \cdots, X_d)$ sigue una distribución de Dirichlet la función de densidad conjunta esta dada por:

$$p(X_1, \cdots . X_k) = \frac{\Gamma(\vec{\alpha})}{\prod_{i=1}^{d+1} \Gamma(\alpha_i)} \prod_{i=1}^{d+1} {X_i}^{\alpha_i - 1} \tag{3.1}$$

donde:

$$\sum_{i=1}^{d} X_i < 1$$

$$|\vec{X}| = \sum_{i=1}^{d}, \quad 0 < X_i < 1 \quad \forall i = 1 \cdots d$$

$$X_{d+1} = 1 - |\vec{X}|$$

$$|\vec{\alpha}| = \sum_{i=1}^{d+1} \alpha_i, \quad \alpha_i > 0 \quad \forall i = 1 \cdots d+1$$

Una Mezcla de Dirichlet con M componentes se define como:

$$p(\vec{X}|\Theta) = \sum_{j=1}^{M} p(\vec{X}|j, \Theta_j) P(j) \tag{3.2}$$

donde $P(j)$ $(0 < P(j) < 1$ y $\sum_{j=1}^{M} P(j) = 1)$ son las proporciones de mezcla y $p(\vec{X}|j, \Theta_j)$ es la distribución de Dirichlet. El símbolo Θ se refiere al conjunto de parámetros a ser estimado

$\Theta = (\vec{\alpha_1}, \cdots, \vec{\alpha_M}, P(1), \cdots, P(M))$

donde α_j es el vector de parámetros para la población (o cluster) j-ésima. Se utiliza la notación $\Theta_j = (\vec{\alpha_j}), \quad \forall j = 1, \cdots, M.$

Utilizamos la estimación por *Maximum Likelihood* (ML) que es la aproximación más utilizada en este tipo de problemas, por lo cual se convierte en la búsqueda de los parámetros que que maximicen la función de densidad de probabilidad de la muestra. En otras palabras:

$$\max_{\Theta} p(\vec{X}|\Theta) \tag{3.3}$$

Haciendo uso de los multiplicadores de Lagrange y las restricciones $\sum_{j=1}^{M} P(j) = 1 \quad P(j) > 0 \forall j \in [1, M]$ se tiene que la función a maximizar es:

$$\Phi(\vec{\mathbf{X}}, \Theta, \Lambda) = \sum_{i=1}^{N} \ln\left(\sum_{j=1}^{M} p(\vec{X_i}|j, \Theta_j) P(i)\right) + \Lambda\left(1 - \sum_{j=1}^{M} P(j)\right) + \mu \sum_{j=1}^{M} P(j) \ln(P(j)) \tag{3.4}$$

donde Λ es el multiplicador de Lagrange, $\sum_{j=1}^{M} P(j) \ln(P(j))$ es un criterio de entropía usado en casos de Mezcla de Gaussianas y μ especifica la compensación entre el *likelihood* requerido de los datos y el número de componentes a ser encontrados.

Para obtener las proporciones de mezcla en cada iteración se realiza de la siguiente manera:

$$P(j)^t = \frac{N_j^{t-1}}{N} \tag{3.5}$$

Donde N_j^{t-1} es el número de elementos que pertenecen al cluster j-ésimo dado los parámetros obtenidos en la iteración $t-1$, en otras palabras

$$C_j^{t-1} = \{X_i | \Theta_j = arg \max_{\Theta_{j'}} \{p(X_i|j, \Theta_{j'})^{t-1}\}\}$$
$$N_j^{t-1} = \#(C_j^{t-1}) \tag{3.6}$$

Para la estimación de los parámetros $\vec{\alpha}$ se utilizo el *Fisher Scoring Method* el cual es una variante del método de Newton-Raphson. Para eliminar la restricción sobre los parámetros α se utiliza el cambio de variable $\alpha_{j,\ell} = e^{\beta_{j,\ell}}$. EL esquema iterativo del método de Fisher esta dado por la ecuación:

$$\begin{bmatrix} \hat{\beta}_{j,1} \\ \vdots \\ \hat{\beta}_{j,k+1} \end{bmatrix}^{t} = \begin{bmatrix} \hat{\beta}_{j,1} \\ \vdots \\ \hat{\beta}_{j,k+1} \end{bmatrix}^{t-1} + \begin{bmatrix} Var(\hat{\beta}_{j,1}) & \cdots & Cov(\hat{\beta}_{j,1}, \hat{\beta}_{j,k+1}) \\ \vdots & & \vdots \\ Cov(\hat{\beta}_{j,k+1}, \hat{\beta}_{j,1}) & \cdots & Var(\hat{\beta}_{j,k+1}) \end{bmatrix}^{t-1} \times \begin{bmatrix} \frac{\partial}{\partial \hat{\beta}_{j,1}} \Phi \\ \vdots \\ \frac{\partial}{\partial \hat{\beta}_{j,k+1}} \Phi \end{bmatrix}^{t-1}$$

donde j es el número de la clase: $1 < j < M$. La matriz de varianza-covarianza del método de Fisher es obtenida como la inversa de la matriz de información de Fisher **F**. La matriz de información de Fisher es:

$$\mathbf{F} = I_{\ell_1,\ell_2} = -E\left[\frac{\partial^2}{\partial\beta_{j,\ell_1}\partial\beta_{j,\ell_2}}\Phi(\vec{\mathbf{X}}, \Theta, \Lambda)\right] \tag{3.8}$$

En la obtención de la matriz de información de Fisher se hace uso de $p(j|\vec{X}_i, \Theta_j)$, la cual es la probabilidad a posteriori de la distribución de Dirichlet ($p(\vec{X}_i|j, \Theta_j)$), a partir de Bayes obtenemos:

$$p(j|\vec{X}_i, \Theta_j) = \frac{p(\vec{X}_i|j, \Theta_j)p(j)}{p(\vec{X}_i)} \tag{3.9}$$

Despreciando el valor de normalización $p(\vec{X}_i)$ tenemos:

$$p(j|\vec{X}_i, \Theta_j) \propto p(\vec{X}_i|j, \Theta_j)p(j) \tag{3.10}$$

la cual será usada para el calculo de la probabilidad a posteriori.

El algoritmo propuesto pro N. Bouguila et. al. [1] se compone de dos algoritmos: El algoritmo de inicialización y el algoritmo de estimación de mezclas de Dirichlet.

- **Algoritmo de inicialización:**

1. **INITIALIZATION**$(\vec{X}, M)$**:**
2. Apply the fuzzy C-means to obtain the elements, covariance matrix and mean of each component.
3. Apply the MM (Method of Moments) for each component j to obtain the vector of parameters $\vec{\alpha_j}$.
4. Assign the data clusters, assuming that the current model is correct.
5. Update the $P(j)$ using the following:
6. $P(j) = \frac{\#OfElementsInClass_j}{N}$

7. If the current model and the new model are sufficiently close to each other, terminate, else go to 3.
8. **Output:** Vector of parameters $\tilde{\alpha}$.

Algoritmo de estimación de mezclas de Dirichlet:

1. **Input:** Dimensional data $X_i, i = 1, \cdots, N$ and over-specified number of clusters M.
2. **DIRICHLET_MIXTURE_ESTIMATION**$(\vec{X}, M)$:
3. INITIALIZATION$(\vec{X}, M)$.
4. Update the $\vec{\alpha_j}$ using(3.7).
5. Update the $P(j)$ using (3.5).
6. If $P(j) < \epsilon$ discard component j, go to 4.
7. If the convergence test is passed, terminate, else go to 4.
8. **Output:** Vector of parameters $\vec{\alpha_j}$ of the cluster j and the number of cluers M.

Las ecuaciones para la obtención de los parámetros $\vec{\alpha_j}$ a partir de los parámetros obtenidos por el MM son las siguientes:

$$\alpha_{j,l} = \frac{(x'_{1,1} - x'_{2,1})x'_{1,\ell}}{x'_{2,1} - (x'_{1,1})^2}. \quad \ell = 1, \cdots, d \tag{3.11}$$

y

$$\alpha_{j,d+1} = \frac{(x'_{1,1} - x'_{2,1})(1 - \sum_{l=1}^{d} x'_{1,\ell})}{x'_{2,1} - (x'_{1,1})^2} \tag{3.12}$$

donde

$$x'_{1,l} = \frac{1}{N}\sum_{i=1}^{N} x_{i,l}, \quad \ell = 1, \cdots, d+1 \tag{3.13}$$

$$x'_{2,1} = \frac{1}{N}\sum_{i=1}^{N} x_{i,1}^2 \tag{3.14}$$

Conjuntos Difuso como Funciones de Densidad de Probabilidad

La función de memebresía de un conjunto difuso C esta definida como $\mu_C : \Re \rightarrow [0, 1]$, y la definición de una función de densidad de probabilidad esta definida como $f : \Re \rightarrow [0, 1]$. Notesé que en general:

$$\int_x f(x)dx \neq \int_x \mu_C(x)dx \tag{3.15}$$

Por lo cual, podemos decir que la función de membresía en un conjunto difuso es una medida de la probabilidad de que el elemento x pertenezca al conjunto difuso C.

Los conjuntos difusos pueden ser vistos como una partición del espacio donde se mueve la variable x, por lo cual cada conjunto difuso puede ser visto como una clase a la cual puede pertenecer x.

Estas relaciones nos permite hacer uso de técnicas probabilísticas de *Clustering*, como lo son los DP, para la determinación de las funciones de membresía de los conjuntos difusos.

3.1.2. Aprendizaje de Reglas Difusas usando ACO

Para la creación y aprendizaje de las reglas difusas nos basaremos en el trabajo de J. Casillas et. al [4]; el cuál hace uso de la técnica del ACO para obtener las reglas difusas.

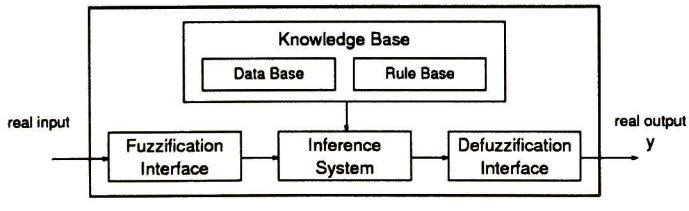


Figura 3.1: Estructura genérica de un Sistema Difuso lingüístico basado en reglas [4].

El trabajo se centra en la obtención de la base de reglas, las cuales forman parte de la base de conocimientos (KB, por sus siglas en inglés) de los Sistemas Difusos Basados en Reglas (FRBS, por sus siglas en inglés). La base de datos (DB, por sus siglas en inglés) ya tiene definidos los los conjuntos difusos. Se puede observar la estructura de un FRBS

lingüístico en la Figura 3.1.

La estructura de una regla lingüística difusa es la siguiente:

$$R_i: \text{ IF } x_1^0 \text{ is } C_{r_1,1} \text{ and } \cdots \text{ and } x_d^0 \text{ is } C_{r_d,d} \text{ THEN } y^0 \text{ is } B_{r_\ell,\ell'} \tag{3.16}$$

donde

- $(x_1^0, \cdots, x_d^0)$ es el vector de entrada, con d el número de variables lingüísticas de entrada.
- y^0 es la salida.
- $C_{r_\ell,\ell}$ es la r_j-ésima etiqueta lingüística activada de la ℓ-ésima variable lingüística de entrada, la cuál esta definida por la función de membresía $\mu_{i,\ell}(\cdot)$.
- $B_{r_\ell,\ell'}$ es el r_ℓ-ésima etiqueta lingüística activada de la ℓ' variable lingüística de salida.

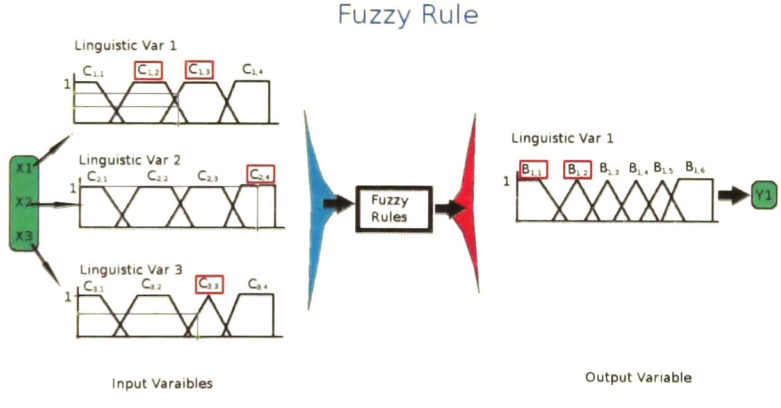


Figura 3.2: Ejemplo: Reglas Difusas.

El ejemplo de la Figura 3.2 nos muestra que dada una entrada se tiene un cierto número de reglas difusas. La entrada activa un número de conjuntos difusos para cada una de las variables lingüísticas y se busca en la base de reglas difusas si se tienen algunas reglas con esa combinación de antecedentes, en el ejemplo las reglas que se obtienen son:

$$R_1 : \text{IF } x_1 \text{ is } C_{2,1} \text{ and } x_2 \text{ is } C_{4,2} \text{ and } x_3 \text{ is } C_{3,3} \text{ THEN } y \text{ is } B_{2,1} \quad (3.17)$$
$$R_2 : \text{IF } x_1 \text{ is } C_{3,1} \text{ and } x_2 \text{ is } C_{4,2} \text{ and } x_3 \text{ is } C_{3,3} \text{ THEN } y \text{ is } B_{1,1}$$

cabe mencionar que el valor y es el que queremos calcular, el cuál se determinará en el motor de inferencia.

El sistema de inferencia esta basado sobre la aplicación del Modus Ponens generalizado [7], que es una extensión del Modus Ponens de lógica clásica. Esto se hace mediante la regla composicional de inferencia, la cuál en su forma más simple se reduce a:

Dado el conjunto difuso A', que representa la parte premisa x es A' y la relación difusa $A \Rightarrow B$, que representa si x es A entonces y es B, un conjunto difuso B' es inferido como:

$$B' = A' \circ R \Rightarrow \mu_{B'}(y) = s[t(\mu_{A'}(x), \mu_R(x, y))] \quad (3.18)$$

donde

$\mu_{A'}(x)$ función de membresía del conjunto difuso A'.

$\mu_R(x, y)$ función de membresía de la relación difusa.

- $\mu_{B'}(y)$ es la función de membresía para el conjunto difuso inferido B'.

t es es el operador conjuntivo (o *t-norm*, en nuestro caso será la función mín()).

s es el operador de disyuntivo (o *s-norm*, en nuestro caso será la función máx()).

El problema del aprendizaje de reglas difusas (FRL, por sus siglas en inglés) son trabajos basados con un conjunto $E = \{e_1, \cdots, e_N\}$ de entradas-salidas, donde $e_k = (x_1^k, \cdots, x_d^k, y^k) = (\vec{x}^k, y^k)$, el cuál representa el comportamiento del problema a ser resuelto. Previamente se ha definido la DB en este tipo de problemas, el cuál esta compuesto de las definiciones de los conjuntos difusos de cada una de las variables de entrada y salida.

En el trabajo de J. Casillas et. al [4] se trata al problema como uno de optmización combinatorial para poder ser representado como un grafo bipartita y así utilizar del ACO para su solución; de esta manera se trata al problema como la asignación de consecuentes

a un número fijo de antecedentes con respecto a un criterio de optimalidad.

Al estar lidiando con un problema de asignación, éste suele ser representado de la misma manera que el usado para resolver el problema de asignación cuadrática (QAP, por sus siglas en inglés) con algunos cambios, en este caso el orden de la asignación de los consecuentes es irrelevante respecto a un antecedente.

La construcción del grafo es de la siguiente manera:

1. **Determinar los antecedentes de las reglas:** Un antecedente de una regla R_i, $i = 1, \cdots, N_r$, denotado por $A_{i'}$ con $i' = 1, \cdots, N_r$,se definida por una combinación:

$$A_{i'} : x_1 \text{ is } C_{r_1,1} \text{ and } \cdots \text{ and } x_d \text{ is } C_{r_d,d} \tag{3.19}$$

toma parte del grafo sii:

$$\exists e_k = (x_1^k, \cdots, x_d^k) \in E \text{ such that } \mu_{C_{r_1,1}}(x_1,^k) \cdot \cdots \cdot \mu_{C_{r_d,d}}(x_d^k) \neq 0 \tag{3.20}$$

Esto es, existe al menos un ejemplo situado en el subespacio de entrada difuso definido por los antecedentes considerados en la regla.

2. **Enlazar los antecedentes a los consecuentes:** El antecedente $A_{i'}$ será enlazado o relacionado al consecuente B_j, $j = 1, \cdots, N_c$ sii se cumple la siguiente condición:

$$\exists e_k = (x_1^k, \cdots, x_d^k) \in E \text{ such that } \mu_{C_{r_1,1}}(x_1^k) \cdot \cdots \cdot \mu_{C_{r_d,d}}(x_d^k) \cdot \mu_{B_j}(y^k) \neq 0 \tag{3.21}$$

Esto es, existe al menos un ejemplo situado en el subespacio de entrada difuso que está cubierto por tal consecuente.

El valor de feromona inicial de cada asignación es obtenida por:

$$\eta_i = \max_{j=1,\cdots,N_c} \eta_{ij} \tag{3.22}$$

$$\tau_0 = \frac{\sum_{i=1}^{N_a} \eta_i}{N_a} \tag{3.23}$$

Esto es, el valor inicial de la feromona será el valor medio del camino construido tomando el mejor consecuente en cada regla de acuerdo a la información heurística:

$$\eta_{ij} = \max_{e_k \in E} \left(\min \left(\mu_{C_{i,1}}(x_1^k), \cdots, \mu_{C_{i,d}}(x_d^k), \mu_{B_j}(y^k) \right) \right) \tag{3.24}$$

La función de adaptabilidad que usan es el error cuadrático medio (MSE, por sus siglas en inglés) con respecto a la salida deseada y a la salida arrojada por la RB obtenida de la i-ésima hormiga, la cuál se define como:

$$MSE(RB_i) = \frac{1}{2 \cdot |E|} \sum_{e_k \in E} (y^k - F_k(\vec{x}^k))^2 \tag{3.25}$$

donde $F_i(\vec{x}^k)$ es la salida obtenida del FRBS (usando el RB generado por la hormiga i-ésima, RB_i) cuando recibe la entrada $\vec{x}^k$ (componente de entrada del ejemplo e_k) y y^k es la salida deseada. Entre mas cerca se encuentre la medida de cero nos da la mejor solución.

Algunas de las adaptaciones propuestas del ACO por J. Casillas et. al [4] son:

El conjunto de nodos alcanzables desde A_i será $J_k(i) = \{j \text{ such that } \eta_{ij} \neq 0\}$ en la regla de transición considerada por el algoritmo ACO.

- La cantidad de feromona que la i-ésima hormiga deja sobre el borde perteneciente a la solución construida por ella será $\frac{1}{MSE(RB_i)}$, con RB_k siendo la base de reglas generada por la hormiga i-ésima.

En la regla de actualización local del rastro de feromonas el modo usual de calcular $\Delta\tau_{ij}$ es $\Delta\tau_{ij} = \tau_0$.

3.2. Componentes del Sistema y Algoritmo.

3.2.1. Sistema.

El diagrama de la Figura 3.3 nos muestra los componentes de cada uno de los módulos y la manera en que fluye el proceso.

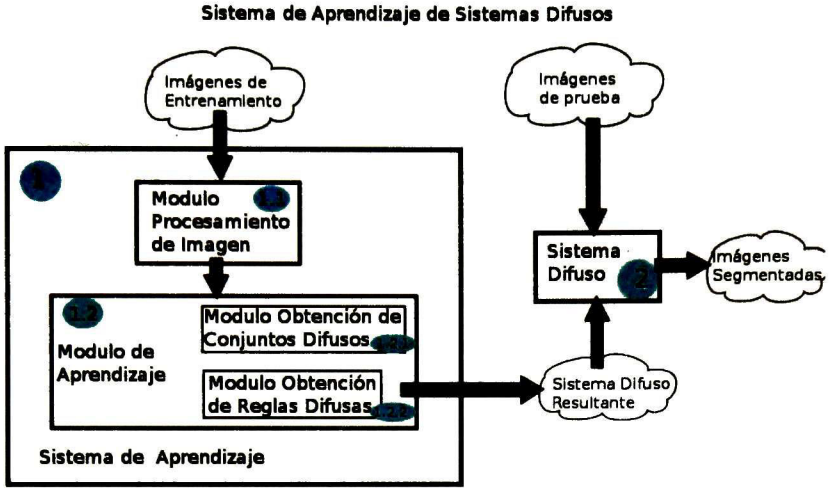


Figura 3.3: Diagrama del Sistema.

Módulo de pre-procesamiento de imagen: En este módulo (Módulo 1.1 de la Figura 3.3) se realiza un pre-procesamiento de la imagen, normalización de los datos de la imagen y/o un cambio a un espacio de color

Módulo de DP para Clustering: Aquí se obtendrán los conjuntos difusos para cada una de las variables de entrada y salida (Módulo 1.2.1 de la Figura 3.3).

Módulo de ACO:
Uso del algoritmo ACO para obtener las reglas del sistema difuso (Módulo 1.2.2 de la Figura 3.3).

El flujo de los datos es de la siguiente manera:

1. En primera instancia los datos de la o las imágenes entran para ser procesados en el módulo Sistema de Aprendizaje. Enseguida los datos entran al módulo de procesamiento de imagen el cual busca dar un pre-procesamiento a la imagen para obtener mejores resultados de los módulos posteriores; en esta etapa solo se realiza un cambio de espacio de color, del RGB al HSV.

2. Después de ser procesada la imagen, pasa al módulo de obtención de conjuntos difusos, esto se logra con el uso de los DP. Los DP automáticamente determinan cada

uno de los de conjuntos de cada una de las entradas, que en este caso serán los componentes de la imagen HSV por separado, así como cada uno de los conjuntos de salida de la clase.

3. Cuando se obtienen los conjuntos difusos se pasa al módulo de obtención de reglas difusas. Para esto se realiza una primera aproximación del número de reglas al hacer las combinaciones de los conjuntos difusos de cada una de las entradas por los datos de la imagen con los conjuntos difusos activados de la variable de salida, llamemos a esta técnica por fuerza "bruta".

4. Luego de realizar la obtención mediante fuerza "bruta" de estas reglas se dispone a hacer uso del ACO para la minimización de las reglas y se almacena este nuevo conjunto de reglas.

5. Por último, se hace la prueba del Sistema Difuso aprendido con la imagen que nos sirvió para el entrenamiento.

3.2.2. Algoritmo.

El algoritmo propuesto sigue los siguientes pasos:

1. **Input:** ImageVector **I**
2. **Preprocessing Image**(I)**:**
3. Separate the R,G and B componets in the vectors $\{\mathbf{X_r}, \mathbf{X_g}, \mathbf{X_b}\}$.
4. Normalize the components and the image vector. $\{\mathbf{X_r}, \mathbf{X_g}, \mathbf{X_b}, \mathbf{X}\}$.
5. **Finding the Fuzzy Sets** $(\mathbf{X_r}, \mathbf{X_g}, \mathbf{X_b}, \mathbf{X})$**:**
6. For each vector do:
7. Use the algorithm of N. Bouguila et. al [1] to find the number of clusters and their parameters (int his case the parameters $\vec{\alpha_i}$).
8. Obtain all the fuzzy variables of the each lingustic variable $(\mathbf{X_r}, \mathbf{X_g}, \mathbf{X_b}, \mathbf{X})$ given by the Bouguila algorithm.
9. **Finding the Fuzzy Rules:**
10. Use the fuzzy sets found by 5) and de vectors $(\mathbf{X_r}, \mathbf{X_g}, \mathbf{X_b}, \mathbf{X})$ to obtain all the rules.
11. Use the algorithm of J. Casilla et. al [4] to learn the fuzzy rules.
12. **Output:** Fuzzy System (Rules + Fuzzy Sets).

3.2.3. Implementación y Metricas.

Implementación.

El prototipado del módulo de DP se realizó en Matlab, en el futuro se buscara implementarlo en C/C++ y al final en CUDA.

La implementación del algoritmo ACO esta realizada en CUDA, dado que el algoritmo es paralelizable podemos aprovechar la tecnología NVIDIA y de esta forma reducir el costo en tiempo del algoritmo.

En el algoritmo del aprendizaje de reglas difusas mediante el ACO se mencionó que los antecedentes con los consecuentes formaban un grafo bipartita y el enlace que hay entre un consecuente y un antecedente es donde se dejara la feromona; a pesar de que está es la definición que se propone a la hora de implementar el algoritmo tiene una variación, la cuál consiste en vez de que las hormigas recorran el grafo bipartita de antecedente a consecuente y viceversa se propone:

> La hormiga en cada iteración se pone en un consecuente y de ahí de acuerdo a la función de probabilidad de transición selecciona un antecedente.
>
> Se pasa al siguiente consecuente de la lista de consecuentes y repite el proceso del punto anterior, sino hay posibilidad de seleccionar un antecedente continuará en el siguiente consecuente.

Una de las ventajas al proponer esta modificación es evitar el problema que surge cuando una hormiga llega a un vértice donde todos sus vecinos están seleccionados y ya no es posible moverse. Otra ventaja es que tendríamos todos los clusters obtenidos por el DP al seleccionar un antecedente como mínimo, de cada uno de los consecuentes en secuencia y esto disminuiría el error de clasificación.

Otra adaptación realizada al algoritmo de ACO para evitar quedarse en mínimos locales fue proveer al algoritmo con la posibilidad de no seleccionar el valor con mayor probabilidad de transición, para esto se genera un número aleatorio entre $[0, 1]$ y se compara con un umbral establecido, en nuestro caso es de 0.4; si el valor aleatorio esta por debajo del umbral se selecciona aleatoriamente uno de los 8 antecedentes con menor valor de transición de lo contrario se selecciona el antecedente con menor valor de probabilidad de transición.

Metricas.

Las Métricas utilizadas para verificar los resultados son las siguientes:

Rand index [25] o *Rand measure* es una medida de similaridad entre dos clusters, está relacionado con la precisión. Le medida se define como:

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.26}$$

donde:

- TP, número de verdaderos positivos, número de elementos clasificados en la clase que pertenecen a la clase.

  FP, número de verdaderos negativos, número de elementos clasificados en la clase que no pertenecen a la clase.
- TN, número de falsos positivos, número de elementos no clasificados en la clase que pertenecen a la clase.
- FN, número de falsos negativos, número de elementos no clasificados en la clase que no pertenecen a la clase.

- La medida F_1 [22] [26] (también conocida como *F-score* o *F-measure*) es una medida de la precisión de una prueba. Se considera tanto la razón de precisión P y la razón de recordatorio (*recall*) R de la prueba para calcularla. La medida puede ser usada para balancear la contribución de los falsos negativos al ponderar recordatorio a través de un parámetro $\beta \geq 0$. Sean la precisión y el recordatorio definidos como sigue:

$$\begin{aligned} P &= \frac{TP}{TP + FP} \\ R &= \frac{TP}{TP + FN} \end{aligned} \tag{3.27}$$

donde TP, FP, FN y TN se definen como en el inciso anterior. Calculamos la medida F como:

$$F_\beta = \frac{(\beta^2 + 1) \cdot P \quad R}{((\beta^2) \cdot P + R)} \tag{3.28}$$

El mejor valor de la métrica es alcanzado cuando $F_\beta = 1$ y el peor cuando $F_\beta = 0$.

El *Jaccard index* [15] es usado para cuantificar la similaridad entre dos conjuntos de datos, el cual toma valores entre 0 y 1. 1 significa que los dos conjuntos de datos son idénticos y un índice 0 indica que los dos conjuntos de datos no tienen elementos en común. El índice de Jaccard se define como:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{(TP + FP + FN)} \tag{3.29}$$

donde A y B son los conjuntos a compararse.

El *Fowlkes-Mallows index* [9] calcula la similitud entre los grupos devueltos por el algoritmo de agrupamiento y las clasificaciones de referencia. Cuanto mayor sea el valor del índice de Fowlkes-Mallows los grupos y las clasificaciones de referencia son más similares.

$$FM = \sqrt{\frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}} \tag{3.30}$$

- Una matriz de confusión [27] se puede utilizar para visualizar rápidamente los resultados de un algoritmo de clasificación (o agrupamiento). Se muestra cómo un clasificación es diferente a la clasificación base o de referencia.

Capítulo 4

Pruebas, Resultados y Análisis

4.1. Casos de Estudio

Para las pruebas se tomo de la base de datos para segmentación de Berkeley [24] usadas para algoritmos de segmentación por color de imágenes. El caso de estudio es realizar el aprendizaje del sistema difuso para segmentar una imagen, se harán seis pruebas diferentes (seis imágenes) y se hará la segmentación con el sistema propuesto y con los algoritmos K-Means y Fuzzy C-Means.

Como se mencionó en la Sección 4.1 se realizará una prueba sobre una imagen. El sistema aprenderá el sistema difuso para la segmentación de la imagen y el resultado obtenido será comparado con el resultado obtenido con los DP. Se seleccionó el espacio de color HSV ya que daba mejores resultados que con el espacio RGB, además que en en muchos de los trabajos sobre segmentación hacen uso de este espacio de color.

4.2. Resultados y Análisis

4.2.1. Caso I

En nuestro primer caso de estudio se puede observar en la Figura 4.1 la imagen original y su descomposición en los canales H, S y V, así como la imagen en escala de grises; también se observan los histogramas de cada uno de los canales así como el de escala de grises. El tamaño de la imagen es de 481×321, que nos da un total de 154401 píxeles.

El histograma se encuentra normalizado; por lo que se puede ver como el porcentaje de apariciones que tiene cada valor del rango de colores, en este caso de $[0, 255]$ representado en

la coordenada horizontal de los histogramas de la Figura 4.1, en la imagen. Si el histograma no estuviera normalizado nos diría el número de apariciones en la imagen de ese valor.

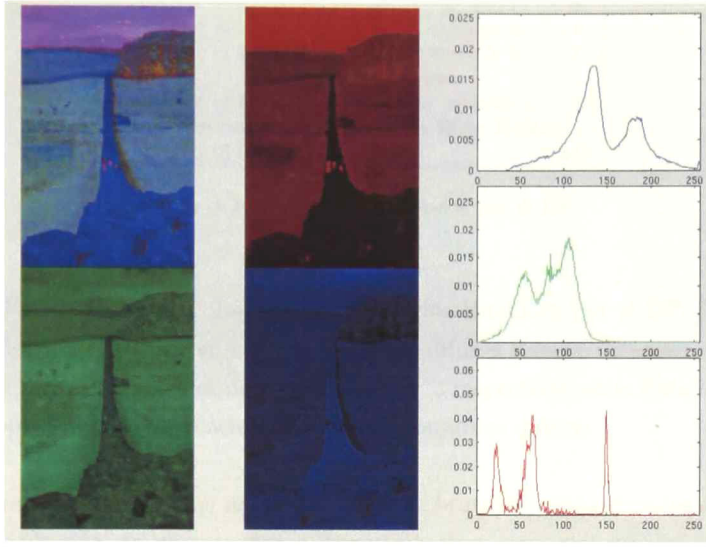


Figura 4.1: Imagen de prueba 1 y sus histogramas.

Dado que el algoritmo de DP [1] trata de modelar los datos con la mezcla de componentes (clases), se puede observar que cada uno de los picos nos dará un componente; tomando de ejemplo el histograma del componente B (en color azul) se pueden observar tres picos sobresalientes por lo cual podemos suponer que el algoritmo dará tres componentes, pero esta suposición no es completamente cierta, ya que algunas variaciones en los puntos contiguos del histograma provocarán que el número de clusters crezca (como se observará en la Figura 4.2).

El sub-sistema (o módulo) del DP fue alimentado con los siguientes datos:

Image:	image01.jpg
Number of clusters:	8
Max Iteration for MM:	30
Max Iteration for DP:	20
Min Diff Betas:	0.001
Min Diff Alphas:	0.001
Epsilon Pjs:	0.001

Tabla 4.1: Archivo de configuración para el DP.

El algoritmo DP nos arrojo los siguientes datos:

Input Variables.	
Number of Fuzzy Sets by input variable 1:	4
Number of Fuzzy Sets by input variable 2:	4
Number of Fuzzy Sets by input variable 3:	7
Output Variables.	
Number of Fuzzy Sets by output variable 1:	4
Knowledge Base (or Data Base Rules).	
Number of Fuzzy Rules:	448

Tabla 4.2: Resultados arrojados por el DP.

En la Tabla 4.2 se pueden observar los resultados obtenidos por el DP. Para las componentes H,S y V se obtuvieron 4, 4 y 7 conjuntos difusos respectivamente, representados en la Tabla 4.2 por las variables de entrada 1, 12 y 3 respectivamente. Para la variable de salida que representa la clasificación, se tienen 4 conjuntos difusos.

Las 448 reglas obtenidas fue mediante la fuerza bruta, se obtuvieron los conjuntos activados por cada una de las entradas y luego se realizó una combinación de todos ellos para obtener los antecedentes. Para obtener a los consecuentes se usaron las tres clases con mayor probabilidad según los DP y se hicieron las combinaciones con los antecedentes ya encontrados.

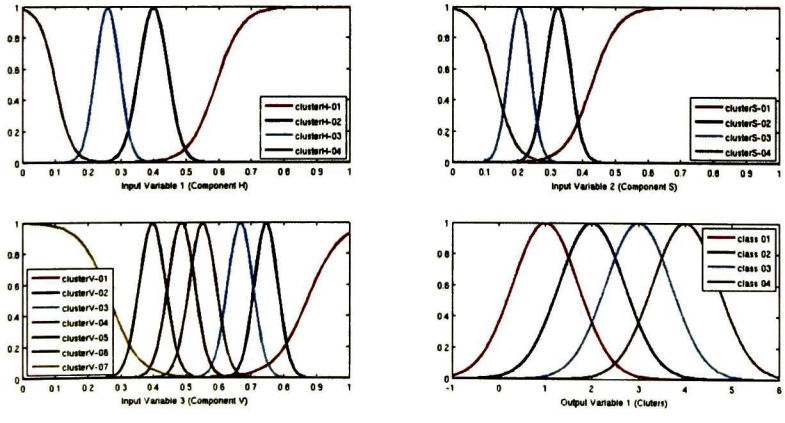


Figura 4.2: Los conjuntos difusos obtenidos mediante el algoritmo DP para la prueba 1.

En la Figura 4.2 se pueden observar los conjuntos difusos para cada uno de los canales (H, S y V) y para la variable de salida, los cuales fueron obtenidos mediante los clusters encontrados por el algoritmo DP (de N. Bouguila [1]).

En la Tabla 4.3 se muestran los datos de configuración para el ACO.

El número de hormigas.

El número de clases de hormigas, esto nos indica cuantos tipos de tamaño de hormigas diferentes hay.

Para cada clase de hormigas se tiene la razón del tamaño de las hormigas de acuerdo al número de reglas. Por ejemplo; —Class Rate of Ant 1: 0.05 nos indica que la clase 1 tiene de tamaño el 5 % de las reglas iniciales, esto es, en el caso de 448 reglas el tamaño de la hormiga sería de 22 reglas.

Tenemos también el número de hormigas que hay por clase.

El coeficiente de evaporación de las feromonas que estan esparcidas (***Evaporation Rate***).

El máximo número e iteraciones del algoritmo.

Number of Ants:	256
-Class of Ants:	8
-Class Types & Rate:	
—Class Rate of Ant 1:	0.05
—Class Rate of Ant 2:	0.05
—Class Rate of Ant 3:	0.075
—Class Rate of Ant 4:	0.075
—Class Rate of Ant 5:	0.10
—Class Rate of Ant 6:	0.10
—Class Rate of Ant 7:	0.14
—Class Rate of Ant 8:	0.14
-Number of Ants by Class:	
-Number of Ants by Class 1:	32
-Number of Ants by Class 2:	32
-Number of Ants by Class 3:	32
-Number of Ants by Class 4:	32
-Number of Ants by Class 5:	32
-Number of Ants by Class 6:	32
-Number of Ants by Class 7:	32
-Number of Ants by Class 8:	32
Evaporation Rate:	0.75
Maximum Number of Iterations:	150

Tabla 4.3: Archivo de configuración para el ACO.

La Tabla 4.4 nos muestra los resultados arrojados por el ACO. Podemos observar que se redujo de 448 reglas a 62 reglas, el error obtenido por ese subconjunto de 62 reglas es de 0.818500, este error es calculado por la Ecuación 3.25 del Capítulo 3. Se puede observar que el número total de iteraciones que realizó fue de 150 en un tiempo total de 481.5326 minutos (alrededor de 8 horas).

Knowledge Base (or Data Base Rules):	
Number of Fuzzy Rules:	62
Metrics:	
Error:	0.818500
Total time:	481.5326m(8hrs)
Average time by iteration:	3.2102m
Number of iteration:	150

Tabla 4.4: Resultados arrojados por el DP.

A pesar de que se obtuvo un error considerable el subconjunto de reglas obtenidas nos permite que todos los datos sean clasificados; en otras palabras, todos los píxeles de la imagen activan por lo menos una regla difusa. Se tiene un promedio de 3.2102 minutos por iteración.

En la Figura 4.3 se muestran los resultados arrojados por: el DP, el sistema difuso con reducción de reglas y dos segmentaciones haciendo uso de los algoritmos de K-Means y Fuzzy C-Means. En la parte superior izquierda se puede observar a la imagen en el espacio de color HSV, en el inferior izquierdo se puede observar a la imagen original (en el espacio de color RGB), en la columna del centro se tienen la imagen de la segmentación por DP (parte superior) y la segmentación por KM (parte inferior), en la parte superior derecha se encuentra la segmentación por el Sistema Difuso aprendido por el ACO y por último en la parte inferior derecha se encuentra la segmentación por Fuzzy C-Means.

En la Figura 4.3 se puede observar que cualitativamente el algoritmo de DP arroja los mejores resultados en comparación de los otros.

En la Figura 4.4 se muestra la comparación visual entre los clusters obtenidos por el DP y el FS.

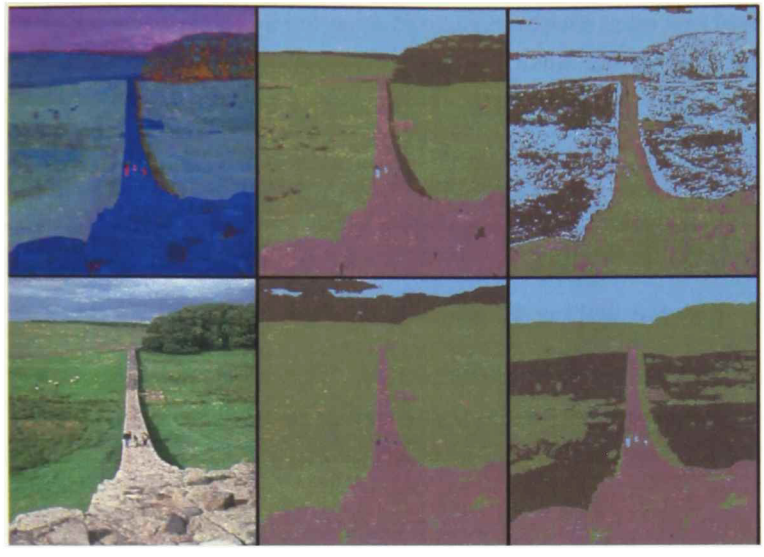


Figura 4.3: Reultados de los diferentes algoritmos de *clustering* utilizados.

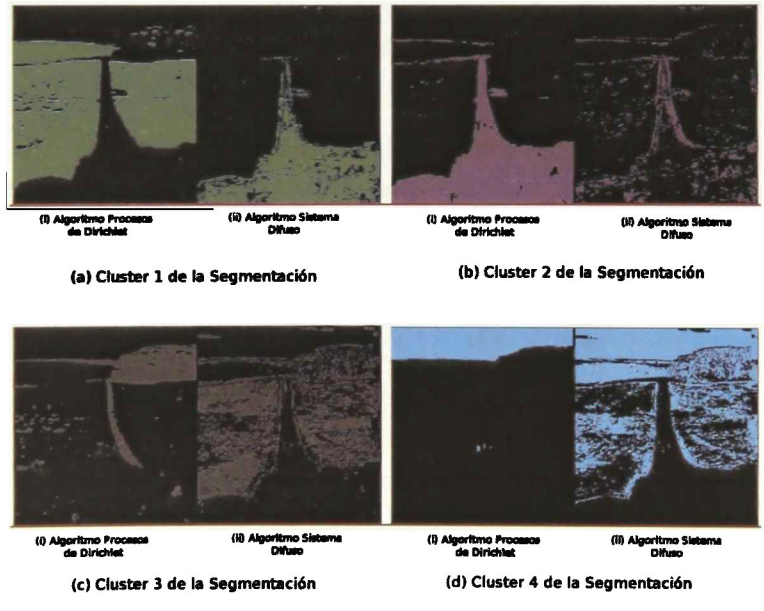


Figura 4.4: Comparación entre clases.

En la Tabla 4.5 se pueden observar las matrices de confusión entre las clases obtenidas por el algoritmo de DP contra las obtenidas por el Sistema Difuso (FS) aprendido, el algoritmo Fuzzy C-Means (FCM) y el algoritmo K-Means (KM).

	FS				FCM				KM			
	Clase 1	Clase 2	Clase 3	Clase 4	Clase 1	Clase 2	Clase 3	Clase 4	Clase 1	Clase 2	Clase 3	Clase 4
Clase 1 DP	1244	3195	36884	32292	24164	48	49402	1	73548	66	1	0
Clase 2 DP	33039	7909	1227	312	3109	39372	4	2	2515	39970	1	1
Clase 3 DP	394	820	4978	10353	16425	0	32	88	16454	0	91	0
Clase 4 DP	80	344	963	20367	0	0	0	21754	0	0	16636	5118

Tabla 4.5: Tabla de la Matriz de Confusión DP vs FCM, KM y FS.

En la Tabla 4.6 se puede observar las matrices de confusión entre las clases obtenidas por el algoritmo de FS aprendido contra las obtenidas por el algoritmo FCM y el algoritmo KM.

	FCM				KM			
	Clase 1	Clase 2	Clase 3	Clase 4	Clase 1	Clase 2	Clase 3	Clase 4
Clase 1 FS	2316	31551	784	106	2741	31910	62	44
Clase 2 FS	2782	7019	2097	370	4713	7185	212	158
Clase 3 FS	12488	699	29880	985	42297	770	579	406
Clase 4 FS	26112	151	16677	20384	42766	171	15876	4511

Tabla 4.6: Tabla de la Matriz de Confusión FS vs FCM y KM.

Para poder entender mejor los valores de la matriz de confusión presentaremos las métricas descritas en el Capítulo 3 en la Sección 3.2.3. A continuación se muestran las métricas por cluster de evaluación externa, tomando como referencia a la segmentación hecha por los DP.

Métrica	Class 1	Class 2	Class 3	Class 4
TP	1244	7909	4978	20367
TN	47273	107555	98782	89690
FP	33513	4359	39074	42957
FN	72371	34578	11567	1387
RI	0.314227	0.747819	0.672016	0.712800
P	0.035791	0.644685	0.113003	0.321632
R	0.016899	0.186151	0.300876	0.936242
F_β	0.018893	0.217023	0.225797	0.677365
$J(A,B)$	0.011612	0.168830	0.089502	0.314738
FM	0.024593	0.346423	0.184391	0.548748

Tabla 4.7: Métricas DP vs FS.

Se puede observar en la Tabla 4.7 que el porcentaje de decisiones acertadas, dado por la métrica RI hechas por el Sistema Difuso es bajo. Solo en tres de los *clusters* obtenemos un valor superior al 60 % mientras que en la clasificación del cluster 1 se obtiene un valor del 35.79 % lo cual es un valor muy bajo para un sistema de clustering. El porcentaje más alto lo tiene la clasificación del cluster 2, si se observa en la Figura 4.4 (b) se ve el por qué, a pesar de que clasifica pobremente los elementos del camino son muy pocos elementos que no pertenecen al cluster 2 lo que aumenta el porcentaje de decisiones acertadas para esta clase.

La precisión del clustering, representada por el valor P, no supera el 65 % y en promedio la precisión no llega ni al 28 %; lo cuál nos indica que nuestra precisión es deficiente. La mayoría de los valores de F_β están cercanos a cero, ésto nos indica que el sistema tiene poca precisión y no esta arrojando malos resultados.

Los valores bajos para la métrica $J(A, B)$ nos indican que los clusters obtenidos por el FS comparados con los obtenidos por el DP tienen muy pocos elementos en común. Esto nos confirma los malos resultados arrojados por el algoritmo. El valor arrojado por la métrica FM nos viene a confirmar los malos resultados del sistema dado que la similaridad entre los clusters es muy baja.

Por otra parte las métricas comparativas entre los DP, el K-Means y el Fuzzy C-Means nos muestra que la segmentación de estos últimos es inferior a la segmentación con los PD.

Métrica	Class 1	Class 2	Class 3	Class 4
TP	73548	39970	91	5118
TN	61817	111848	121218	132646
FP	18969	66	16638	1
FN	67	2517	16454	16636
RI	0.876711	0.983271	0.785675	0.892248
P	0.794967	0.998351	0.005440	0.999805
R	0.999090	0.940758	0.005500	0.235267
F_β	0.950289	0.951739	0.005488	0.277745
$J(A, B)$	0.794392	0.939299	0.002742	0.235256
FM	0.891204	0.969127	0.005470	0.484996

Tabla 4.8: Métricas DP vs KM.

A continuación se muestran las métricas entre los resultados de los DP y el Fuzzy C-Means.

Métrica	Class 1	Class 2	Class 3	Class 4
TP	24164	39372	32	21754
TN	61252	111866	88450	132556
FP	19534	48	49406	91
FN	49451	3115	16513	0
RI	0.553209	0.979514	0.573066	0.999411
P	0.552977	0.998782	0.000647	0.995834
R	0.328248	0.926683	0.001934	1.000000
F_β	0.357289	0.940258	0.001384	0.999164
$J(A,B)$	0.259412	0.925638	0.000485	0.995834
FM	0.426044	0.962058	0.001119	0.997915

Tabla 4.9: Métricas DP vs FCM.

En las Tablas 4.8 y 4.9 se puede observar mejores resultados que en la Tabla 4.7. Las métricas de DP con KM nos arroja los mejores resultados, pero en su clasificación de la clase 3 (Los árboles y el pasto de color oscuro) sus métricas quedan muy por debajo, como se puede observar en la Figura 4.5 que muestra las limitaciones del algoritmo.

En cuanto a la comparación cualitativa de las imágenes segmentadas resultantes, los DP realizan una mejor segmentación al separar de una forma muy buena cielo, pasto, árboles y camino; mientras que a los algoritmos de KM y FCM les cuesta hacer la separación entre pasto y árboles y una buena clasificación del cielo en el caso del KM. Esto demuestra las ventajas de los DP sobre las técnicas de KM y FCM, aunado a la ventaja de la necesidad de los KM y FCM por un número de clusters, mientras los DP no necesitan de esta información.

A pesar de los resultados poco alentadores arrojados se tiene que el sistema al reducir el número de reglas va mejorando los resultados como se puede observar en la Figura 4.5. Se observa que mientras mayor sea el número de reglas mayor será el ruido que se introduzca al sistema Figura 4.5 (a),(d) y (b); así mismo ocurre con un número de reglas bajo, donde se mezclan clusters y se tiene entradas que no son activadas por ninguna de las reglas difusas, lo cual provoca una no clasificación de los datos, Figura 4.5 (f).

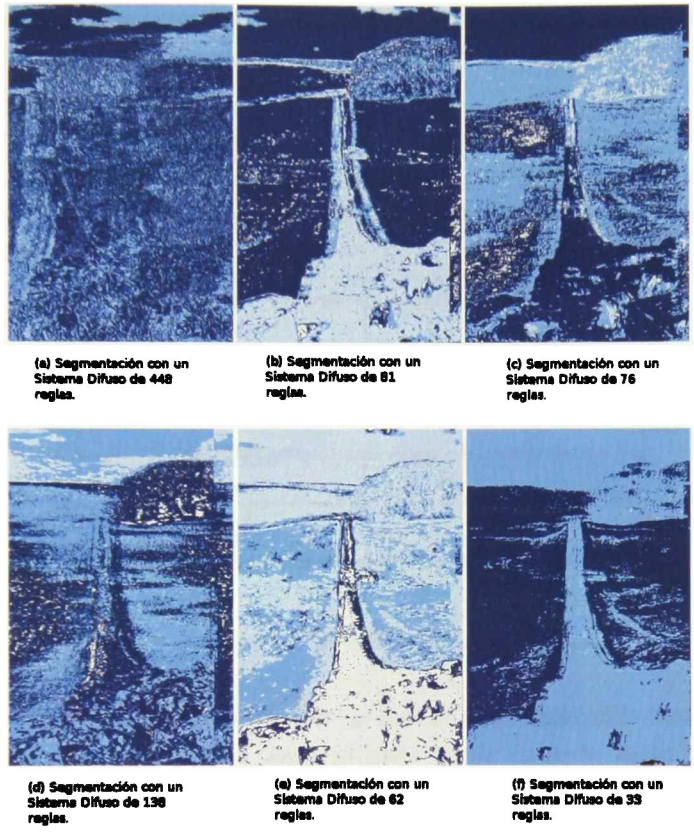


Figura 4.5: Diferentes resultados según el número de reglas.

El mejor resultado esta dado por la imagen del inciso (e) de la Figura 4.5, que son los resultados presentados anteriormente. Los tiempos obtenidos para cada uno de los algoritmos es el siguiente:

	DP	FCM	KM	FS
Tiempo	5.0292028hrs	171.0s	1.19s	6.0s

Tabla 4.10: Tiempos de los algoritmos DP, FCM, KM y FS.

Los tiempos presentados en la Tabla 4.11 nos indican que el algoritmo con menor tiempo de ejecución es el K-Means, le sigue el sistema difuso y el Fuzzy C-Means y al final el DP el cual presenta una gran desventaja con los otros algoritmos en tiempo, pero como la

implementación del sistema fue hecha en Matlab y los de los demás algoritmos en C/C++ no podemos decir que sea una mala elección.

El historial de la minimización del error por iteración y el número de hormigas, así como el tiempo de ejecución, se presentan en las gráficas de la Figura 4.6. Cabe mencionar que el tamaño de cada grupo de hormigas fue de: 22, 33, 44 y 62 reglas difusas.

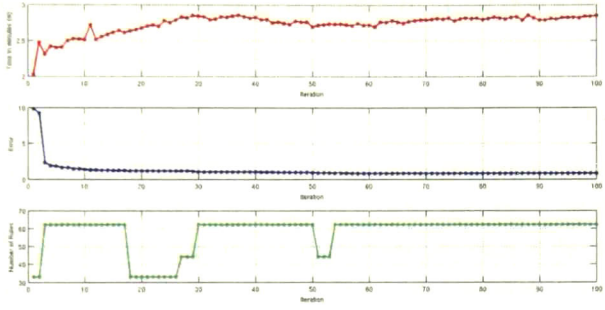


Figura 4.6: Graficas de evolución de las hormigas.

Se puede observar en la gráfica de error (gráfica de color azul) de la Figura 4.6 que en la primer iteración del algoritmo se tienen un error inicial alrededor de 10 y en la gráfica de color verde se observa que en esa misma iteración que el número de reglas es de 33. Mientras van avanzado el número de iteraciones se observa en la gráfica del error que este va disminuyendo, lo cual nos indica que el algoritmo esta buscando disminuir el error como se esperaba. La gráfica de color verde nos muestra la evolución el número de reglas de la mejor hormiga en cada iteración.

Al algoritmo solo le tomo dos iteraciones para llegar a un punto donde el error iba lentamente decrementándose con valores muy pequeños y de esta manera el número de reglas se empezó a estabilizar y dando como resultado 64 reglas.

La gráfica de color rojo de la Figura 4.6, nos indica el tiempo de procesamiento de cada una de las iteraciones. Se puede observar que el tiempo no se dispara y se mantiene dentro de un rango de 2 a 3 minutos, esto es debido a la gran cantidad de datos (154401 píxeles), al número de hormigas (256) y al tamaño de cada una de ellas.

El resultado condensado se presenta en la siguiente tabla:

Número de reglas iniciales	448
Número de reglas finales	62
Tiempo total DP	5.0292028hrs
Tiempo total ACO	481.5326m(8hrs)
Tiempo promedio ACO	3.2102m
Error mínimo del ACO	0.818500
Tiempo de ejecución del FS	6.0s

Tabla 4.11: Tiempos de los algoritmos DP, FCM, KM y FS.

4.2.2. Caso II

En nuestro segundo caso de estudio se puede observar (Figura 4.7) la imagen original y su descomposición en los canales H, S y V, así como la imagen en escala de grises; también se observan los histogramas de cada uno de los canales así como el de escala de grises. El tamaño de la imagen es de 481 × 321, que nos da un total de 154401 píxeles.

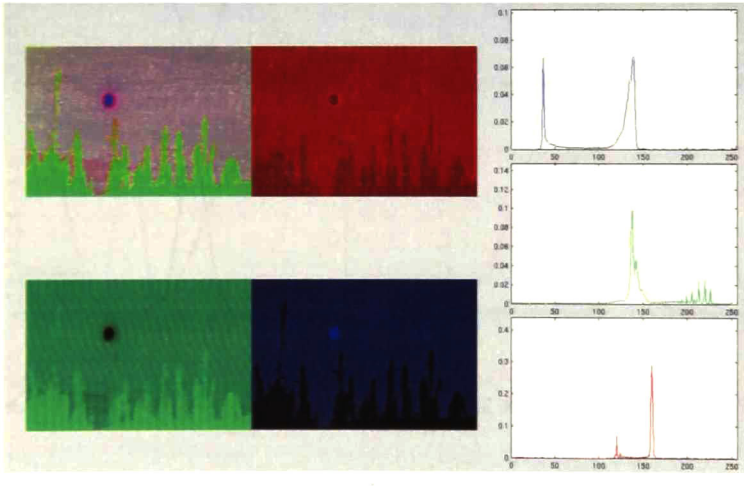


Figura 4.7: Imagen de prueba dos y sus histogramas.

El sub-sistema (o módulo) del DP fue alimentado con los mismos datos que los mostrados en la Tabla 4.1 de la prueba anterior.

En la Tabla 4.12 se pueden observar los resultados obtenidos por el DP. Para las com-

ponentes H, S y V se obtuvieron 2, 5 y 4 conjuntos difusos respectivamente, representados en la Tabla 4.12 por las variables de entrada 1, 12 y 3 respectivamente. Para la variable de salida que representa la clasificación, se tienen 4 conjuntos difusos y se obtuvieron un total de 160 reglas mediante la fuerza bruta.

Input Variables.	
Number of Fuzzy Sets by input variable 1:	2
Number of Fuzzy Sets by input variable 2:	5
Number of Fuzzy Sets by input variable 3:	4
Output Variables.	
Number of Fuzzy Sets by output variable 1:	4
Knowledge Base (or Data Base Rules).	
Number of Fuzzy Rules:	160

Tabla 4.12: Resultados arrojados por el DP.

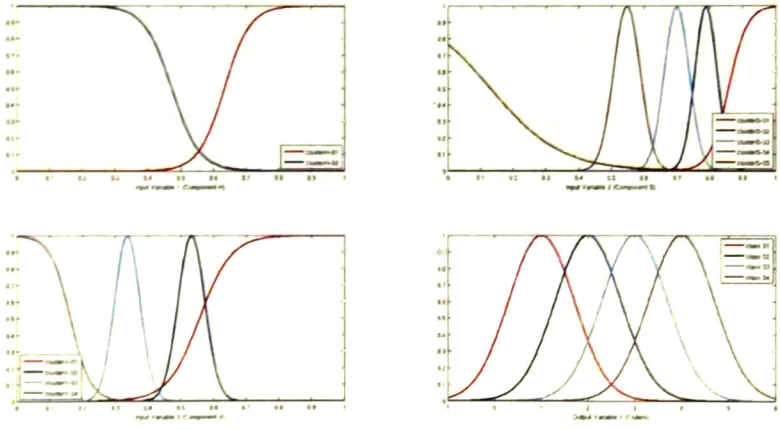


Figura 4.8: Los conjuntos difusos obtenidos mediante el algoritmo DP para la prueba dos.

En la Figura 4.8 se pueden observar los conjuntos difusos para cada uno de los canales (H, S y V) y para la variable de salida, los cuales fueron obtenidos mediante los clusters encontrados por el algoritmo DP.

En la Tabla 4.13 se muestran los datos de configuración para el ACO.

La Tabla 4.14 nos muestra los resultados arrojados por el ACO. Podemos observar que

se redujo de 162 reglas a 24 reglas, el error obtenido por ese subconjunto de 24 reglas es de 0.080281, este error es calculado por la Ecuación 3.25 del Capítulo 3. Se puede observar que el número total de iteraciones finales fue de 142 en un tiempo total de 672.8597 minutos (alrededor de 11 horas).

El error es mejor al obtenido en el caso de estudio anterior, se tiene un menor número de reglas con un total de 22 datos que no activaron ninguna regla, es decir, no se clasificaron correctamente.

Number of Ants:	256
-Class of Ants:	8
-Class Types & Rate:	
—Class Rate of Ant 1:	0.6
—Class Rate of Ant 2:	0.5
—Class Rate of Ant 3:	0.4
—Class Rate of Ant 4:	0.3
—Class Rate of Ant 5:	0.25
—Class Rate of Ant 6:	0.2
—Class Rate of Ant 7:	0.15
—Class Rate of Ant 8:	0.1
-Number of Ants by Class:	
-Number of Ants by Class 1:	32
-Number of Ants by Class 2:	32
-Number of Ants by Class 3:	32
-Number of Ants by Class 4:	32
-Number of Ants by Class 5:	32
-Number of Ants by Class 6:	32
-Number of Ants by Class 7:	32
-Number of Ants by Class 8:	32
Evaporation Rate:	0.75
Maximum Number of Iterations:	150

Tabla 4.13: Archivo de configuración para el ACO.

Knowledge Base (or Data Base Rules):	
Number of Fuzzy Rules:	24
Metrics:	
Error:	0.080281
Total time:	672.8597m(11hrs)
Average time by iteration:	4.7384m
Number of iteration:	142

Tabla 4.14: Resultados arrojados por el DP.

En la Figura 4.9 se muestran los resultados arrojados por: el DP, el sistema difuso con reducción de reglas y dos segmentaciones haciendo uso de los algoritmos de K-Means y Fuzzy C-Means.

En la parte superior (de la Figura 4.9), izquierda se puede observar a la imagen en el espacio de color HSV, en el inferior izquierdo se puede observar a la imagen original (en el espacio de color RGB), en la columna del centro se tienen la imagen de la segmentación por DP (parte superior) y la segmentación por KM (parte inferior), en la parte superior derecha se encuentra la segmentación por el Sistema Difuso aprendido por el ACO y por último en la parte inferior derecha se encuentra la segmentación por Fuzzy C-Means.

Figura 4.9: Resultados de los diferentes algoritmos de *clustering* utilizados.

En la Figura 4.9 se puede observar que cualitativamente el algoritmo de DP arroja los mejores resultados en comparación de los otros.

En la Figura 4.9 se muestra la comparación visual entre los clusters obtenidos por el DP y por el FS.

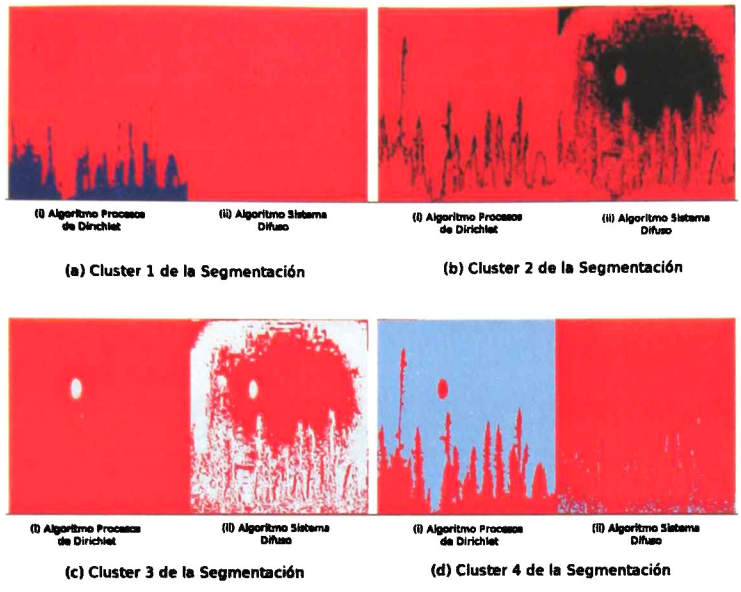


Figura 4.10: Comparación entre clases.

En la Tabla 4.15 se pueden observar las matrices de confusión entre las clases obtenidas por el algoritmo de DP contra las obtenidas por el Sistema Difuso (FS) aprendido, el algoritmo Fuzzy C-Means (FCM) y el algoritmo K-Means (KM). Se puede observar que el FCM nos arroja los peores resultados de todos.

	FS				FCM				KM			
	Clase 1	Clase 2	Clase 3	Clase 4	Clase 1	Clase 2	Clase 3	Clase 4	Clase 1	Clase 2	Clase 3	Clase 4
Clase 1 DP	0	1574	18708	4733	1398	23617	0	0	25015	0	0	0
Clase 2 DP	0	3969	9671	478	13032	190	896	0	11909	0	0	2209
Clase 3 DP	34	279	407	23	13	0	144	586	0	0	480	263
Clase 4 DP	0	63649	50876	0	57	0	52150	62318	5	0	0	114520

Tabla 4.15: Tabla de la Matriz de Confusión DP vs FCM, KM y FS.

En la Tabla 4.16 se puede observar las matrices de confusión entre las clases obtenidas por el algoritmo de FS aprendido contra las obtenidas por el algoritmo FCM y el algoritmo KM.

	FCM				KM			
	Clase 1	Clase 2	Clase 3	Clase 4	Clase 1	Clase 2	Clase 3	Clase 4
Clase 1 FS	0	0	0	34	0	0	34	0
Clase 2 FS	3554	1588	12884	51445	4490	0	108	64873
Clase 3 FS	10294	17660	40306	11402	27228	0	315	52119
Clase 4 FS	652	4559	0	23	5211	0	23	0

Tabla 4.16: Tabla de la Matriz de Confusión FS vs FCM y KM.

A continuación se muestran las Tablas 4.17 4.19 y 4.18 donde se muestran las métricas de precisión de cada uno de los algoritmos con respecto al algoritmo base (el algoritmo DP).

Métrica	Class 1	Class 2	Class 3	Class 4
TP	0	3969	407	0
TN	129352	74781	74403	34642
FP	34	65502	79255	5234
FN	25015	10149	336	114525
RI	0.837767	0.510036	0.484518	0.224364
P	0.0	0.057132	0.005109	0.0
R	0.0	0.281130	0.547779	0.0
F_β	-	0.157571	0.024627	-
$J(A,B)$	0.0	0.049849	0.005088	0.0
FM	0.0	0.126734	0.052902	0.0

Tabla 4.17: Métricas DP vs FS.

Métrica	Class 1	Class 2	Class 3	Class 4
TP	25015	0	480	114520
TN	117472	140283	153658	37404
FP	11914	0	0	2472
FN	0	14118	263	5
RI	0.922837	0.908563	0.998297	0.983957
P	0.677381	-	1.0000	0.978870
R	1.00000	0.0	0.646030	0.999956
F_β	0.913029	-	0.695249	0.995667
$J(A,B)$	0.677381	0.0	0.646030	0.978828
FM	0.823032	-	0.803760	0.989357

Tabla 4.18: Métricas DP vs KM.

Métrica	Class 1	Class 2	Class 3	Class 4
TP	1398	190	144	62318
TN	116284	116666	100612	39290
FP	13102	23617	53046	586
FN	23617	13928	599	52207
RI	0.762184	0.756834	0.652561	0.658079
P	0.096414	0.007981	0.002707	0.990684
R	0.055886	0.013458	0.193809	0.544143
F_β	0.061016	0.011834	0.012820	0.598057
$J(A,B)$	0.036677	0.005035	0.002677	0.541373
FM	0.073405	0.010364	0.022906	0.734217

Tabla 4.19: Métricas DP vs FCM.

Los tiempos obtenidos para cada uno de los algoritmos se muestran en la Tabla 4.21. Se puede apreciar que de los algoritmos implementados en C/C++ se puede apreciar que el que muestra el mayor tiempo corresponde al algoritmo FCM, el cuál arroja los peores resultados, como se observó con las métricas mostradas anteriormente.

	DP	FCM	KM	FS
Tiempo	6.2072hrs	65.97s	0.15s	1.31s

Tabla 4.20: Tiempos de los algoritmos DP, FCM, KM y FS.

El historial de la minimización del error por iteración y el numero de hormigas, así como el tiempo de ejecución, se presentan en las gráficas de la Figura 4.11. Cabe mencionar que el tamaño de cada grupo de hormigas fue de: 16, 24, 32, 40, 48, 64, 90 y 96 reglas difusas.

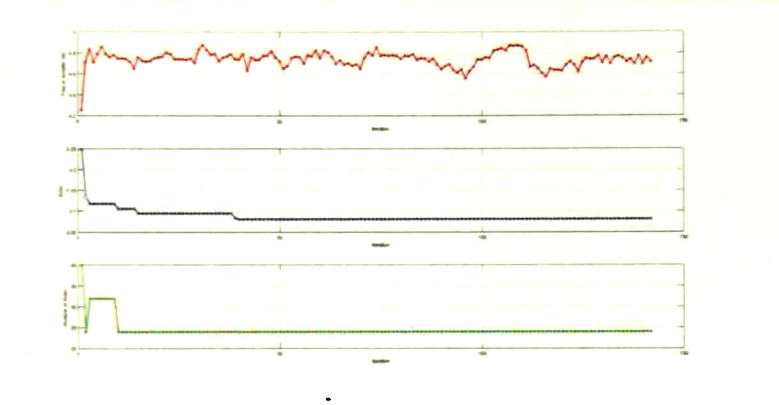


Figura 4.11: Gráficas de evolución de las hormigas.

Se puede observar que los comportamientos de las gráficas de error y de número de reglas por iteración realizan lo esperado. Se inicia con un número de reglas 40 para la mejor hormiga y en un lapso de 11 iteraciones el sistema consigue alcanzar el número mínimo de reglas. El error disminuye en la iteración 37 a pesar de que el número mínimo de reglas se alcanzo en iteraciones tempranas. Estos comportamientos nos indican que el algoritmo se comporta de acuerdo a lo esperado, minimiza el error entre los datos obtenidos por el FS con las reglas definidas por cada hormiga y los datos de entrenamiento y a su vez disminuye el número de reglas.

El resultado condensado se presenta en la siguiente tabla:

Número de reglas iniciales	160
Número de reglas finales	24
Tiempo total DP	6.2072hrs
Tiempo total ACO	672.8597m(11hrs)
Tiempo promedio ACO	4.7384m
Error mínimo del ACO	0.080281
Tiempo de ejecución del FS	1.31s

Tabla 4.21: Tiempos de los algoritmos DP, FCM, KM y FS.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

En este trabajo se desarrollo un sistema para el aprendizaje de sistemas difusos haciendo uso de los Procesos de Dirichlet y del algoritmo de Optimización por Colonia de Hormigas , se obtuvieron resultados satisfactorios en la creación de los conjuntos difusos para cada una de las variable de entrada así como los conjuntos de la variable de salida (clusters). Estos resultados fueron obtenidos por el módulo de DP. A pesar de que el ACO no arrojo resultados completamente satisfactorios, se logra una reducción en el número de reglas con resultados cualitativos mejores que con el número de reglas iniciales.

La gran cantidad de datos que se introducen en el sistema provocan ruido en el ACO, debido a ello es necesario re-plantear el algoritmo y buscar variaciones de este, que manejen gran cantidad de datos sin que esto produzcan alteración en los resultados. Nuestro trabajo presenta ventajas al manejar una gran cantidad de datos en comparación con trabajos relacionados ya que ellos utilizan una pequeña cantidad. Por ello emerge una pregunta ¿Qué cantidad de datos de entrenamiento afecta a los algoritmos de optimización como el ACO?.

La implementación del algoritmo minimiza el número de reglas y el error entre los datos de entrenamiento y los datos arrojados por el sistema difuso

De acuerdo a las gráficas de comportamiento del error y del número de reglas obtenidas por iteración, podemos concluir que algoritmo minimiza en gran cantidad estos dos valores. Los resultados cuantitativos entre clusters nos muestra la baja precisión del sistema difuso aprendido lo que nos hace concluir que la función del error cuadrático medio con la cual se evalúan las hormigas y con la que se determina la cantidad de feromonas que se dejan en el enlace visitado es la parte débil del algoritmo, por lo cual es necesaria una mayor

investigación sobre que métrica usar para obtener una mejor evaluación de las hormigas, dado que nos hace pensar lo siguiente:

- Supongamos que se tienen dos datos y dos hormigas con un conjunto de reglas similares pero que varían por una regla difusa. Supongamos que el error de la primer hormiga es similar entre los dos datos de entrada, digamos que es de 0.5, ahora supongamos que para la segunda hormiga esos errores varían de tal forma que solo uno de los datos queda muy bien aproximado (cercano al valor de entrenamiento), y el otro con un error de 0.7. Cuando se calcula el SME tendríamos que el error dado por la primer hormiga es $2 * (0.5^2) = 0.5$ y el error dado por la segunda sería $0.0 + 0.7^2 = 0.49$ esto nos daría que la mejor hormiga es la segunda, pero se tiene que la primer hormiga tiene un menor error para cada dato por individual por lo que se seleccionara la hormiga incorrecta.

Una de las grandes aportaciones en este trabajo es la implementación del algoritmo ACO en paralelo haciendo uso de CUDA en una de las tarjetas NVIDIA que hasta el momento no hemos encontrado trabajos relacionados. Esto nos ayudará a poder implementar otro tipos de algoritmos de optimización, los cuales sean altamente paralelizables de una manera más eficiente y fácil.

A pesar de ser un sistema de aprendizaje no supervisado es necesario darle una cantidad inicial de clusters para el módulo de DP. Dado que muchos algoritmos de clustering necesitan un número inicial de clusters, esto le quita algo de ventaja al algoritmo, pero al comparar los resultados arrojados con los resultados de algoritmos como el K-Means y el Fuzzy C-Means se observa una mejora que se sobrepone a la necesidad de un número de clusters inicial.

Para nuestro trabajo utilizamos un número de clusters relativamente grande ya que un número pequeño nos arrojaría un número erróneo de conjuntos difusos. debido a ello el algoritmo tarda más en converger.

5.2. Trabajo Futuro

1 Implementar las mejoras en la función de fitness utilizada así como el manejo de grandes cantidades de datos en el algoritmo del ACO. Pasar los Procesos de Dirichlet de Matlab a C/C++ y por último a CUDA para minimizar el tiempo de esa

parte del sistema. Nuestro sistema es un sistema de aprendizaje offline, por lo cual queremos explotar las capacidades de las tarjetas NVIDIA y la alta paralelización de los algoritmos para llegar lo mas cercano posible a un sistema de aprendizaje online.

II Ampliar los casos de estudio donde cada caso utilice más de una imagen del escenario para la segmentación en colores de este.

III El mejoramiento del algoritmo en: hacer al sistema robusto a los cambios de iluminación, análisis de los espacios de color existentes para obtener mejores resultados (como HSI, HSV, CIELab, CIELuv, etc) para hacer uso del que nos arroje mejores resultados y usar algoritmos de clustering basados en DP que no requieran un número inicial de clusters.

IV El uso de otras técnicas para mejorar el módulo de obtención de reglas difusas, como: PSO, los mismos DPs, AG, etc.

V La implementación del sistema para problemas de Control de robots, modelado de movimiento de avatares, toma de decisiones o para renderizado 3D donde los sistemas difusos han mostrado tener buenos resultados cuando son diseñados por un experto.

VI Uso de una medida de esparcidad en la función de fitness del algoritmo ACO, Ecuación 3.25 Capítulo 3, para dar una mayor esparcida a las reglas.

VII Nuestro sistema es un sistema de aprendizaje *offline*, pero se busca que en un futuro sea *online* con la ayuda de la paralelización de los algoritmos que intervienen en el sistema.

Bibliografía

[1] N. Bouguila, D. Ziou, and J. Vaillancourt. Unsupervised learning of a finite mixture model based on the dirichlet distribution and its application. *Image Processing, IEEE Transactions on*, 13(11):1533 –1543, nov. 2004.

[2] Sabri Boutemedjet, Nizar Bouguila, and Djemel Ziou. A hybrid feature extraction selection approach for high-dimensional non-gaussian data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1429–1443, 2009.

[3] James Bruce, Tucker Balch, and Manuela Veloso. Fast and cheap color image segmentation for interactive robots. In *Proceedings of IROS-2000*, pages 2061–2066. Addison Wesley Publishing, 2000.

[4] Jorge Casillas, Oscar Cordón, and Francisco Herrera. Learning fuzzy rules using ant colony optimization algorithms. In *Abstract proceedings of ANTS2000 From Ant Colonies to Arti Ants: A Series of International Workshops on Ant Algorithms*, pages 13–21, 2000.

[5] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Distributed optimization by ant colonies. In *European Conference on Artificial Life*, pages 134–142, 1991.

[6] Guo Dong and Ming Xie. Color clustering and learning for image segmentation based on neural networks. *Neural Networks, IEEE Transactions on*, 16(4):925 – 936, 2005.

[7] D. Dubois and H. Prade. Fuzzy logics and the generalized modus ponens revisited. *Cybernetics and Systems*, 15(15):293–331, 1984.

[8] T. S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230, 1973.

[9] E. B. Fowlkes and C. L. Mallows. A Method for Comparing Two Hierarchical Clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[10] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32 - 40, jan 1975.

[11] Tom Germano. Self organizing maps. Website, 1994. [Online; Accessed: 06/09/2012],`http://davis.wpi.edu/~matt/courses/soms/`.

[12] Timo Honkela, T. Kohonen.

[13] R. Huang, G. Yu, and Z. Wang. Dirichlet process mixture model for document clustering with feature partition. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1, 2012.

[14] Color image segmentation using histogram multithresholding and fusion. F. kurogullo et al.

[15] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[16] Chia-Feng Juang and Chi-Yen Wang. A self-generating fuzzy system with ant and particle swarm cooperative optimization. *Expert Syst. Appl.*, 36:5362–5370, April 2009.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[18] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982. 10.1007/BF00337288.

[19] Bart Kosko. Fuzzy Systems as Universal Approximators. *IEEE Trans. Comput.*, 43(11):1329–1333, 1994.

[20] Qiang Lu, Jack G. Conrad, Khalid Al-Kofahi, and William Keenan. Legal document clustering with built-in topic segmentation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 383–392, New York, NY, USA, 2011. ACM.

[21] E. H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. *Procedings of IEEE*, 121(12):1585–1588, 1974.

[22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[23] Majid Gholamiparvar Masooleh, Seyyed Ali, and Seyyed Moosavi. An improved fuzzy algorithm for image segmentation. *World Academy of Science, Engineering and Technology.*

[24] Berkeley University of California. The berkeley segmentation dataset and benchmark. Website. [Online; Accessed: 17/04/2012],`http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/`.

[25] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[26] C. J. Van Rijsbergen. *Information Retrieval.* Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[27] Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77 – 89, 1997.

[28] Y. W. Teh. A tutorial on dirichlet processes and hierarchical dirichlet processes. Gatsby Computational Neuroscience Unit University College London, mar 2007.

[29] Y. W. Teh. Dirichlet processes. In *Encyclopedia of Machine Learning.* Springer, 2010.

[30] M. Tkalcic and J.F. Tasic. Colour spaces: perceptual, historical and applicational background. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, volume 1, pages 304 – 308 vol.1, sept. 2003.

[31] J.H. van der Lee, W.Y. Svrcek, and B.R. Young. A tuning algorithm for model predictive controllers based on genetic algorithms and fuzzy decision making. *ISA Transactions*, 47(1):53 – 59, 2008.

[32] Hanzi Wang and David Suter. Color image segmentation using global information and local homogeneity.

[33] Inc. Wikipedia, The MathWorks. Ant colony optimization algorithms. Website. [Online; Accessed: 07/09/2012],`http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms`.

[34] K.A.D.N.K. Wimalawarne. Fast nonparametric image segmentation with dirichlet processes. In *Information and Automation for Sustainability, 2008. ICIAFS 2008. 4th International Conference on*, pages 336 –340, dec. 2008.

[35] Tianbing Xu, Zhongfei M. Zhang, Philip S. Yu, and Bo Long. Dirichlet Process Based Evolutionary Clustering. pages 648–657, December 2008.

[36] Lu Yi-su and Chen Wu-fan. Mr image segmentation by nonparametric model. In Yongsheng Ding, Yonghong Peng, Riyi Shi, Kuangrong Hao, and Lipo Wang, editors, *BMEI*, pages 390–394. IEEE, 2011.

[37] Guan Yu, Ruizhang Huang, and Zhaojun Wang. Document clustering via dirichlet process mixture model with feature selection. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 763–772, New York, NY, USA, 2010. ACM.

[38] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[39] A. Zare and P.D. Gader. Endmember detection using the dirichlet process. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008.

[40] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985. 10.1007/BF00940812.

Apéndice A

Glosario

Óptimo: En el contexto de la Economía y las finanzas públicas; es el mejor Valor que puede tomar una variable determinada en relación a un objetivo particular, considerando los medios que tiene para alcanzar ese objetivo. Concepto acuñado por Wilfredo Pareto para designar una situación en que se ha alcanzado la mejor asignación de recursos posible, cualquier cambio produciría, en consecuencia, una posición inferior.

***Evolutionary Clustering*:** Es el problema de procesar datos con estampas/etiquetas de tiempo para producir una secuencia de *clusterings*; esto es, un *clustering* por cada paso de tiempo del sistema. Cada *clustering* en la secuencia debe ser similar al *clustering* del paso de tiempo anterior/previo, y debe reflejar exactamente el arribo de datos en ese paso de tiempo.

***Latent Dirichlet Allocation* (LDA):** Es un modelo generativo el cual permite que conjuntos de observaciones puedan ser explicados por grupos no-observados que expliquen por que algunas partes de los datos son similares. Por ejemplo, si las observaciones son palabras en documentos, es posible que cada documento sea una mezcla de un pequeño número de tópicos y que cada aparición de cada palabra se pueda atribuir a uno de los temas del documento.

Self-organizing maps [18] [12] [11] (SOMs) es una técnica de visualización de datos inventada por el profesor Teuvo Kohonen, la cual reduce las dimensiones de los datos a través del uso de redes neuronales auto organizadas (*self-organizing neural networks*). El problema que intenta resolver la visualización de datos es que los humanos simplemente no pueden visualizar datos de grandes dimensiones así que esta técnica fue creada para ayudarnos a entender estos datos de grandes dimensiones. La manera en que trabajan los

SOMs sobre la reducción de dimensiones es mediante la producción de un mapa, usualmente de una o dos dimensiones, que gráfica la similaridades de los datos mediante la agrupación de puntos de datos similares juntos.

Espacio métrico: En matemática, es un conjunto junto con una función distancia (porque cumple con unas propiedades concretas atribuidas a las distancias) definida sobre él, de modo que cualquier par de puntos (o elementos) del conjunto están a una cierta distancia asignada por dicha función. Formalmente, es un conjunto $\mathbb{M}$ (a cuyos elementos se le denominan puntos) con una función asociada (también llamada una métrica) $d : \mathbb{M} \times \mathbb{M} \rightarrow \Re$.

Apéndice B

Anexo A

Funciones requeridas por el algoritmo del ACO:

Función que nos da la probabilidad de movernos del vertice r al vertice s:

1. **aggTransProb_rs(r,s)**
2. $i \leftarrow 0$
3. $sum \leftarrow 0$
4. For $j \leftarrow 0$ to $\#(V)$ do
5. $sum \leftarrow sum + pow(\tau(r,s),\alpha)pow(\eta(r,j),\beta)$
6. return $pow(\tau(r,s),\alpha)pow(\eta(r,s),\beta)/sum$

Función de costo del borde dado por los vertices r y s:

1. **aggDeltaTau_ith_rs(r,s,k,MODEL_TYPE)**
2. switch($MODEL\_TYPE$)
3. case $ANT\_QUANTITY$:
4. return $Q_1/aggEdgeCost(r,s)$
5. case $ANT\_DENSITY$:
6. return Q_2
7. case $ANT\_QUANTITY$:
8. return -1.0

La función $aggEdgeCost()$ que aparece en el función de costo depende del problema y nos indica que peso hay que darle a cada uno de los bordes que conectan dos vértices en el grafo.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Uso de los Procesos de Dirichlet y Optimización por Colonia de Hormigas para el Aprendizaje de Sistemas Difusos

del (la) C.

Arturo GARCÍA GARCÍA

el día 18 de Enero de 2013.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3C
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González
Pico
Investigador CINVESTAV 2C
CINVESTAV Unidad Guadalajara

Dr. Andrés Méndez Vázquez
Investigador CINVESTAV Guadalajara
2C
CINVESTAV Guadalajara