TESIS DEFENDIDA POR

Eva Romero Vásquez

Y aprobada por el siguiente comité:

Dr. Gustavo Olague Caballero

Director del Comité

M.C. José Luis Briseño Cervantes

Miembro del Comité

Dr. José Antonio García Macías

Miembro del Comité

Dr. Hugo Homero Hidalgo Silva

Miembro del Comité

Dr. Eugenio Rafael Méndez Méndez

Miembro del Comité

Dr. Pedro Gilberto López Mariscal

Coordinador del Programa de posgrado en Ciencias de la Computación

Dr. Raúl Ramón Castro Escamilla

Director de Estudios de Posgrado

25 de Septiembre de 2006

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN
SUPERIOR DE ENSENADA

POSGRADO EN CIENCIAS
EN CIENCIAS DE LA COMPUTACIÓN

Reconocimiento de clases de objetos basado en texturas mediante algoritmos genéticos

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN CIENCIAS

Presenta:

Eva Romero Vásquez

Ensenada, Baja California, México. Septiembre de 2006.

RESUMEN de la tesis de **Eva Romero Vásquez**, presentada como requisito parcial para obtener el grado de MAESTRO EN CIENCIAS en CIENCIAS DE LA COMPUTACIÓN. Ensenada, Baja California. Septiembre de 2006.

Reconocimiento de clases de objetos basado en texturas mediante algoritmos genéticos

Resumen aprobado por:

Dr. Gustavo Olague Caballero

Director de Tesis

La Computación Genética y Evolutiva es un campo de investigación reciente en ciencias computacionales, esta trabaja con sistemas adaptativos y técnicas de optimización inspiradas en las reglas de la evolución natural. Una de sus metas es dotar a las computadoras con la capacidad de procesamiento de la información comparable a la que se encuentra en la naturaleza. Por otro lado, la textura es una de las primitivas cruciales en visión humana y sus características han sido usadas ampliamente en la literatura para identificar el contenido de imágenes digitales.

Este trabajo se concentra en resolver dos problemas: el reconocimiento de expresiones faciales y el reconocimiento de objetos, los cuales se pueden englobar como reconocimiento de clases de objetos. Para resolverlo se realiza la búsqueda de las mejores regiones en un conjunto de imágenes digitales para el reconocimiento de clases de objetos haciendo uso de las características de textura de dichas regiones. La selección de las regiones así como las características de textura de éstas se lleva a cabo por medio de la estrategia de algoritmos genéticos. La clasificación de los objetos dentro de las regiones encontradas se realiza mediante la máquina de soporte vectorial. Los resultados obtenidos después de la aplicación de un cuestionario a un grupo de personas así como la aplicación del método propuesto sobre dos conjuntos de imágenes confirman la validez y desempeño de la solución. Los resultados experimentales muestran resultados comparables a los obtenidos por humanos y los vistos en la literatura.

Palabras clave: Análisis de textura, matriz de coocurrencia, algoritmos genéticos, máquinas de soporte vectorial.

ABSTRACT of the thesis presented by **Eva Romero Vásquez**, as a partial requirement to obtain the MASTER SCIENCE degree in COMPUTER SCIENCES. Ensenada, Baja California. September 2006.

Class object recognition based on textures by genetic algorithms

Abstract approved by:

Dr. Gustavo Olague Caballero

Thesis director

Genetic and Evolutionary Computation is a recent research field in computer science which deals with adaptative systems and optimization techniques inspired by the rules of natural evolution. One of its goals is to endow computers with information-processing capabilities comparable to those found in nature. On the other hand, texture is one of the crucial primitives in computer vision and its characteristics have been used to identify image information content.

This work is concentrated in solving two problems: the face expression recognition and the object recognition, which can be included like class object recognition. In order to solve it, we search the best regions in a set of digital images in order to improve the class object recognition using the characteristics of these texture regions. The selection of the regions, as well as the characteristics of texture is performed using the Genetic Algorithms strategy. Moreover, the object classification with all found regions is performed using Support Vector Machine. The results obtained after aplying a questionnaire to a group of people, as well as the application of the proposed method on two set of images confirm the validity and performance of the solution. The experimental results show comparable results obtained by humans and those presented in the literature.

Keywords: Texture analysis, coocurrence matriz, genetic algorithms, support vector machine.

Dedicado a mis padres

Eva Vásquez Aguilar
y
José Romero Quintero

Agradecimientos

Agradezco a mi familia por el apoyo que me ha dado.

Al Dr. Gustavo Olague, por su colaboración y por compartir sus conocimientos a lo largo de la realización de este trabajo.

A los miembros del Comité de Tesis, por su interés y aportaciones a este trabajo.

Al Centro de Investigación Científica y de Educación Superior de Ensenada y al Consejo Nacional de Ciencia y Tecnología, por darme las facilidades para estudiar en esta institución.

A León, Carlos, Victor, Marcos, Eddie, Juan, Alina y Cesar por su amistad y apoyo.

A todas las personas que han sido parte del proceso de llegar a ser lo que soy ahora.

Ensenada, México
25 de Septiembre de 2006.

Eva Romero Vásquez

Tabla de Contenido

Capítulo Página

Tabla de Contenido (Continuación)

Lista de Figuras

Lista de Figuras (Continuación)

Lista de Figuras (Continuación)

Lista de Tablas

Capítulo I

Introducción

La inteligencia artificial (AI, Artificial Intelligence) es el campo de la investigación científica que intenta imitar la inteligencia humana a través de la creación de herramientas (término regularmente relacionado con las computadoras). Dentro de la inteligencia artificial existen diferentes subcampos como son: visión por computadora, redes neuronales artificiales, computación evolutiva, sistemas expertos, entre otros; siendo la visión por computadora el principal punto de referencia en este trabajo (Wikipedia, 2004).

La *visión por computadora* no sólo se dedica a la extracción de la información del mundo que se encuentra en una imagen, ésta involucra un proceso mucho más complejo. Se puede considerar a la visión por computadora como *un conjunto de técnicas y modelos que permiten el procesamiento, el análisis y la interpretación de la información considerada relevante la cual se obtiene a partir de imágenes digitales* (INAOE, 2004).

La investigación realizada en este subcampo ha influido grandemente al constante estudio del funcionamiento del sistema visual humano pues de manera análoga se podrá cumplir el principal propósito de la visión por computadora: *"crear una herramienta capaz de detectar, ubicar y entender los objetos que se encuentran en el mundo real de la manera más precisa posible"* (INAOE, 2004).

El título de esta tesis, ***reconocimiento de clases de objetos basado en texturas mediante algoritmos genéticos***, puede ser utilizado como introducción por sí mismo. Por medio de éste, y a través de las definiciones antes mencionadas, se puede inferir que este trabajo pertenece al subcampo de visión por computadora y a su vez,

de manera más general, al campo de la inteligencia artificial.

Existen diferentes problemas relacionados a este campo, concerniente a robots, sistemas de vigilancia, bases de datos multimedia y otros. De cualquier modo, los problemas de visión por computadora están divididos en diferentes categorías basados no sólo en el campo de las aplicaciones en el mundo real, sino también en el objetivo científico del problema.

Así, algunas categorías en problemas de visión por computadora son la detección de rostros, extracción del fondo, reconstrucción de escenas 3D y otras. La mayoría de estos problemas generalmente pueden ser catalogados como problemas de *reconocimieto de objetos*, no importa si tratamos de identificar un rostro, extraer la información de primer plano de la del fondo o seguir el movimiento de objetos en una secuencia de video; estamos tratando de reconocer algún tipo de objeto.

I.1 Problema del reconocimiento de objetos

La mayoría de los problemas de visión por computadora pueden describirse como problemas de reconocimiento. A partir de aquí, estos problemas se dividen en 2 categorías: reconocimiento de un objeto *específico* o reconocimiento de una *clase* de objeto.

Imaginemos el problema de detectar nuestro carro. El objeto que estamos buscando obviamente es un carro. La tarea no es buscar cualquier carro. Lo que buscamos es un objeto específico que tiene atributos únicos; en este caso los que corresponden al carro que nos pertenece.

Por otro lado, el problema de reconocer una clase de objeto es más general. Un ejemplo podría ser el problema de detectar carros, como una clase de objeto. En este caso, se utiliza la información que se tiene de los datos de entrenamiento para crear un modelo interno de la idea de un carro, y entonces se trata de detectar objetos y clasificarlos como carros o no-carros.

De manera formal, el *problema de reconocimiento de objetos* (del inglés object recognition) es el siguiente:

Definición 1. Problema del reconocimiento de objetos. *Dada una imagen I, una base de datos de k objetos y una representación R_j para cada objeto j en la base de datos; el reconocimiento de objetos se puede expresar como:*

$$Q = arg\,min\, c(R_j, I) \qquad j \epsilon 1, .., k \quad , \tag{1}$$

donde $c(R_j, I)$ es una función la cual proporciona la compatibilidad o la consistencia de representar el objeto j con la imagen I.

Una definición más sencilla propuesta por Russell y Norving (1995) se expresa como sigue:

1. *Dada una escena que contiene uno o más objetos elegidos de una colección de objetos $O_1, O_2, ..., O_n$ conocidos a priori, y*
2. *Dada una imagen de la escena tomada desde una posición y orientación desconocida;*

determinar lo siguiente:

1. *¿Cuáles objetos $O_1, O_2, ..., O_n$ están presentes en la escena?*
2. *De tal suerte que, para cada objeto, se pueda determinar la posición y orientación relativa al observador.*

I.2 Métodos para el reconocimiento de objetos

Como se ha mencionado, el problema de reconocimiento de objetos tiene un amplio rango de aplicaciones. Además, diferentes investigadores lo han enfocado desde distintos ángulos, dando como resultado un campo en el cual su revisión detallada es prácticamente imposible. De cualquier manera, se puede dar una revisión rápida de la investigación actual. Los diferentes métodos de investigación caen dentro de cuatro subcategorías principales: los métodos que utilizan propiedades invariantes y características espaciales, los que utilizan descriptores estructurales, los métodos de alineamiento y los basados en el análisis de textura.

La primer subcategoría se basa en la suposición de que los objetos tienen ciertas propiedades invariantes y características espaciales que son comunes entre ellos desde todos sus ángulos de vista. Usualmente, estas características se calculan utilizando una función aplicada a un conjunto de vistas de objetos obteniendo así un valor real. Esta función debe ser fácil de calcular, ya que será utilizada muchas veces. Un trabajo que se encuentra dentro de esta subcategoría es el realizado por Trujillo y Olague (2006) el cual presenta un enfoque novedoso que utiliza el aprendizaje para construir una función de extracción de características de bajo nivel para aplicaciones de visión por computadora. El tipo de características de bajo nivel que se extraen son puntos de interés en imágenes. El método se enfoca en la detección de puntos de interés como un problema de optimización, y desarrolla una metodología de aprendizaje que permite generar automáticamente detectores de puntos de interés. La detección de puntos de interés es adecuada en la presencia de tres transformaciones: traslación, rotación y cambios de iluminación.

La segunda subcategoría es la descomposición estructural. La suposición que se hace es que todos los objetos se crean a partir de cierto conjunto de componentes genéricos. Estos componentes son llamados genéricos pues se supone que todos los objetos se pueden describir como una combinación de dichos componentes. Un ejemplo de esto, es que todas las letras se pueden describir como combinaciones de líneas y curvas. La letra L es un objeto formado por la combinación de dos líneas que se tocan formando un ángulo recto. Este método tiene un aspecto intuitivo, ya que describe un objeto utilizando componentes genéricos familiares para las personas. El problema es que aún cuando empatan correctamente estos componentes genéricos, su combinación puede llegar a ser muy compleja y difícil de expresar. El número de combinaciones posibles incrementa exponencialmente la cantidad de componentes genéricos. Mientras se utilicen más componentes genéricos, a fin de tener una mayor precisión, será más

difícil expresar sus combinaciones.

La tercer subcategoría es el método de alineamiento. La idea básica es comparar los objetos y los modelos correspondientes almacenados en memoria a fin de estimar las transformaciones para separarlos o identificarlos. Este método es mucho más costoso computacionalmente ya que, usualmente, los modelos son representaciones internas en 3D de los objetos. Esto es esencial a fin de detectar los objetos bajo tranformaciones no sólo en escala y condiciones de iluminación, sino también considerando diferentes ángulos de vista. El resultado es un programa que es capaz de identificar los objetos y de estimar la posición actual y ubicación 3D de los objetos.

La última subcategoría es la basada en el análisis de textura. La textura es una de las características más importantes que permiten definir los objetos en una imagen. Se caracteriza por una distribución espacial de niveles de gris en un vecindario. Con el propósito de capturar la dependencia espacial de valores de niveles de gris, se toma en cuenta la matriz de análisis de textura. Dicha matriz muestra la relación existente entre un pixel y sus vecinos, los cuales se encuentran separados a cierta distancia y en cierta orientación. Existen distintas características de textura que se pueden obtener a partir de la información contenida en la matriz de análisis de textura. Estas caracterizan las propiedades de textura de los objetos en las imágenes, tal como el contraste, rugosidad, orientación, etc.

I.3 Clasificación

El problema de clasificación es mucho más general que el concerniente a la visión por computadora por no decir al reconocimiento de objetos. La idea básica en clasificación es reconocer objetos basados en características. Las características de un objeto se pueden representar como un punto en el espacio de características N-dimensional definidas en función de la tarea de reconocimiento de objetos dada. Por otro lado, los

métodos de reconocimiento de patrones (del inglés pattern recognition) caen dentro de esta categoría y su potencial se ha demostrado en muchas aplicaciones. A continuación se definen los principales métodos de clasificación usados más frecuentemente en la literatura (Russell y Norving, 2003; Jain *et al.*, 2000).

I.3.1 Máquina de soporte vectorial

El método de máquina de soporte vectorial (SVM, support vector machine) consta de dos operaciones matemáticas: la transformación no lineal de un vector de entrada a uno en el espacio de características de mayor dimensión y la construcción de un hiperplano óptimo que separe los vectores de características transformados, ver Figura 1. En la primera operación, los vectores de características originales se mapean a un espacio de mayor dimensión donde las dos clases pueden ser linealmente separables por medio de un kernel. Los kérneles más utilizados son el lineal y la función en base radial (RBF, radial basis function). En la segunda operación, se construye un hiperplano separador óptimo para maximizar el margen de separación entre muestras positivas y negativas.

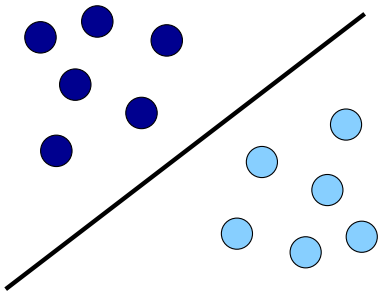


Figura 1: *Ejemplo de un hiperplano que divide a dos clases.*

I.3.2 Vecinos más cercanos

El método de vecinos más cercanos (del inglés nearest neighbor) supone que para cada clase de objeto existe un modelo conocido (con valores de características ideales) y se representa a través de la clase i como $f_{i,j},\ j = 1, ..., N$. Ahora, supongamos que se detectan y se miden las características del objeto desconocido U y se representa como $u_j,\ j = 1, ..., N$. Para características espaciales en dos dimensiones, la situación se muestra en la Figura 2.

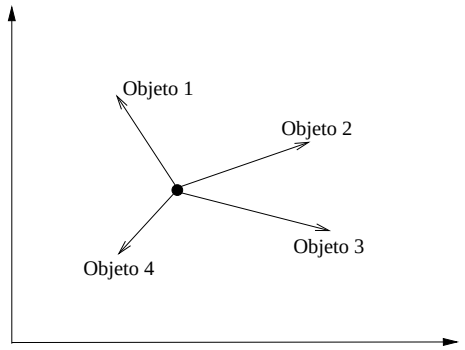


Figura 2: *Los prototipos de cada clase se representan como puntos en el espacio de características. Un objeto desconocido se asigna a la clase más cercana utilizando una medida de distancia en este espacio.*

Para decidir la clase del objeto, se mide la similitud con cada clase calculando la distancia de los puntos, representando cada clase en el espacio de características, y se asigna a la clase más cercana. La distancia puede ser la Euclidiana o cualquier combinación ponderada de características. En general, se calcula la distancia d_j del objeto desconocido de la clase j a través de la siguiente expresión

$$d_j = \left[\sum_{i=1}^{N} (u_j - f_{i,j})^2 \right]^{\frac{1}{2}} . \tag{2}$$

De esta manera, el objeto es asignado a la clase R utilizando la distancia mínima:

$$d_R = min_{j=1}^{N} [d_j] \ . \tag{3}$$

En resumen, la pertenencia de un objeto a una clase se calcula considerando la distancia mínima al punto de la característica, de manera que quede asignado a un prototipo del objeto. En la práctica, puede resultar difícil encontrar un prototipo del objeto. Sabiendo que muchos objetos pertenecen a una clase, se pueden considerar valores característicos para todos los objetos conocidos de una clase. Esta situación se muestra en la Figura 3, donde cada clase se representa como un conjunto de puntos en el espacio de características. Se puede tomar como prototipo para la clasificación el centroide del conjunto representativo de la clase o el punto más cercano a cada clase. Los dos enfoques más comunes en esta situación son:

1. Considerar el centroide del conjunto como punto característico del prototipo del objeto y calcular la distancia a este.

2. Considerar la distancia al punto más cercano de cada clase.

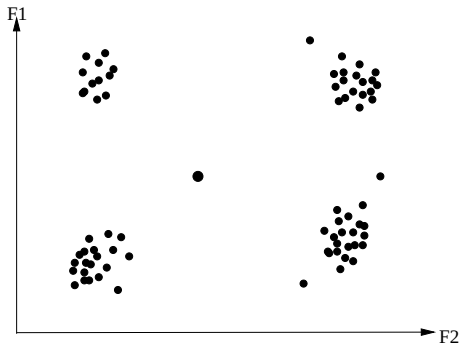


Figura 3: *Todos los objetos conocidos de cada clase se representan como puntos en el espacio de características.*

I.3.3 Método Bayesiano

El método Bayesiano se utiliza para el reconocimiento de objetos cuando la distribución de los objetos no es tan evidente como en el caso mencionado anteriormente. En general, existe una superposición significativa en los valores característicos de los diferentes objetos. Así, como se muestra en el espacio de características de una dimensión en la Figura 4, donde diferentes objetos pueden tener el mismo valor característico.

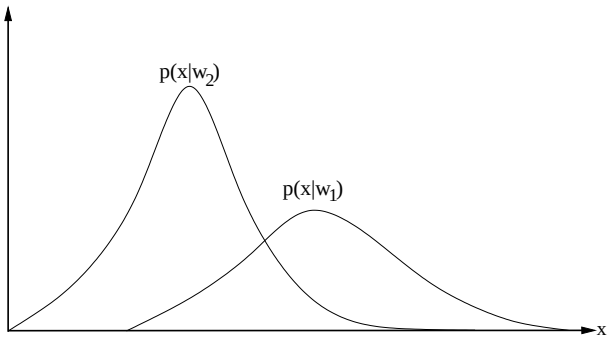


Figura 4: *La función de densidad condicional para $p(x|w_j)$. Esta gráfica muestra la probabilidad de los valores característicos de dos clases que siguen una distribución Gaussiana.*

En el método Bayesiano, se utiliza el conocimiento probabilístico acerca de las características de los objetos y la frecuencia de dichos objetos. Supóngase que se conoce $p(w_j)$ la probabilidad de los objetos de la clase j. Esto significa que conocemos $P(w_j)$ la probabilidad de que aparezca un objeto de la clase j, y por lo tanto en ausencia de cualquier otro conocimiento se puede minimizar la probabilidad de error relacionado a asignar un objeto desconocido a la clase para la cual $p(w_j)$ es máximo.

Las decisiones acerca de la clase de un objeto usualmente están basadas en observaciones características. Supóngase que se conoce la probabilidad $p(x|w_j)$ como se

muestra en la Figura 4. La probabilidad condicional nos dice que, basada en la información probabilística provista, sabemos que si un valor característico x es observado, entonces la probabilidad de que el objeto pertenezca a la clase j es $p(x|w_j)$. Basados en este conocimiento, se puede calcular la probabilidad *a posteriori* $p(w_j|x)$ del objeto. La probabilidad *a posteriori* es la probabilidad de que, para la información y observaciones dadas, el objeto desconocido pertenezca a la clase j. Utilizando la regla de Bayes, la probabilidad está dada por:

$$p(w_j|x) = \frac{p(x|w_j)p(w_j)}{p(x)} \ , \tag{4}$$

donde

$$p(x) = \sum_{j=1}^{N} p(x|w_j)p(w_j) \ . \tag{5}$$

El objeto desconocido se puede asignar a la clase con más alta probabilidad *a posteriori* $p(w_j|x)$. Como se puede ver en las Ecuaciones (4) y (5), y como lo muestra la Figura 5, una probabilidad *a posteriori* depende del conocimiento *a priori* acerca de los objetos. Si la probabilidad *a priori* de los objetos cambia, también el resultado.

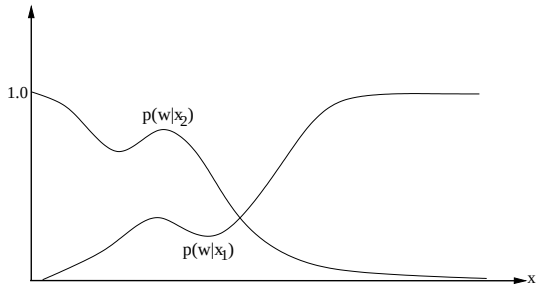


Figura 5: *Las probabilidades a posteriori para dos valores diferentes de los objetos considerando sus probabilidades a priori.*

I.3.4 Redes neuronales

Las redes neuronales (del inglés neural networks) han sido propuestas para resolver el problema de reconocimiento de objetos. Estas implementan un método de clasificación. Su atractivo se encuentra en su habilidad para dividir el espacio de características utilizando fronteras no lineales para las clases. Dichas fronteras se obtienen a través del entrenamiento de la red. Durante la fase de entrenamiento, se presentan muchas instancias de los objetos para que sean reconocidas. Si se selecciona cuidadosamente el conjunto de entrenamiento para representar todos los objetos encontrados durante la fase de reconocimiento, entonces la red puede aprender las fronteras de clasificación en su espacio de características. Durante la fase de reconocimiento, la red trabaja como cualquier otro clasificador.

La característica más atractiva de las redes neuronales es su habilidad para utilizar fronteras de clasificación no lineales y habilidades de aprendizaje. La limitación más importante es su inhabilidad para introducir hechos conocidos acerca del dominio de la aplicación y la dificultad de corregir sus errores de desempeño.

I.4 Motivación

Se sabe que la textura es una de las primitivas cruciales en visión humana y sus características (contraste, homogeneidad, entropía, etc.) han sido usadas ampliamente en la literatura para identificar el contenido de imágenes digitales (Malick y Perona, 1990; Materka y Strzelecki, 1998; Tuceryan y Jain, 1998).

Lo que se desea hacer en este trabajo es implementar un sistema que realice la búsqueda de las mejores regiones dentro de imágenes digitales para el reconocimiento de clases de objetos, como por ejemplo expresiones faciales[1] y ciertos objetos (carros,

[1]Cada expresión facial puede ser vista como un objeto.

casas, etc.), haciendo uso de las características de textura de dichas regiones. Que, como se mencionó anteriormente, son muy importantes en la visión humana.

I.5 Objetivo

Puesto que el propósito del reconocimiento de objetos es encontrar los modelos de las clases que sean lo suficientemente *invariantes* de tal suerte que se puedan incorporar variaciones intraclases y suficientemente *discriminativas* a fin de distinguir entre clases, el objetivo general de la tesis es:

Implementar un sistema que realice el reconocimiento de objetos mediante la búsqueda automática de modelos de clases. Dicho sistema deberá extraer información relevante de textura de los objetos en las imágenes digitales mediante el uso de algoritmos genéticos.

Utilizando la terminología correspondiente, lo que se desea es encontrar los vectores de soporte que presenten el número de descriptores suficientes para cada región en el espacio de la imagen, de tal manera que estos maximicen la función de evaluación a fin de obtener un porcentaje tal que permita una mejor clasificación.

I.6 Contribuciones

En comparación con el trabajo publicado en este campo, las principales contribuciones de esta tesis son:

- La implementación de un sistema novedoso con el uso de un conjunto de métodos ya existentes (algoritmos genéticos, descriptores de textura, máquina de soporte vectorial) que, en su conjunto, no habían sido utilizados anteriormente.

- La obtención de las características (descriptores) relevantes en el reconocimiento de objetos.

- La mayoría de los métodos de reconocimiento publicados trabajan bien con imágenes seleccionadas cuidadosamente para sus experimentos. En este trabajo, se evalúa al algoritmo implementado utilizando un conjunto de datos:

  - con variaciones significativas dentro del mismo conjunto de objetos.

  - con presencia de oclusión parcial y variaciones de apariencia debido a cambios de ángulo de vista, iluminación y escala.

- La ampliación de la base de datos existente CALTECH (2005) con imágenes obtenidas de la web.

I.7 Organización de la tesis

La tesis está organizada de la siguiente manera. En el capítulo II se presenta el marco teórico con los temas, conceptos y definiciones para la comprensión de este trabajo. En el capítulo III se formalizan la combinación de los métodos utilizados. En el capítulo IV se describe el software y hardware utilizados para la implementación del sistema, se explica detalladamente el conjunto de datos tanto de imágenes digitales como cuestionarios y se muestran los experimentos realizados y los resultados obtenidos de los cuales se hace un análisis. En el capítulo V se muestran las conclusiones y se enuncian los trabajos que pueden realizarse para dar continuidad al presente. Finalmente, se anexa un apéndice el cual muestra una lista de conferencias relacionadas a la visión por computadora y a los algoritmos de evolución.

Capítulo II

Marco teórico

II.1 Imágenes digitales

Una imagen digital es una representación en dos dimensiones (2D) de una escena tridimensional, la cual se representa como un conjunto finito de valores digitales, llamados pixeles (pixel, picture element).

II.1.1 Tipos de imágenes

Cada pixel de una imagen se encuentra asociado a una posición específica en alguna región 2D. Dicho pixel tiene un valor directamente proporcional a la cantidad de luz captada por el sensor correspondiente. Una forma muy importante de clasificar las imágenes es según la naturaleza de los sensores, y la cantidad y tipo de información que se asigne a cada pixel.

- **Binarias** (1 bit por pixel): Tienen solamente dos valores posibles para cada pixel (0 ó 1).
- **Escala de grises** (8 bits por pixel): El valor de cada pixel es una muestra de una serie de posibles valores en el rango [0-255], representando la intensidad del nivel de gris en la imagen.
- **Color**: Contienen información de color. Estas se dividen en varios tipos, aunque las más utilizadas son las siguientes:

  - *RGB* (24 bits por pixel): Se basan en tres canales que representan los elementos básicos para formar el color (rojo, verde y azul), así se requieren tres

muestras por cada pixel en la imagen con una serie de posibles valores en el rango [0-255] para cada una de las muestras.

- *CMYK* (32 bits por pixel): Se requieren cuatro muestras por cada pixel en la imagen con una serie de posibles valores en el rango [0-255] para cada una de las muestras (cyan, magenta, amarillo y negro).

- **Falso color**: Representa un objeto en color que difiere de la percepción humana de dicho objeto. Se deriva de una imagen en escala de grises mapeando cada valor de pixel a un color de acuerdo a una tabla o función.

- **Multiespectrales**: Son imágenes del mismo objeto tomadas en diferentes bandas, visible o infrarrojo, del espectro electromagnético. Este tipo de imágenes se pueden adquirir por medio de radiómetros. Los radiómetros detectan, en paralelo, la radiación de los objetos a capturar en un número de bandas de longitudes de onda. Cada uno captura una imagen digital en el espectro visible y la convierten en longitudes de onda infrarroja, clasificadas como infrarrojo cercano (NIR, Near InfraRed), infrarrojo medio (MIR, Middle InfraRed) e infrarrojo lejano o térmico (FIR, Far InfraRed).

II.2 Reconocimiento de objetos

II.2.1 Definiciones

Como se estableció en la introducción, el rconocimiento de objetos se puede definir formalmente de la siguiente manera:

Definición 2. Reconocimiento de objetos. *El reconocimiento de objetos es la tarea de determinar si los objetos de una o más clases (y/o cuáles) aparecen en una colección/secuencia de imágenes.*

De manera formal, como ya se había mencionado anteriormente, el problema de reconocimiento de objetos (object recognition) es el siguiente:

Dada una imagen I, una base de datos de k objetos y una representación R_j para cada objeto j en la base de datos; el reconocimiento de objetos se puede expresar como:

$$Q = arg\,min\, c(R_j, I) \qquad j \epsilon 1, .., k \quad ,$$

donde $c(R_j, I)$ es una función la cual proporciona la compatibilidad o la consistencia de representar el objeto j con la imagen I.

Una definición más sencilla propuesta por Russell y Norving (1995) se expresa como sigue:

1. *Dada una escena que contiene uno o más objetos elegidos de una colección de objetos $O_1, O_2, ..., O_n$ conocidos a priori, y*

2. *Dada una imagen de la escena tomada desde una posición y orientación desconocida;*

determinar lo siguiente:

1. *¿Cuáles objetos $O_1, O_2, ..., O_n$ están presentes en la escena?*

2. *De tal suerte que, para cada objeto, se pueda determinar la posición y orientación relativa al observador.*

Una *clase de objetos* es un conjunto de objetos cuyos miembros comparten ciertas características. De aquí en adelante, el término reconocimiento de objetos será utilizado para denotar la búsqueda de un objeto específico así como la búsqueda de objetos de una clase general en imágenes. La suposición implícita es que la variación en apariencia

de los miembros de una clase y un simple objeto se pueden describir de la misma forma. Un simple objeto se puede ver como miembro de una clase de objetos. Por esto, los términos objeto y clase de objetos serán utilizados indistintamente y se diferenciará entre ellos sólo cuando sea necesario.

La tarea del reconocimiento

La noción de las definiciones de imagen digital, región de imagen y apariencia de un objeto son fundamentales para entender la definición de reconocimiento. Una *imagen digital* es el mapeo $P : S \rightarrow D$ donde D es el conjunto de medidas de intensidad posibles y S es el subconjunto finito de Z^2 llamado dominio espacial el cual, para una imagen I, se denota como S_I. Una *región de imagen* R_I de la imagen I se define como un subconjunto del dominio espacial de la imagen ($R \subseteq S_I$). La región de una imagen, la cual es una proyección del objeto O en la imagen I se denota como R_O^I. En general, la proyección del objeto solo cubrirá parte de la imagen entera observada ($R_O^I \subseteq S_I$). El objeto cuya proyección corresponde a la imagen entera ($R_O^I = S_I$) es entonces la *escena* completa limitada sólo por el campo de visión. La *apariencia* A_O^I del objeto O es el par (R_O^I, P_O^I) donde $P_O^I : R_O^I \rightarrow D$. El mapeo P_O^I es una función de las propiedades del objeto (por ejemplo, su forma y reflejo de la superficie) y los parámetros que influyen en la formación de la imagen tales como ángulos de vista, iluminación y sensibilidad de la cámara.

Ahora, supongamos que el conjunto finito C de K clases de objetos $C = C_1, C_2, ..., C_k$. Cada C_j, $j = 1, .., k$ es un conjunto de objetos l_j donde dichos objetos pueden ser infinitos, entonces $C_j = O_j^1, O_j^2, ..., O_j^{l_j}$. Una definición más formal de la tarea de reconocimiento es como sigue:

Definición 3. Tarea general del reconocimiento de objetos. *Dado C y una imagen I, la tarea general de reconocimiento es determinar si uno o más objetos de una clase*

$C_j \in C$ aparecen en la imagen. Esto es, si $R^I_{O^i_j} \neq 0$, para toda i, j donde $i = 1, ..., l_j$.

En algunas aplicaciones, además del reconocimiento, se requiere de la *ubicación de objetos*.

Definición 4. Ubicación de objetos. *La ubicación de objetos es la tarea de establecer la región $R^I_{O^i_j}$, la cual es la proyección del objeto O^i_j en la imagen I.*

Cuando C consta de una sola clase, a la tarea de reconocimiento se le llama *detección de objetos*. El caso especial de una sola clase se define a continuación:

Definición 5. Detección de objetos. *La detección de objetos es la tarea de determinar si los objetos de la clase C están presentes en la imagen. Esto es, si $R^I_{O_j} \neq 0$, donde $j = 1, ..., l_C$ es el número de objetos en C.*

Se pueden distinguir dos casos especiales en la tarea de reconocimiento. En algunas aplicaciones, se sabe que *exactamente* un objeto (*clase*) puede estar presente en la imagen. En este caso, el reconocimiento es un mapeo del conjunto de todas las imágenes $I \rightarrow J$, donde $J = 1, ..., k$ representa el conjunto de índices de las clases de objetos definidos. Formalmente, el objetivo es determinar la clase C_j para la cual $R_{O^i_j} \neq 0$ donde $i = 1, ..., l_j$. En la literatura, esta tarea se llama *identificación de objetos*.

En otras aplicaciones, se requiere identificar *a lo más* un objeto (clase) presente en una imagen. La tarea se define exactamente como la anterior, con la suma de una clase $\bar{C}$ de todos los objetos diferentes a los definidos. En este caso, el reconocimiento es el mapeo $I \rightarrow k + 1$ tomando en cuenta la categoría extra.

II.2.2 Proceso para el reconocimiento de objetos

El proceso general que se sigue para realizar el reconocimiento de objetos en una imagen, consta de seis etapas (aunque pueden ser menos):

1. *Se adquieren las imágenes* utilizando un sensor de imágenes (por ejemplo, una cámara fotográfica) y se digitaliza la señal para producir una imagen digital.

2. *Preprocesar* la imagen digitalizada con el propósito de mejorarla y eliminar el "ruido" producido en la misma y prepararla para los procesos posteriores.

3. *Segmentar* la imagen, proceso consistente en dividirla en sus partes constituyentes u objetos.

4. *Representar* los objetos a través de alguna estructura de datos.

5. Extraer los rasgos característicos de los segmentos que sean de interés, es decir, se debe hacer una *descripción* o *selección de rasgos* para establecer clases de objetos.

6. *Reconocer* e *interpretar* los resultados al asignar una etiqueta a un objeto basándose en la información proporcionada por sus descriptores y a fin de atribuir un significado al conjunto de objetos reconocidos.

Para poder realizar este proceso, es necesario contar con una *base de conocimiento* en la que exista un conocimiento del dominio del problema con la cual interactuar. Este conocimiento *a priori* guiará y controlará la operación de cada etapa para llegar a reconocer objetos exitosamente.

En todo sistema de visión se trabaja con ciertas problemáticas inherentes a cada proceso:

1. En la etapa del preprocesado, puede llegarse a eliminar parte de la imagen al confundirse con "ruido" e incorporar elementos que no existen.

2. El proceso de segmentación puede verse afectado por las condiciones existentes durante la captación de la imagen. Por ejemplo, la iluminación puede llegar a modificar la intensidad de los niveles de gris y afectar la división de la imagen

en regiones, lo que determinará en gran medida el éxito o fracaso del análisis completo de la imagen.

3. Por otro lado, la representación y descripción de los objetos dependen en parte del éxito en la extracción de características a partir de la imagen, las cuales pueden estar incompletas y/o hacer una selección inadecuada de los descriptores para los modelos en estudio.

4. Finalmente, en el reconocimiento e interpretación de las escenas, la reducción del espacio de búsqueda en la base de datos de los modelos, puede llevar a no indexar todos los grupos de modelos como correctos e indexar muchos grupos de modelos de forma incorrecta.

II.2.3 El sistema más sencillo para el reconocimiento de objetos

El objetivo principal de un sistema de reconocimiento de objetos es realizar la identificación de objetos dentro de un imagen. Regularmente, el sistema se divide en dos estados. En el estado de adquisición, se crean los modelos para las diferentes clases de objetos. En el estado de recuperación, se extraen un conjunto de características de la imagen. El punto crucial en la construcción de un sistema de reconocimiento de objetos es la selección de un modelo apropiado que represente a los datos. Una definición de un sistema de reconocimiento de objetos es la siguiente:

Definición 6. Sistema para el reconocimiento de objetos. *El sistema memoriza un número suficiente de imágenes de todas las diferentes clases de objetos a reconocer. Cuando se presenta una nueva imagen (con un objeto a reconocer), esta se compara con todas las imágenes memorizadas en el sistema. La clase de objetos memorizada que sea más parecida a la nueva imagen, identifica al objeto representado en la imagen ahora reconocida.*

Problemas

Los problemas en un sistema de reconocimiento de objetos se pueden dividir en dos categorías:

- El espacio de todas las posibles vistas de todos los objetos a ser reconocidos debe ser prohibitivamente grande.
- Y, fundamentalmente, la imagen a ser reconocida no será suficientemente parecida a cualquier imagen vista con anterioridad.

El primer punto implica que la memoria requerida para almacenar todas las distintas imágenes debe ser muy grande. Al mismo tiempo, la búsqueda en la memoria de la computadora es muy lenta.

El segundo punto se basa en el hecho de que un objeto puede ser visto con una gran variedad de apariencias. Las principales fuentes de variación incluyen:

- *Ángulo de vista*: Los objetos tridimensionales se pueden observar desde diferentes ángulos de vista (direcciones y distancias), y estos ángulos pueden incrementar el número de imágenes necesarias para el reconocimiento.
- *Efectos fotométricos*: Estos incluyen la posición y distribución de las fuentes de luz en la escena, sus longitudes de onda, los efectos de iluminación causada por otros objetos y la distribución de sombras.
- *Escenario del objeto*: Los objetos se pueden ver contra algún fondo y/o estar parcialmente ocluídos por otros objetos.
- *Cambios de forma*: Muchos objetos pueden cambiar su forma tridimensional (por ejemplo el cuerpo humano).

II.2.4 Estrategias computacionales para el reconocimiento de objetos

De acuerdo a Suetens *et al.* (1992) existen principalmente cuatro estrategias computacionales las cuales se resumen esquemáticamente en la Figura 6, estas son:

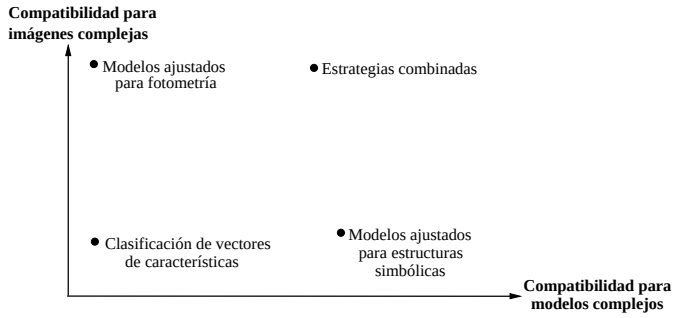


Figura 6: *Espacio de clasificación para las estrategias de reconocimiento de objetos.*

- *Clasificación de vectores de características*: Los métodos de vectores de características dependen de un modelo trivial de características de un objeto y regularmente se aplican a datos sencillos.

  En este enfoque, los objetos se modelan como vectores de características, cada uno de los cuales corresponde a un punto en el espacio multidimensional de características. Ejemplos de estas características incluyen valores de gris, color, intensidad infrarroja o ultravioleta, área, perímetro, etc. Para utilizar este enfoque, se deben seleccionar las características que son relevantes, determinar una manera de medirlas y definir un criterio para distinguir entre los objetos deseados de otros. Una vez que se define el espacio de características, este se debe dividir en regiones correspondientes a los diferentes modelos de objetos. Esto permite la asignación de objetos desconocidos a las clases de objetos conocidas. Las fronteras de

decisión usualmente se construyen durante la fase de aprendizaje/entrenamiento; por ejemplo, se toma un número grande de ejemplos de objetos con los valores de las características, se construye una gráfica de densidad de sus valores y se anotan las fronteras de las clases con diferentes etiquetas. La selección de clases se puede realizar por medio de técnicas de clasificación.

- *Modelos ajustados para fotometría*: Cuando los modelos sencillos son suficientes pero los datos fotométricos son ruidosos y ambiguos, pueden resultar efectivos un número de métodos que extraigan instancias de modelos sencillos. Tales métodos buscan características con formas globales y propiedades fotométricas predeterminadas. Estos métodos pueden utilizar modelos rígidos, dependiendo de un conjunto limitado de parámetros, o modelos flexibles, especificados por un conjunto de restricciones genéricas sobre las características de un objeto. Un ejemplo sencillo, es el procedimiento de buscar círculos a través de encontrar arcos con un radio y centro determinados, lo que es contrario a encontrar fronteras de claridad-oscuridad en la intensidad (valores de pixel) en los datos.

- *Modelos ajustados para estructuras simbólicas*: Estos modelos asumen que un conjunto de características se extraen de las imágenes por medio de alguna operación de preprocesamiento. Estas características se encuentran a través de operadores basados en estadísticas locales, sin utilizar información de la forma o conocimiento de la escena. Este proceso se conoce como segmentación. Las características, sin embargo, pueden ser intensidades de pixel o etiquetas. En estos casos, los procesos subsecuentes utilizan la salida simbólica del proceso inicial sin tomar en cuenta de nuevo a la imagen.

- *Estrategias combinadas*: Cuando los datos y las instancias del modelo deseado son complejas, el éxito del reconocimiento de objetos requiere de una combinación de

estrategias.

II.2.5 Aplicaciones

Existen diferentes trabajos y aplicaciones del reconocimiento de objetos (Draper y Baek, 2000; Edelman, 1997; Smart y Zhang, 2004; Winkeler y Manjunath, 1997). En esta sección se explican tres de ellas (reconocimiento tradicional de objetos, recuperación de imágenes basada en objetos y notas de video), además de las suposiciones y requerimientos de cada una.

Las aplicaciones de reconocimiento tradicional de objetos se enfocan en el reconocimiento de objetos tridimensionales en imágenes adquiridas por una cámara. Los ángulos de vista y la iluminación cambian entre imágenes, sin embargo, los parámetros de la formación de las imágenes se deben conocer *a priori*.

Un número de aplicaciones interesantes en reconocimiento incluyen servicios de robots, manejo de la calidad e inspección industrial de productos manufacturados.

La recuperación basada en objetos es un caso especial de la *recuperación de imágenes basada en el contenido*, la cual es la tarea de seleccionar imágenes relevantes para una aplicación, dada una colección normalmente grande de imágenes llamada *base de datos*. En un enfoque basado en objetos, las imágenes relevantes son aquellas que muestran al objeto buscado.

Utilizando las definiciones de la tarea de reconocimiento, la recuperación de imágenes basada en objetos es esencialmente idéntica a repetir la detección de objetos en una colección de imágenes. Sin embargo, los sistemas de recuperación mantienen la interacción con el usuario, lo cual no es tradicional en los sistemas de detección de objetos. Las aplicaciones de recuperación incluyen la búsqueda de catálogos, la recuperación de videos, la navegación visual en colecciones de arte, la exploración de satélites, bases de datos aéreas y médicas y el manejo de imágenes digitales del web y de librerías

de videos.

La búsqueda y recuperación de secuencias grandes de archivos de video son tareas embarazosas y costosas. Una solución para este problema es hacer notas automáticas de video; calcular las descripciones concisas de los datos de video, los cuales se usan para indexar (catalogar) y recuperar las secuencias originales. La tarea de las notas de video se puede reducir a la anotación de imágenes cuando un video se representa como una colección de imágenes independientes. Las propiedades de las secuencias de las diferentes imágenes/video se pueden explotar para representar datos visuales. Estos pueden ser atributos visuales como color y textura u otras propiedades detectadas como texto, movimiento, audio, etc. La medición de estos atributos se realiza a cada imagen como un todo (enfoque global) o como una descripción compuesta de regiones seleccionadas de la imagen las cuales deben corresponder a objetos particulares que aparecen en la imagen (enfoque local o basado en objetos).

Se han demostrado aplicaciones de anotación exitosas para recuperación de video en general. Por ejemplo, en anotaciones en videos de deportes se puede describir la secuencia por el tipo de deporte, el número de jugadores, los colores de sus uniformes, etc. Entonces, la recuperación se realiza simplemente consultando el sistema en las secuencias que muestren un equipo deseado, deporte, jugador, etc.

II.3 Análisis de expresiones faciales

El análisis de expresiones faciales tuvo sus inicios antes del siglo XIX. Darwin demostró en 1872 la universalidad de las expresiones faciales y su continuidad en hombres y animales y afirmó otras cosas, que existen emociones innatas específicas las cuales se originaron en los hábitos asociados. En 1971, Ekman y Friesen postularon seis emociones primarias cada una de las cuales posee un contenido distintivo asociado a una expresión facial única. Las representaciones de estos prototipos emocionales se conocen

como ***emociones básicas***. Estas son diferentes a través de las diferentes etnias y culturas y son: *felicidad, tristeza, miedo, disgusto, sorpresa y enojo.* En el pasado, el análisis de las expresiones faciales fue sujeto de investigación para los psicólogos, pero en 1978 Suwa *et. al.* presentaron una investigación preliminar en el análisis automático de expresiones faciales de una secuencia de imágenes. Ya en el siglo XX, la investigación en el análisis de expresiones faciales tuvo mucha inercia empezando con el trabajo de Mase y Pentland.

El análisis de expresiones faciales (del inglés facial expression analysis) es un área de investigación muy importante dentro de la comunidad de visión por computadora debido a sus implicaciones en el desarrollo de sistemas interactivos humano-computadora.

Las razones de este renovado interés en expresiones faciales son muchas pero principalmente son debido a los avances en áreas relacionadas como *la detección, seguimiento y reconocimiento de rostros* además de la reciente disponibilidad de poder computacional barato. Se pueden concebir distintas aplicaciones utilizando el análisis de expresiones faciales para un futuro cercano, fomentando el interés en hacer investigación en diferentes áreas incluyendo el entendimiento de imágenes, estudios sicológicos, nervios faciales en medicina y animación de rostros.

El análisis automático de expresiones faciales es una tarea compleja ya que la apariencia de los rostros puede variar considerablemente de un individuo a otro debido a su edad, etnia, género, productos cosméticos y oclusión de objetos como lentes y cabello. En la Tabla I se presentan diversos trabajos a realizar en el área y sus fuentes de variación correspondiente.

El proceso del análisis de expresiones faciales se puede dividir en varios estados para la construcción de un sistema automático de expresiones faciales. Esto se muestra gráficamente en la Figura 7.

A continuación se describen los estados principales del proceso dentro de esta área.

Tabla I: Fuentes de variación en apariencias faciales.

Fuente	*Tareas posibles*
Identidad	Clasificación, conocido-desconocido, verificación, identificación completa
Expresión facial	Inferencia de emociones o intenciones
Habla	Lectura de labios
Sexo	Decisión de si es hombre o mujer
Edad	Estimación de la edad

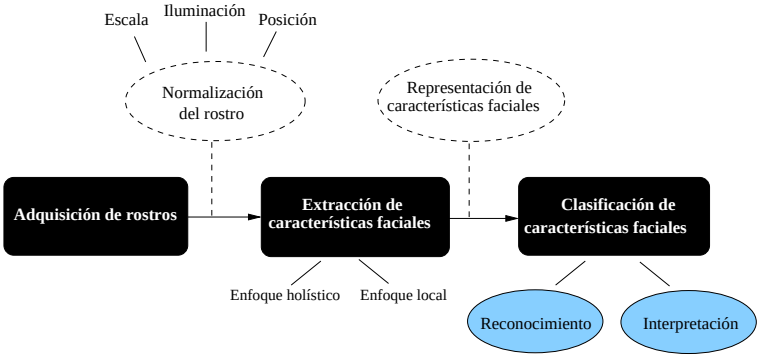


Figura 7: *Esquema genérico del análisis de expresiones faciales*

II.3.1 Adquisición de rostros

Idealmente, en la fase de adquisición de rostros se requiere de un detector automático de rostros que mantenga la ubicación de éstos en escenas complejas con la presencia de fondos desconocidos. Ciertos métodos de análisis de rostros necesitan la posición exacta de la cara con el fin de extraer las características faciales que son de interés mientras que en otros solo se encuentra disponible la ubicación de ésta.

El análisis de rostros es complejo por los cambios de apariencia del rostro, causado principalmente por la variación de la posición de la cara y los cambios en la iluminación.

- **Posición**: El rostro es una figura en tercera dimensión (3D). La apariencia de

las expresiones faciales depende del ángulo y la distancia a la cual se observa un rostro dado.

- **Iluminación**: La variación de la iluminación es inevitable y se presenta con solo cambiar la posición del rostro. Las condiciones de luz pueden variar de un día a otro y de ambientes al aire libre (exteriores) e interiores.

II.3.2 Extracción de características

Los métodos de extracción de características se pueden categorizar, principalmente, si éstos actúan de manera local u holística.

Enfoque local contra enfoque holístico

El procesamiento de las características del rostro puede tomar lugar de manera holística, donde las caras son procesadas como un todo, o localmente, enfocándose en las características del rostro o áreas que están propensas a cambiar con las expresiones faciales. Se pueden distinguir dos tipos de características faciales:

- *Características faciales no-transitorias* están siempre presentes en la cara pero pueden deformarse debido a las expresiones faciales. Entre estas están los ojos, las cejas y la boca.

- *Características faciales transitorias* abarcan diferentes tipos de arrugas y protuberancias que ocurren con las expresiones faciales. Especialmente, la frente, regiones alrededor de la boca y los ojos son propensos a contener éstas características faciales transitorias.

II.3.3 Clasificación de expresiones faciales

La clasificación de las características se lleva a cabo en la última etapa de un sistema de análisis automático de expresiones faciales. Esta se realiza haciendo el reconocimiento

de expresiones faciales.

Reconocimiento de expresiones faciales

Enfoques tradicionales han sido introducidos por científicos en computación en el curso de sus investigaciones en expresiones faciales. Al inicio era necesaria la experiencia humana para mapear estas representaciones simbólicas en emociones. La tarea de hacer clases de emociones manualmente es sumamente difícil por lo que se ha hecho investigación en esta área para realizarlo de manera automatizada.

Resumiendo el proceso de análisis de expresiones faciales: antes de analizar una expresión facial, primero se debe detectar el rostro en una escena. Después se conciben los mecanismos para la extracción de la información de expresiones faciales del rostro observado en la imagen. Al proceso de extracción de la información de las expresiones faciales se le conoce como *ubicación* del rostro y sus características en la escena. El rostro se puede representar de varias maneras ya sea como una unidad completa (representación holística o integral), como un conjunto de características (representación analítica o local) o como la combinación de éstas (enfoque híbrido). La elección de la representación del rostro y del tipo de imágenes de entrada determinan los mecanismos seleccionados para la extracción automática de la información de las expresiones faciales. El paso final es definir algún conjunto de categorías las cuales se querrán usar para la clasificación y/o interpretación de las expresiones faciales.

II.4 Análisis de textura

El análisis de textura es una tarea básica en procesamiento de imágenes y visión por computadora el cual ha sido un tema de investigación activa por más de tres décadas. Este es un tema clave en muchas áreas de aplicación tales como: reconocimiento de objetos, sensado remoto, recuperación de imágenes/video basadas en su contenido,

entre otras. Aún cuando existe un trabajo considerablemente grande en la utilización de textura visual para la resolución de distintos problemas, la definición de textura es todavía imprecisa, sin embargo, proporciona intuitivamente medidas de propiedades tales como suavizado, rugosidad y regularidad, ver Figura 8.

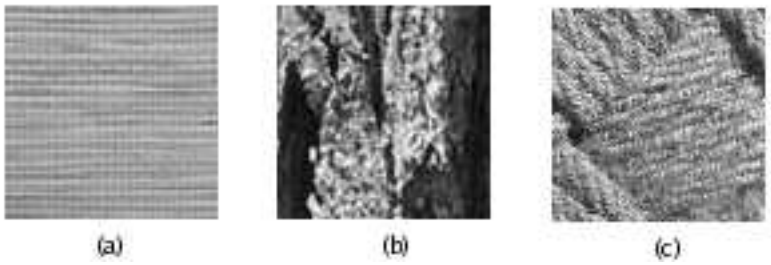


Figura 8: *Ejemplos de texturas:* (a)*suavizado,* (b)*rugosidad y* (c)*regularidad*

Wechsler (1981) define a la textura como una calidad de superficie de objetos o fenómenos con una dimensión visual y táctil. Van Gool *et al.* (1986) la definen como un número de elementos estructurados combinados. Ninguno de estos elementos estructurizados tienen significados por sí mismos. Haralick *et al.* (1973) la asumen como una distribución espacial estadística de niveles de gris.

Como no existe un acuerdo en la definición formal de textura, una característica de esta es la *repetición de un patrón o patrones dentro de una región.* El patrón se puede repetir exactamente o como un conjunto de pequeñas variaciones. En texturas sintéticas, como en líneas horizontales, líneas verticales o un tablero de ajedrez; la estructura básica se repite exactamente. En texturas naturales, como en el pasto, madera, arena o rocas existe una variación aleatoria en el tamaño, forma, intensidad o color en las repeticiones de la estructura básica, ver Figura 9.

Existen cuatro tareas principales en análisis de textura:

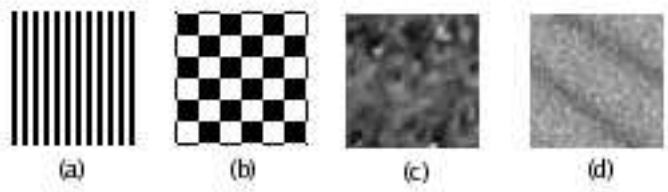


Figura 9: *Ejemplos de texturas artificiales* (a, b) *y naturales* (c, d)

- **Extracción de características**: obtener una característica de una imagen digital con el fin de describir numéricamente sus propiedades de textura.
- **Discriminación de texturas**: dividir una imagen texturizada en regiones perceptualemnte homogéneas (para segmentación de imágenes).
- **Clasificación de texturas**: Determinar a qué clase, dentro de un número finito de clases definido físicamente, pertenece una región homogénea dada.
- **Forma por medio de texturas**: Reconstruir superficies tridimensionales a través de información de texturas.

La extracción de características es la primera etapa en el análisis de textura en imágenes. Es de gran importancia ya que los resultados obtenidos de esta se utilizan para la discriminación, clasificación o determinación de la forma por medio de textura.

II.4.1 Extracción de características

Como se mencionó anteriormente, durante esta etapa se obtienen características de una imagen con el fin de describir numéricamente sus propiedades de textura. Dependiendo del problema a resolver, la extracción de características se puede realizar de tres maneras:

- **Extracción automática de características**: un sistema es capaz de determinar una región rectangular en la imagen la cual representa las características de textura de dicha imagen. Puesto que esta región es más pequeña que la imagen entera y es una parte representativa de esta, trabajar con dicha región disminuye el tiempo de extracción de características.

- **Extracción semi-automática de características**: en muchas aplicaciones, los usuarios no están interesados en la textura de una imagen entera pero sí de una región de interés específica. La región de interés es seleccionada por el usuario.

- **Extracción de características de una imagen entera**: este es el caso por omisión y es significativo cuando es de interés la imagen entera.

La mayoría de las personas entienden lo que es la textura y si se les pregunta acerca de una definición de su percepción de una superficie texturizada pueden dar una respuesta, pero cuantificar dicha percepción varía de persona a persona. Por este motivo se han desarrollado diferentes métodos que describen la textura de una imagen de manera automática.

II.4.2 Descriptores de textura

La noción de primitiva es clave en la descripción de texturas. Un texel (texture element, elemento de textura) es una primitiva visual con ciertas propiedades invariantes que ocurren repetidamente en diferentes posiciones, deformaciones y orientaciones en una región dada. Una propiedad básica de dichos elementos podría ser que sus pixeles tienen un nivel de gris constante. El aspecto más relevante de la definición de texel es que las primitivas deben ocurrir repetidamente dentro de una región dada. Un método muy empleado para la descripción de regiones de imágenes consiste en cuantificar su

contenido de textura. Los tres métodos principales más utilizados para describir la textura de una región son los estructurales, espectrales y estadísticos (Gónzalez y Woods, 1996). Las técnicas estructurales tratan de la composición de primitivas de imágenes, por ejemplo la descripción de texturas basadas en líneas paralelas regularmente espaciadas. Las técnicas espectrales se basan en las propiedades del espectro de Fourier y se utilizan primordialmente para detectar la periodicidad global de una imagen mediante la identificación de picos estrechos de alta energía del espectro. Las soluciones estadísticas proporcionan características de texturas tales como suavidad, rugosidad, granularidad, y otras similares.

Métodos estructurales

Se aplican a texturas altamente regulares, es decir, se pueden describir en base a formas básicas que se repiten uniformemente. Cuando los patrones de textura están muy bien definidos, el plano aparece con una teselación[2] muy bien ordenada. En una teselación regular todos los polígonos que rodean un vértice tienen el mismo número de lados. Las teselaciones semi-irregulares tienen dos clases de polígonos con diferentes números de lados rodeando un vértice. Este tipo de teselaciones se pueden describir convenientemente listando, en orden, el número de lados de los polígonos rodeando cada vértice. Así, para la Figura 10 la teselación (a) sería $(6, 6, 6)$, para la (b) $(3, 3, 3, 3, 3, 3)$ y para la (c) $(3, 6, 3, 6)$.

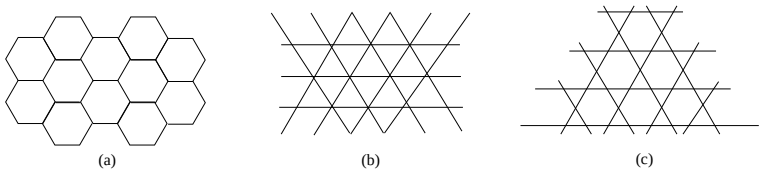


Figura 10: *Dos estructuras regulares* $(a$ *y* $b)$ *y una semi-irregular* (c)

[2]Teselación: división del plano en sectores de forma idéntica.

Otra forma de describir los métodos estructurales es a través de gramáticas. Una de las más interesantes es la denominada gramática de forma. Una gramática de forma se define como una 4-tupla (V_t, V_m, R, S) donde:

- V_t es un conjunto finito de formas.
- V_m es un conjunto finito de formas tales que $V_t \cap V_m = \varnothing$
- R es un conjunto finito de pares ordenados (u, v) tales que u es la forma que consta de un elemento de V_{t^+} combinada con un elemento de V_{m^+}, y v es la forma que consta de un elemento de V_{t^*} combinada con un elemento de V_{m^*}.
- S es una forma que consta de un elemento de V_{t^*} combinada con un elemento de V_{m^*}.

Los elementos del conjunto V_t se denominan elementos de forma terminales (o simplemente terminales). Los elementos del conjunto V_m se denominan elementos de forma no terminales (o marcadores). Los elementos del conjunto V_{t^+} están formados por la unión finita de uno o más elementos de V_t en el cual cualquier elemento puede usarse un número finito de veces en cualquier localización, orientación o escala. El conjunto $V_{t^*} = V_{t^+} \cup \{A\}$ donde:

- A es la forma vacía.
- Los conjuntos V_{m^+} y V_{m^*} se definen de forma similar.
- Los elementos de (u, v) de R se denominan reglas de forma y se escriben como uv, donde u es la parte izquierda de la regla y v la parte derecha. La forma inicial de la regla se conoce como S y normalmente contiene una u tal que hay un (u, v) que es un elemento de R.

Una textura se genera a partir de una gramática de forma comenzando con la forma inicial y aplicando repetidamente las reglas de forma. El resultado de aplicar una regla de forma R a una forma dada S es otra forma que consta de S con el lado derecho de R sustituído en S para una ocurrencia del lado izquierdo de R.

El proceso de aplicación de reglas a una forma se realiza de la siguiente manera:

1. Encontrar la parte de la forma que es geométricamente similar al lado izquierdo de una regla en términos tanto de elementos terminales como no terminales (marcadores). Debe haber una correspondencia uno a uno entre los terminales y marcadores en el lado izquierdo de la regla, y los terminales y marcadores en la parte derecha a la cual se aplica la regla.

2. Encontrar las transformaciones geométricas (escala, rotación, traslación) que hace el lado izquierdo de la regla idéntico a la parte correspondiente en la forma.

3. Aplicar esas transformaciones al lado derecho de la regla.

4. Sustituir al lado derecho transformado de la regla por la parte de la forma que corresponde al lado izquierdo de la regla.

5. Terminar el proceso de generación cuando no hay regla en la gramática que pueda aplicarse.

En la Figura 11 se muestra un ejemplo sencillo de aplicación, considerando la generación de una textura hexagonal.

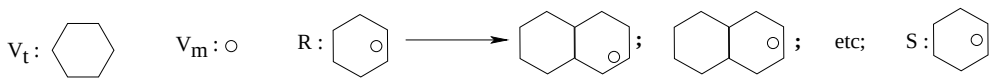


Figura 11: *Ejemplo de generación de una textura hexagonal.*

Métodos espectrales

El espectro de Fourier se utiliza idealmente para describir la direccionalidad de patrones bidimensionales periódicos o casi periódicos de una imagen. Estos patrones de textura global, aunque son fácilmente distinguibles como concentraciones de alta energía del espectro, generalmente son bastante difíciles de detectar con métodos espaciales a causa de la naturaleza local de estas técnicas.

Se pueden considerar tres características del espectro de Fourier que son útiles para la descripción de textura:

1. Picos prominentes del espectro que dan la dirección principal de los patrones de textura.

2. La localización de los picos en el plano de frecuencia proporciona el período espacial fundamental de los patrones.

3. La eliminación de los componentes periódicos mediante el filtrado deja elementos de la imagen no periódicos, que se pueden describir por métodos estadísticos.

Es importante saber que el espectro de una imagen real es simétrico con respecto al origen, así pues sólo se necesita considerar la mitad del plano de frecuencia. Por lo tanto, a efectos del análisis, cada patrón periódico está asociado con un sólo pico del espectro, en lugar de con dos.

La detección e interpretación de las características del espectro que se acaban de mencionar a veces se simplifican expresando el espectro en coordenadas polares para producir una función $S(r, \theta)$, siendo S la función del espectro y r y θ las variables de este sistema de coordenadas. Para cada dirección θ, se pueden considerar $S(r, \theta)$ como una función unidimensional $S_\theta(r)$. De forma similar, para cada frecuencia r, $S_r(\theta)$ es una función unidimensional. Analizando $S_\theta(r)$ para un valor fijo de θ se obtiene el

comportamiento del espectro (tal como la presencia de picos) a lo largo de una dirección radial desde el origen, mientras que analizando $S_r(\theta)$ para un valor fijo de r se obtiene el comportamiento a lo largo de un círculo centrado en el origen.

Una descripción más global se obtiene integrando (sumando variables discretas):

$$S(r) = \sum_{\theta=0}^{\pi} S_\theta(r) \quad , \tag{6}$$

y

$$S(\theta) = \sum_{r=1}^{R} S_r(\theta) \quad , \tag{7}$$

siendo R el radio de un círculo centrado en el origen. Para un espectro de tamaño $N \times N$, R se escoge normalmente como $\frac{N}{2}$.

Los resultados de (6) y (7) constituyen un par de valores $[S(r), S(\theta)]$ para cada par de coordenadas (r, θ). Variando estas coordenadas se pueden generar dos funciones unidimensionales, $S(r)$ y $S(\theta)$, que constituyen una descripción de la energía espectral de la textura para una imagen o región completa. Más aún, los mismos descriptores de estas funciones se pueden calcular para caracterizar su comportamiento cuantitativamente. Los descriptores que normalmente se utilizan para este fin son la localización del valor máximo, de la media y la varianza de la amplitud y de las variaciones axiales, y la distancia entre los valores máximo y medio de la función.

Se muestra un ejemplo en la Figura 12 la cual ilustra el empleo de (6) y (7) para una descripción global de la textura. La Figura 12(a) muestra una imagen con textura periódica, y la Figura 12(b) muestra su espectro. Las Figuras 12(c) y 12(d) muestran representaciones de $S(r)$ y $S(\theta)$, respectivamente. La representación de $S(r)$ es una estructura típica , que tiene un alto contenido de energía cerca del origen y progresivamente valores inferiores en las frecuencias más altas. La representación de $S(\theta)$ muestra

picos prominentes a intervalos de 45°, que claramente corresponden a la periodicidad del contenido de la textura de la imagen.

Haciendo una ilustración de cómo se podría utilizar una representación de $S(\theta)$ para diferenciar entre dos formas de textura, la Figura 12(e) muestra una imagen cuya forma de textura predomina en las direcciones horizontal y vertical. La Figura 12(f) muestra la representación de $S(\theta)$ para el espectro de la imagen. Como era de esperar, esta representación muestra picos elevados a intervalos de 90°. El discriminar entre dos formas de textura analizando sus formas de onda correspondientes $S(\theta)$ sería un proceso directo.

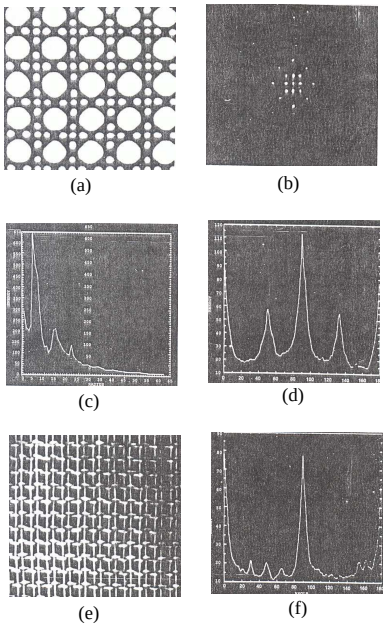


Figura 12: (a) *imagen con una textura periódica;* (b) *espectro;* (c) *diagrama de* $S(r)$*;* (d) *diagrama de* $S(\theta)$*;* (e) *otra imagen con un tipo de textura diferente;* (f) *diagrama de* $S(\theta)$.

Métodos estadísticos

Los métodos estadísticos analizan la distribución de alguna propiedad de textura para cada uno de los pixeles de una imagen. Dependiendo del número de puntos que definen la textura se clasifican en estadísticas de primer orden, segundo orden u órdenes superiores.

Métodos estadísticos de primer orden

Por primer orden se entienden las estadísticas en las que se ven involucrados pixeles simples en contraposición a las estadísticas de más de un pixel (pares, tripletas, etc.). En las estadísticas de primer orden se puede utilizar el histograma del nivel de gris de la textura, cuya normalización proporciona la función de densidad de probabilidad de la imagen caracterizada por la textura. Se puede pues, comparar los histogramas normalizados del nivel de gris de imágenes de texturas, o utilizar varias medidas derivadas, tales como la media, la mediana o la varianza, ver Figura 13. *En estas medidas no se considera la relación entre los pixeles.*

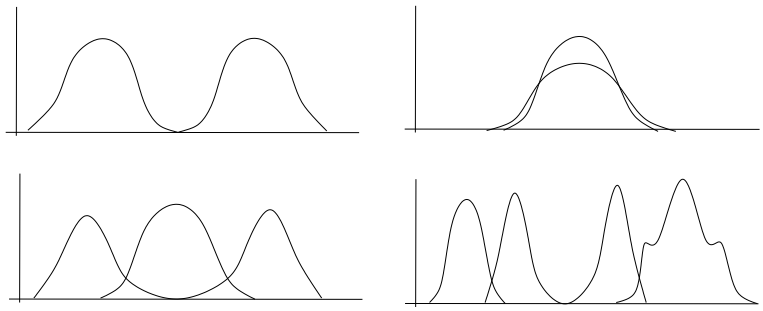


Figura 13: *Histogramas con distinta media, desviación estandar, coeficiente de asimetría y apuntamiento.*

Sea z una variable aleatoria que representa la intensidad discreta de la imagen y sea $p(z_i)$ con $i = 1, ..., L$ su histograma correspondiente, donde L es el número de niveles de

intensidades diferentes. El momento n-ésimo de z respecto de la media se define como:

$$\mu_n(z) = \sum_{i=1}^{L}(z_i - m)^n p(z_i) \quad , \tag{8}$$

donde m es el valor medio de z y por tanto la intensidad media de la imagen

$$m = \sum_{i=1}^{L} z_i p(z_i) \quad . \tag{9}$$

- *Momento de segundo orden*, denominado varianza σ es de particular interés para la descripción de texturas. La varianza es una medida de contraste de intensidad que se puede usar para obtener descriptores de suavidad relativa

$$\sigma^2 = \sum_{i=1}^{L}(z_i - m)^2 p(z_i) \quad . \tag{10}$$

- *Momento de tercer orden* es una medida de la oblicuidad del histograma, o coeficiente de asimetría

$$\mu^3 = \sum_{i=1}^{L}(z_i - m)^3 p(z_i) \quad . \tag{11}$$

- *Momento de cuarto orden* es una medida de cuán plano es el histograma, apuntamiento o Kurtosis

$$\mu^4 = \sum_{i=1}^{L}(z_i - m)^4 p(z_i) \quad . \tag{12}$$

A partir del momento de quinto orden no es tan fácil relacionar los momentos con la forma del histograma, pero sirven para obtener más información cuantitativa de discriminación de texturas.

La principal limitación de las estadísticas de primer orden es su falta de sensibilidad ante permutaciones de los pixeles; por ejemplo, un patrón de textura de un tablero de

ajedrez que tenga su primera mitad izquierda blanca y su primera mitad derecha negra tiene la misma estadística de primer orden que un patrón con el orden inverso.

Métodos estadísticos de segundo orden

Al estar basados en el histograma, los estadísticos de primer orden tienen el inconveniente de no captar toda la información espacial. Así se obtendría el mismo resultado para un tablero de ajedrez que para otro con los colores blanco y negro intercambiados. Es por ello que se desarrollaron los métodos estadísticos de segundo orden los cuales son medidas que consideran la relación de coocurrencia entre *grupos de dos pixeles* de la imagen original y a una distancia dada.

1. Concepto de matriz de coocurrencia

La matriz de coocurrencia en niveles de gris (GLCM, gray level coocurrence matrix) describe la frecuencia de un nivel de gris que aparece en una relación espacial específica con otro valor de gris, dentro del área de una ventana determinada. La matriz de coocurrencia es un resumen de la forma en que los valores de los pixeles ocurren al lado de otro valor en una pequeña ventana.

De manera formal, la GLCM $M_{i,j}(\pi)$ define una función de densidad de probabilidad $f(i,j|\mathbf{V},\pi)$ donde i y j son los niveles de gris de dos pixeles separados por un vector $\mathbf{V}$, y $\pi = \mathbf{V}, R$ es el conjunto de parámetros para $M_{i,j}(\pi)$. La GLCM muestra qué tan seguido los pixeles, que definen al vector $\mathbf{V}(d,\theta)$ y que se encuentran separados por una cierta cantidad de valores de intensidad $\Delta = i - j$, aparecen en una región R de una imagen I dada. El vector $\mathbf{V}$ define la distancia d y la orientación θ entre los dos pixeles. Se puede o no tomar en cuenta la dirección de $\mathbf{V}$ cuando se calcula la GLCM.

Normalmente el procedimiento de generación de imágenes de textura requiere que el analista defina tres variables:

1. Tamaño de la ventana.

2. Las características de textura a utilizar.

3. El vector **V** (la distancia entre pixeles y la orientación para el cálculo de la co-ocurrencia).

Respecto del tamaño de la ventana, esta debe ser cuadrada y con un número impar de pixeles. El resultado del cálculo de la textura es un único número que representa la ventana completa, el cual es colocado en el lugar del pixel central. Luego, la ventana se mueve un pixel y el cálculo se repite obteniendo una nueva matriz de coocurrencia para esta nueva ventana y resultando un nuevo valor, para el pixel central de esta nueva posición de la ventana. De este modo se construye toda una nueva imagen con valores de texturas.

Cada celda de la ventana debe situarse en una celda que esté ocupada en la imagen original. Esto significa que el pixel central de la ventana *no* puede ocupar un borde de la imagen. Si una ventana tiene dimensiones $N \times N$, una franja de $\frac{N-1}{2}$ pixeles alrededor de la imagen permanecerán sin resultados. Usualmente los pixeles de borde representan una pequeña fracción de las imágenes, por lo cual es un problema menor. Sin embargo, si la imagen es muy pequeña o la ventana muy grande, este efecto debe considerarse en el análisis de los resultados. Una forma de solucionar este problema es llenar esas celdas con el valor calculado para el pixel más cercano.

Por otra parte, el tamaño relativo de la ventana y de los objetos en la imagen determinan la utilidad de esta medida para la clasificación. Es recomendable que la ventana sea menor que el objeto y lo suficientemente grande como para capturar la variabilidad del mismo. Por ejemplo, en un bosque la textura está determinada por las luces y sombras de las copas. Una ventana con el tamaño de un solo árbol no medirá la textura del bosque. Otra ventana cubriendo todo el bosque y los campos vecinos a él, tampoco medirá la textura del mismo.

La orientación del pixel de referencia y su vecino puede ser en cualquiera de las 8 direcciones: norte, sur, este, oeste y las 4 diagonales (esto también se representa como 0°, 45°, 90° y 135° y sus opuestos).

2. Cálculo de la matriz de coocurrencia

En la Figura 14 se muestra un ejemplo de una GLCM. La matriz corresponde a una pequeña ventana de la imagen de la izquierda, la cual se calculó con una distancia $d = 1$ y una orientación $\theta = 0$.

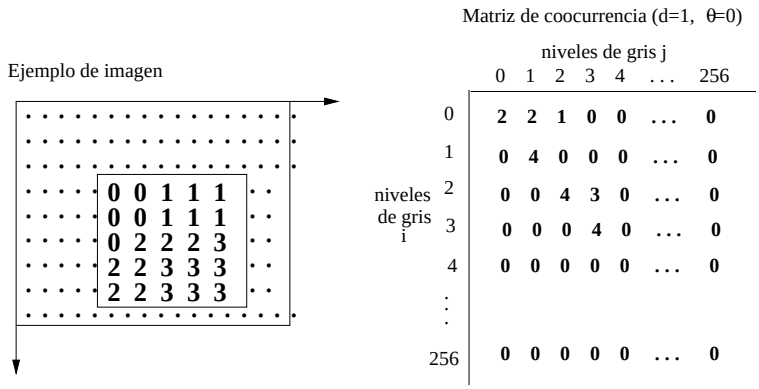


Figura 14: *Matriz de coocurrencia*

3. Relación espacial entre dos pixeles

La GLCM considera la relación espacial entre dos pixeles, llamados *pixel de referencia* y *pixel vecino*. Por ejemplo, si se escoge el pixel vecino que está situado un pixel a la derecha de cada pixel de referencia, esto se expresa como $(d = 1, \theta = 0°)$.

Cada pixel en la ventana se va convirtiendo sucesivamente en el pixel de referencia, empezando por el ubicado arriba a la izquierda y finalizando abajo a la derecha. Los pixeles ubicados en el margen derecho de la imagen original, no tienen vecino a la derecha por lo tanto no son utilizados en el cálculo.

4. Distancia entre pixeles

Cuando la ventana es lo suficientemente grande, se puede usar una distancia mayor, sin que haya diferencias en la metodología de su cálculo, por lo tanto, *existen diferentes GLCMs para cada relación espacial, según se considere la orientación del vecino.*

5. Normalización de la matriz de coocurrencia

Si el cálculo de la GLCM se realiza a una sola orientación entonces el número de veces que aparece la combinación i, j no es el mismo que la combinación j, i, por lo tanto, la matriz no es simétrica respecto a la diagonal.

Sin embargo, la simetría es necesaria para el cálculo, esto se logra si cada par de pixeles se cuentan dos veces: una vez a la derecha y otra vez a la izquierda (se intercambian los pixeles de referencia y vecino en el segundo cálculo).

Para obtener una matriz simétrica, la forma más sencilla, en vez de contar dos veces, es sumarle a esta matriz su matriz traspuesta. La matriz traspuesta se obtiene intercambiando las filas y columnas de la matriz original.

Una vez obtenida la matriz simétrica, el paso siguiente es expresar esa matriz como probabilidad. La definición más simple de la probabilidad es: *el número de veces que ocurre un evento, dividido por el número total de posibles eventos* y la fórmula para su cálculo es

$$P_{i,j} = \frac{I_{i,j}}{\sum_{i,j=0}^{N-1} I_{i,j}} \quad , \tag{13}$$

donde:

i es el número de filas y j es el número de columnas.

I es el valor de intensidad de la celda i, j en la ventana.

$P_{i,j}$ es la probabilidad en la celda i, j.

N es el número de filas o columnas.

La fórmula (13) transforma la GLCM en una aproximación de tabla de probabilidad. Decimos que es una aproximación, porque una verdadera probabilidad requiere de valores continuos, y los valores en niveles de gris son valores enteros, por lo tanto discretos.

En general, cuanto mayor es el número de la diagonal en la GLCM, más homogénea es la textura en esa parte de la imagen que está siendo analizada.

Con respecto a la GLCM simétrica y normalizada hay algunos aspectos a resaltar:

- Los elementos de la diagonal representan pares de pixeles que no tienen diferencias en sus niveles de gris. Si estos elementos tienen probabilidades grandes, entonces la imagen no muestra mucho contraste, la mayoría de los pixeles son idénticos a sus vecinos.

- Sumando los valores de la diagonal tenemos la probabilidad que un pixel tenga el mismo nivel de gris que su vecino.

- Las líneas paralelas a la diagonal separadas una celda, representan los pares de pixeles con una diferencia de un nivel de gris. De la misma manera, sumando los elementos separados dos celdas de la diagonal, tenemos los pares de pixeles con dos valores de grises de diferencia. A medida que nos alejamos de la diagonal, la diferencia entre niveles de gris es mayor.

- Sumando los valores de estas diagonales paralelas obtenemos la probabilidad de que un pixel tenga uno, dos, tres, etc., niveles de gris de diferencia con su vecino.

6. Propiedades de la matriz de coocurrencia

- **Cuadrada**. El rango de los valores de los pixeles de referencia y el de los vecinos es el mismo, por lo tanto el número de las filas y las columnas es idéntico.

- **Tiene el mismo número de filas y columnas que el número de bits en la imagen**. Los datos de 8 bits tienen 256 ($2^8 = 256$) valores posibles, así la matriz de coocurrencia es de 256×256, con $65,536$ celdas. Datos en 16 bits originan matrices de $65,536 \times 65,536$ con $429,496,720$ celdas.

- **Es simétrica con respecto a la diagonal**. Una matriz simétrica significa que los mismos valores ocurren en las celdas opuestas a la diagonal. Por ejemplo, el valor de la celda i, j debería ser el mismo que el valor en la celda j, i para que la matriz sea simétrica.

7. Descriptores de textura

Una desventaja de la GLCM es que, cuando la cantidad de niveles de gris de la región R crece, las dimensiones de la GLCM hacen difícil su manejo o uso directamente. Afortunadamente, la información que proporciona la GLCM se puede expresar por un conjunto variado de descriptores númericos estadísticos. Esto reduce la dimensionalidad de la información utilizada. El conjunto de descriptores disponibles (Haralick *et al.*, 1973; Haralick, 1979) y ampliamente utilizados (Bolon *et al.*, 1995; Parker, 1997) que se obtienen de la GLCM $M(i, j)$ son los siguientes:

- ***Entropía***: Término aplicado comúnmente en termodinámica o mecánica estadística. La entropía mide la cantidad de desorden o caos en un sistema. Imágenes con alta homogeneidad tienen asociado un valor pequeño de entropía, mientras que las imágenes que tienen poca homogeneidad poseerán un valor alto de entropía. La entropía está dada por la siguiente función:

$$H = 1 - \frac{1}{Nc.ln(Nc)} \sum_i \sum_j M(i,j).ln(M(i,j)).1_{M(i,j)} \quad , \tag{14}$$

donde $1_{M(i,j)} = 0$ cuando $M(i, j) = 0$ y 1 en otro caso, y Nc es el número de

ocurrencias en M.

- ***Contraste***: Es una medida de la variación local en una imagen, es decir, mide la diferencia entre valores de intensidad de pixeles vecinos. Su medida es baja si existen valores altos cerca de la diagonal; y si los términos mayores se encuentran más lejanos de la diagonal, entonces más grandes serán los valores de contraste. Esto ocurre cuando se pasa frecuentemente de un pixel muy claro a un pixel muy oscuro o visceversa. El contraste se calcula como sigue:

$$C = \frac{1}{Nc(L-1)^2} \sum_{k=0}^{L-1} k^2 \sum_{|i-j|=k} M(i,j) \quad , \tag{15}$$

donde L es el número de niveles de gris en la imagen. Entonces la matriz tiene un tamaño de $L \times L$.

- ***Homogeneidad***: Es la similitud entre los datos, es decir, la cantidad de datos que cumplen con propiedades para ser considerados homogéneos dentro de la imagen. Es lo opuesto al contraste, esto es, mientras más grandes sean los valores cerca de la diagonal será mayor la homogeneidad y esta dada por:

$$Ho = \frac{1}{Nc^2} \sum_i \sum_j M(i,j)^2 \quad . \tag{16}$$

- ***Homogeneidad local***: Esta medida también depende de la homogeneidad de la imagen. Debido al factor de ponderación $(1 + (i-j)^2)^{-1}$, este descriptor obtiene pequeñas contribuciones de $M(i,j)$ cuando $i \neq j$ (valores de M relacionados con imágenes poco homogéneas) y contribuciones mayores de la diagonal de M (valores relacionados con imágenes homogéneas). El resultado será un valor elevado para imágenes homogéneas y viceversa. La homogeneidad local está dada por:

$$G = \frac{1}{Nc} \sum_i \sum_j \frac{M(i,j)}{1+(i-j)^2} \quad . \tag{17}$$

- ***Directividad***: El valor es grande si se tienen dos pixeles del mismo nivel de gris separados por una traslación dada una imagen. La expresión es:

$$D = \frac{1}{Nc} \sum_i M(i,i) \quad . \tag{18}$$

- ***Uniformidad***: Es importante si se encuentran muchos pixeles de un mismo nivel de gris dada una dirección. La uniformidad se calcula de la siguiente manera:

$$U = \frac{1}{Nc^2} \sum_i M(i,i)^2 \quad . \tag{19}$$

- ***Momentos***: Los momentos expresan información estadística comunes, como la varianza que corresponde al segundo momento. El valor del descriptor crece cuando la mayoría de los valores de $M(i,j)$ se encuentran fuera de la diagonal.

$$Mom_k = \sum_i \sum_j (i-j)^k M(i,j) \quad . \tag{20}$$

- ***Momentos inversos***: Es lo opuesto al descriptor anterior.

$$Mom_k^{-1} = \sum_i \sum_j \frac{M(i,j)}{(i-j)^k}, \qquad i \neq j \quad . \tag{21}$$

- ***Probabilidad máxima***: Considerando que la GLCM es una aproximación de la densidad de probabilidad entre pixeles, este descriptor extrae la diferencia más probable entre niveles de gris, es decir, es el valor más alto dentro de la matriz. Al calcular varias matrices con diferentes orientaciones y conocer cuál contiene el máximo, esto puede indicar la dirección para examinar la textura.

$$max(M(i,j)) \quad . \tag{22}$$

- ***Correlación***: Esta es una medida de la dependencia lineal de niveles de gris entre pixeles con posiciones relativas específicas respecto a otros. Es decir, pixeles más cercanos están más correlacionados entre sí que los pixeles más distantes.

$$S = \frac{1}{Nc.\sigma_x.\sigma_y} |\sum_i \sum_j (i - m_x)(j - m_y) M(i,j)| \tag{23}$$

$$m_x = \tfrac{1}{Nc} \textstyle\sum_i \sum_j i M(i,j)$$
$$m_y = \tfrac{1}{Nc} \textstyle\sum_i \sum_j j M(i,j)$$
$$\sigma_x^2 = \tfrac{1}{Nc} \textstyle\sum_i \sum_j (i - m_x)^2 M(i,j)$$
$$\sigma_y^2 = \tfrac{1}{Nc} \textstyle\sum_i \sum_j (j - m_y)^2 M(i,j)$$

Métodos estadísticos de tercer orden y de orden superior

Se consideran las relaciones *entre 3 y más pixeles*. Si bien su cálculo es teóricamente posible, no se implementan pues requieren mucho tiempo de cálculo y su resultado es de difícil interpretación.

II.5 Computación evolutiva

Dentro de la inteligencia artificial se pueden distinguir dos enfoques: el subsimbólico y el simbólico (ver Figura 15). El enfoque subsimbólico, primer período de la AI comprendido entre 1950 y 1965, utilizó representaciones numéricas del conocimiento. El enfoque simbólico, comprendido entre 1965 y 1975, utilizó representaciones simbólicas basadas en un número finito de primitivas y de reglas para manipulación de símbolos. Curiosamente, es el enfoque subsimbólico en donde se ha visto un mayor interés en los últimos años.

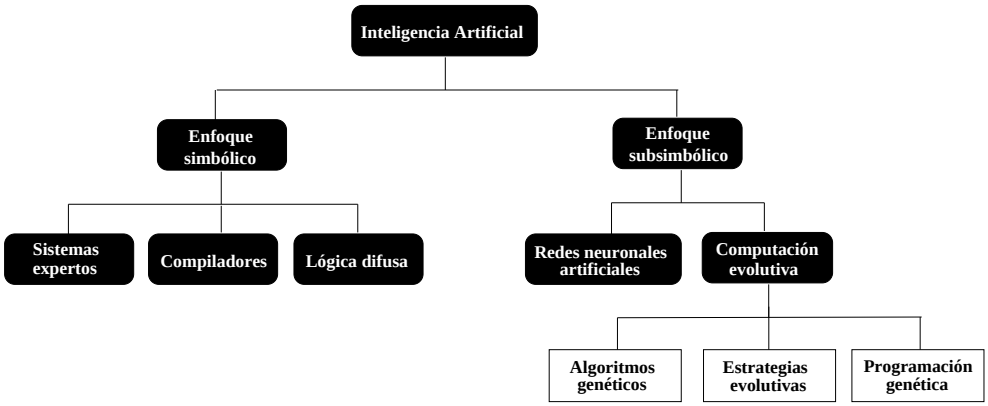


Figura 15: *La Computación Evolutiva como una rama de la inteligencia artificial y sus campos de estudio.*

El enfoque subsimbólico se caracteriza por crear sistemas con capacidad de aprendizaje. Este se puede obtener a nivel de individuo imitando el cerebro (redes neuronales); o a nivel de especie imitando la evolución (computación evolutiva). Por tanto, la inteligencia artificial no sólo consiste en idear algoritmos y estructuras de datos para solucionar problemas. También trata de la inteligencia humana, y por ende, sobre la vida.

Dentro de la IA, la computación evolutiva *observa* e *interpreta* la vida natural como una inmensa máquina de resolver problemas con el propósito de encontrar el origen de dicha potencialidad para utilizarla computacionalmente. El origen de esta capacidad se encuentra en la evolución, producida por la selección natural, que favorece la perpetuación de los individuos más adaptados a su entorno.

Alan Turing reconoció, en 1950, que debe haber una conexión obvia entre el aprendizaje de máquina y la evolución y señaló que se podrían desarrollar programas para jugar ajedrez usando esta técnica. Campbell conjeturó en 1960 que en todos los procesos que llevan a la expansión del conocimiento, se involucra un proceso ciego de variación y supervivencia selectiva. Los primeros intentos de aplicar de manera formal la teoría de

la evolución aparecieron en las áreas de control de procesos estadísticos, aprendizaje de máquina y optimización de funciones. Tal vez el primer intento serio de este tipo se dió en el trabajo que realizaró Box (1957), en el desarrollo de una técnica que denominó *operación evolutiva*, la cual se aplicó a una planta de manufactura para manejarla y que se implantó sobre la base de votos de un comité de jefes técnicos. La calidad del producto avanzaba a través de mutaciones aleatorias y la selección era determinada por el comité. El trabajo de Bremermann (1958), se enfocó más a la optimización, introduciendo el manejo importante de un valor de aptitud y definiendo a un individuo como una cadena de símbolos binarios (unos y ceros). Bremermann advirtió, acertadamente, que la mutación jugaba un papel importante en la evolución, pues impedía el estancamiento en mínimos locales.

Fogel (1962) introdujo la primera técnica evolutiva que funcionó dentro de los lineamientos actuales de la computación evolutiva, así como los conceptos de población y selección. Su programación evolutiva consistía en hacer evolucionar autómatas de estados finitos por medio de mutaciones.

Otra técnica evolutiva dirigida particularmente a la optimización de funciones continuas de alta complejidad se desarrolló en Alemania, en 1965, por Rechenberg y Schwefel. Esta técnica, llamada *estrategia evolutiva*, se utilizó inicialmente para resolver problemas ingenieriles. La estrategia evolutiva utiliza a la mutación como operador principal, y en su versión más reciente usa el cruzamiento como operador secundario.

Aunque el australiano Fraser propuso, desde fines de los años 50, un procedimiento muy similar al que John Holland llamó planes evolutivos a fines de los años 60, es al segundo al que se le suele atribuir la creación de la técnica que se conoce como *algoritmo genético*, a raíz de que Holland (1975) publicara el libro *Adaptation in Natural and Artificial Systems*. La principal diferencia entre el algoritmo genético con las técnicas antes mencionadas, es que este último utiliza el cruzamiento como operador principal

y a la mutación como operador secundario e incluso es opcional.

En resumen, la computación evolutiva nace en el año de 1993 y, como se ve anteriormente, retoma conceptos de la evolución y genética para resolver, principalmente, problemas de optimización. Esta tiene sus raíces en tres desarrollos relacionados pero independientes entre sí:

- **Algoritmos géneticos**: Fueron desarrollados por John H. Holland en la década de 1960 y su motivación inicial fue la de proponer un modelo general del proceso de adaptación. En un algoritmo genético los elementos de las cadenas (genes) son típicamente bits, y en ciertos casos, valores numéricos.

- **Programación evolutiva**: Fue creada en la década de 1960 y su creador fue L. J. Fogel. Este desarrollo comenzó como un esfuerzo encaminado a crear inteligencia artificial basado en la evolución de máquinas de estados finitos.

- **Estrategias evolutivas**: Fueron propuestas por Ingo Rechenberg y Hans-Paul Schwefel en la década de 1970. Su principal objetivo era resolver problemas de optimización de parámetros.

Los algoritmos genéticos han demostrado ser una estrategia enormemente poderosa y exitosa para resolver problemas, demostrando el poder de los principios evolutivos. Se han utilizado algoritmos genéticos en una amplia variedad de campos para desarrollar soluciones a problemas tan difíciles o más difíciles que los abordados por los diseñadores humanos. Las soluciones que consiguen son a menudo más eficientes, más elegantes o incluso más complejas que las que una persona produciría. Además, la representación de las soluciones es más sencilla de entender que las de las otras dos técnicas.

II.5.1 Algoritmos genéticos

En la década de 1970, de la mano de John Holland surgió una de las líneas más prometedoras de la inteligencia artificial, la de los *algoritmos genéticos*. Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular.

Dado un problema específico a resolver, la entrada del GA es un conjunto de soluciones potenciales a ese problema, *codificadas* de alguna manera, y una métrica llamada *función de aptitud* que permite evaluar cuantitativamente a cada solución. Estas soluciones (*individuos*) se sabe que funcionan aunque no de manera óptima, con el objetivo de que el GA las mejore, las cuales se suelen generar aleatoriamente.

Después, el GA evalúa cada solución de acuerdo con la función de aptitud. En un conjunto de soluciones (*población*) generadas aleatoriamente, por supuesto, la mayoría no funcionarán en absoluto, y serán eliminadas. Sin embargo, por puro azar, unas pocas pueden ser prometedoras, pueden mostrar actividad, aunque sólo sea actividad débil e imperfecta, hacia la solución del problema.

Estas soluciones prometedoras se conservan y se les permite reproducirse (*cruzamiento*). Se realizan múltiples copias de ellas, pero las copias no son perfectas; se introducen cambios aleatorios (*mutación*) durante el proceso de copia. Luego, esta descendencia prosigue con la siguiente generación, formando un nuevo conjunto de soluciones candidatas, y son sometidas a una ronda de evaluación de aptitud. Las soluciones que han empeorado o no han mejorado con los cambios en su código son eliminadas; pero, de nuevo, por puro azar, las variaciones aleatorias introducidas en la población pueden haber mejorado a algunos individuos, convirtiéndolos en mejores soluciones del problema, más completas o más eficientes. De nuevo, se selecionan y copian estos individuos vencedores hacia la siguiente generación con cambios aleatorios, y el proceso se repite. Las expectativas son que la aptitud media de la población se incrementará en cada ronda y, por tanto, repitiendo este proceso cientos o miles de veces (*generaciones*),

pueden descubrirse soluciones muy buenas del problema.

Esquema general del algoritmo

A continuación se presenta el esquema general de un algoritmo genético (ver Figura 16):

1. Generar aleatoriamente una población de soluciones a un problema, representadas por una estructura de datos adecuada.
2. Evaluar cada una de las soluciones, y asignarle una puntuación o *aptitud* según lo bien que lo hayan hecho.
3. Seleccionar de la población las soluciones que tengan una aptitud mayor.
4. Cruzar y mutar las diferentes soluciones seleccionadas para reconstruir la población.
5. Repetir un número determinado de veces, o hasta que se haya encontrado la solución deseada.

Métodos de representación

Antes de que un GA pueda ponerse a trabajar en un problema, se necesita un método para codificar las soluciones potenciales del problema de forma que una computadora pueda procesarlas. Un enfoque común es codificar las soluciones como cadenas binarias: secuencias de 1s y 0s, donde el dígito de cada posición representa el valor de algún aspecto de la solución. Otro método similar consiste en codificar las soluciones como cadenas de enteros o números decimales, donde cada posición, de nuevo, representa algún aspecto particular de la solución. Este método permite una mayor precisión y complejidad que el método restringido de utilizar sólo números binarios, y a menudo

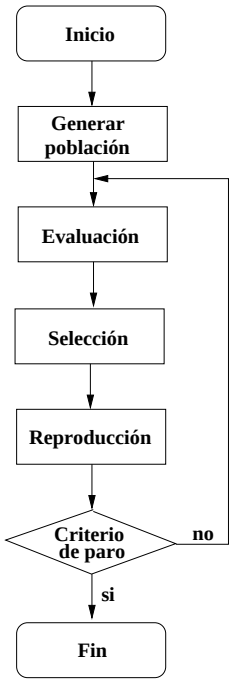


Figura 16: *Esquema general de un algoritmo genético.*

está, intuitivamente, más cerca del espacio de problemas. Un tercer método consiste en representar a los individuos de un GA como cadenas de letras, donde cada letra representa un aspecto específico de la solución.

La virtud de estos tres métodos es que facilitan la definición de operadores que causen los cambios aleatorios en las candidatas seleccionadas: cambiar un 0 por un 1 o viceversa, sumar o restar al valor de un número una cantidad elegida al azar, o cambiar una letra por otra.

Métodos de selección

Existen diferentes técnicas para seleccionar a los individuos que deben copiarse hacia la siguiente generación. Algunos de estos métodos son mutuamente exclusivos, pero otros pueden utilizarse en combinación. Los más usados son:

- **Elitista**: Se garantiza la selección de los miembros más aptos de cada generación.

La mayoría no utilizan elitismo puro, sino que usan una forma modificada en la que el mejor individuo, o algunos de los mejores, son copiados hacia la siguiente generación en caso de que no surja nada mejor.

- **Proporcional a la aptitud**: Los individuos más aptos tienen más probabilidad de ser seleccionados, pero no existe certeza.

- **Ruleta**: Una forma de selección proporcional a la aptitud en la que la probabilidad de que un individuo sea seleccionado es proporcional a la diferencia entre su aptitud y la de sus competidores.

- **Torneo**: Se eligen subgrupos de individuos de la población, y los miembros de cada subgrupo compiten entre ellos. Sólo se elige a un individuo de cada subgrupo para la reproducción.

- **Por rango**: A cada individuo de la población se le asigna un rango numérico basado en su aptitud, y la selección se basa en esta asignación, en lugar de las diferencias absolutas en aptitud. La ventaja de este método es que puede evitar que individuos muy aptos ganen dominancia al principio a expensas de los menos aptos, lo que reduciría la diversidad genética de la población y podría obstaculizar la búsqueda de una solución aceptable.

- **Generacional**: Sustituir todos los individuos de una generación para obtener la siguiente.

Métodos de cruzamiento

El cruzamiento implica seleccionar a dos individuos para que intercambien segmentos de su código, produciendo una *descendencia* cuyos individuos son combinaciones de sus padres. Las formas comunes de cruzamiento son:

- **En un punto**: Se establece un punto de intercambio en un lugar aleatorio del genoma de los dos individuos. El individuo a contribuye todo su código anterior a ese punto y el individuo b contribuye todo su código a partir de ese punto para producir una descendencia. Se repite el proceso intercambiando los individuos para crear a la segunda descendencia, ver Figura 17.

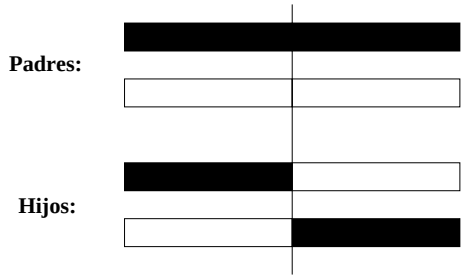


Figura 17: *Cruzamiento de un punto.*

- **En dos puntos**: Se establecen dos puntos de intercambio en dos lugares aleatorios del genoma de los individuos, intercambiando sus códigos para producir dos hijos, ver Figura 18.

Figura 18: *Cruzamiento de dos punto.*

- **Uniforme**: Se genera un individuo aleatorio de 1s y 0s, y se intercambian los

bits de los dos cromosomas que coincidan donde hay un 1 en el individuo. O bien se genera un número aleatorio para cada bit, y si supera una determinada probabilidad se intercambia ese bit entre los dos cromosomas.

Métodos de mutación

La mutación es un operador genético utilizado para mantener la diversidad genética de una población de individuos en el cambio de una generación a la siguiente. Es análogo a la mutación biológica.

El ejemplo clásico de un operador de mutación involucra una probabilidad donde un bit arbitrario en una secuencia genética será modificado de su estado original. Un método común para la implementación de un operador de mutación es generar una variable aleatoria para cada bit que conforma al individuo. Esta variable aleatoria indica si un bit en particular será o no modificado.

El propósito de la mutación es evitar los mínimos locales previniendo las poblaciones de individuos muy similares entre sí, lo cual vuelve lenta o incluso detiene la evolución.

Criterio de paro

El proceso generacional se repite hasta que se encuentre un criterio de paro. Los criterios de paro más comunes son:

- Se encuentra una solución que satistace un criterio mínimo.
- Se alcanza un número de generaciones dado.
- No se producen mejores individuos después de un número determinado de generaciones.
- Revisión manual.
- Combinaciones de las anteriores.

En resumen, estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y cruzamiento genético). Así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más aptos, que sobreviven, y cuáles, los menos aptos, son descartados.

Ventajas de los GAs

- La primera y más importante es que los algoritmos genéticos son intrínsecamente paralelos. La mayoría de los otros algoritmos son en serie y sólo pueden explorar el espacio de soluciones hacia una solución en una dirección al mismo tiempo, y si la solución que descubren resulta subóptima, no se puede hacer otra cosa que abandonar el trabajo hecho y empezar de nuevo. Sin embargo, ya que los GAs tienen descendencia múltiple, pueden explorar el espacio de soluciones en múltiples direcciones a la vez. Si un camino resulta no ser satisfactorio, puede ser eliminado fácilmente y continuar el trabajo en caminos más prometedores, dándoles una mayor probabilidad de encontrar la solución en cada ejecución.

- Debido al paralelismo que les permite evaluar implícitamente muchos esquemas a la vez, los algoritmos genéticos funcionan particularmente bien resolviendo problemas cuyo espacio de soluciones potenciales es realmente grande (demasiado vasto para hacer una búsqueda exhaustiva en un tiempo razonable). La mayoría de los problemas que caen dentro de esta categoría se conocen como *no lineales*. En un problema no lineal, la aptitud de cada componente es independiente, por lo que cualquier mejora en alguna parte dará como resultado una mejora en el sistema completo.

- Otra ventaja notable de los algoritmos genéticos es que se desenvuelven bien en problemas con un paisaje adaptativo complejo (aquellos en los que la función de

aptitud es discontinua, ruidosa, que cambia con el tiempo, o que tiene muchos óptimos locales).

- Otra área en la que destacan los algoritmos genéticos es en su habilidad para manipular muchos parámetros simultáneamente. Muchos problemas de la vida real no pueden definirse en términos de un único valor que hay que minimizar o maximizar, sino que deben expresarse en términos de múltiples objetivos, a menudo involucrando contrapartidas: uno sólo puede mejorar a expensas de otro. Los GAs son muy buenos resolviendo estos problemas: en particular, su uso del paralelismo les permite producir múltiples soluciones, igualmente buenas, al mismo problema, donde posiblemente una solución candidata optimiza un parámetro y otra candidata optimiza uno distinto, y luego un supervisor puede seleccionar una de esas candidatas para su utilización.

- Finalmente, una de las cualidades de los algoritmos genéticos que, a primera vista, puede parecer un desastre, resulta ser una de sus ventajas: a saber, los GAs no saben nada de los problemas que deben resolver. En lugar de utilizar información específica conocida *a priori* para guiar cada paso, realizan cambios aleatorios en sus soluciones candidatas y luego utilizan la función de aptitud para determinar si esos cambios producen una mejora.

II.6 Clasificación

Como se mencionó en la introducción, el problema de clasificación es mucho más amplio que el concerniente a la visión por computadora por no decir al reconocimiento de objetos. La idea básica en clasificación es reconocer objetos basado en características.

Las formas de clasificación de objetos son susceptibles a dividirse según el esquema de la Figura 19.

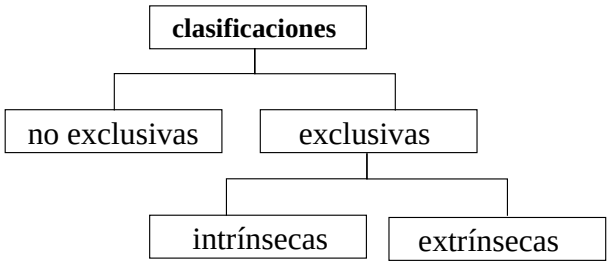


Figura 19: *Esquema de clasificación.*

- *No exclusivas*: Un mismo objeto puede pertenecer a varias clases.
- *Exclusivas*: Cada objeto pertenece solamente a una clase.
- *Intrínsecas (no supervisadas)*: la clasificación se realiza en base a las características propias de los objetos, sin conocimiento previo sobre las clases a las que pertenecen.
- *Extrínsecas (supervisadas)*: Las clases a las que pertenecen los objetos están predefinidas y se conocen ejemplos de cada una, o algunos de los objetos ya están clasificados y son utilizados por el mismo algoritmo para aprender a clasificar a los demás.

La clasificación utiliza habitualmente uno de los siguientes procedimientos: *clasificación estadística* (o teoría de decisión), *clasificación sintáctica* (o estructural). El reconocimiento estadístico de patrones está basado en las características de los patrones, asumiendo que han sido generados por un sistema probabilístico. El reconocimiento estructural de patrones está basado en las relaciones estructurales de las características.

Para la clasificación se puede utilizar un conjunto de aprendizaje, del cual ya se conoce la clasificación de la información *a priori* y se usa para entrenar al sistema, siendo la estrategia resultante conocida como *aprendizaje supervisado*. El aprendizaje

puede ser también *no supervisado*, donde el sistema no tiene un conjunto para aprender a clasificar la información *a priori*, sino que se basa en cálculos estadísticos para clasificar los patrones.

II.6.1 Clasificación estadística

La clasificación estadística es un procedimiento estadístico en el cual puntos individuales forman grupos basados en información cuantitativa de una o más características inherentes al punto y basados en un conjunto de entrenamiento de puntos previamente etiquetados.

Formalmente, el problema se puede definir como sigue:

Dado un conjunto de entrenamiento $\{(x_1, y), ...(x_n, y)\}$ *producir un clasificador* $h : X \rightarrow Y$ *el cual mapea un objeto* $x \in X$ *a su etiqueta de clasificación* $y \in Y$.

Los métodos de clasificación estadística resuelven uno de tres problemas matemáticos los cuales se describen a continuación.

El primer problema es encontrar un mapeo de un espacio de características, el cual es regularmente un vector multidimensional en el espacio de decisión, a un conjunto de etiquetas. Esto es equivalente a particionar el espacio de características en regiones, y después asignar una etiqueta a cada región.

El segundo problema es considerar a la clasificación como un problema de estimación, donde la meta es estimar una función de la forma

$$P(clase|\mathbf{x}) = f(\mathbf{x}; \theta) \quad , \tag{24}$$

donde la entrada del vector de características es $\mathbf{x}$, y la función f consta de algunos parámetros θ.

El tercer problema se encuentra relacionado al segundo, pero en este caso, el problema es estimar las probabilidades de la clase condicional $P(\mathbf{x}|clase)$ y luego obtener la probabilidad de la clase como en el segundo problema.

A continuación se definen los principales métodos de clasificación estadísticos usados más frecuentemente en la literatura (Russell y Norving, 2003; Jain *et al.*, 2000). Se hace la suposición de que las características de un objeto se pueden representar como un punto en el espacio de características N-dimensional definidas para la tarea de reconocimiento de objetos en particular.

Máquina de soporte vectorial

Con el fin de introducir el concepto de máquina de soporte vectorial (SVM, support vector machine) (Burgues, 1998; Hsu *et al.*, 2003) se explica el caso más sencillo, un clasificador de dos clases.

Supóngase que se quieren clasificar algunos puntos en dos clases. Estos puntos no se encuentran necesariamente en R^2, estos pueden ser puntos multidimensionales. Lo deseable es separarlos por medio de un *hiperplano* (una generalización del plano en tres o más dimensiones). Esta forma se conoce como clasificación lineal. También se desea escoger un hiperplano que separe los puntos con la máxima distancia a los puntos más cercanos de ambas clases. Dicha distancia se conoce como *margen*. Lo que se espera es que si se añade un punto a los puntos que ya se tienen, el porcentaje de clasificación del nuevo punto por medio de la separación entre las dos clases sea grande. Entonces, si tal hiperplano existe, a este se le conoce como *hiperplano de margen máximo* o *hiperplano óptimo* y los puntos(vectores) que se encuentran más cercanos a este hiperplano se conocen como *vectores de soporte*.

El uso del hiperplano de margen máximo está motivado en la teoría Vapnik Chervonenkis(CV) la cual provee una frontera de error de prueba probabilística que se minimiza

cuando el margen se maximiza.

1. Hiperplano de separación óptimo

Linealmente separable

En este primer caso, dadas l muestras de datos de entrenamiento $(x_1, y_1), ..., (x_l, y_l)$, $x \in R^d$, $y \in \{-1, 1\}$, estos se pueden separar por medio de un hiperplano de decisión

$$f(\mathbf{x}) = \mathbf{w}.\mathbf{x} + b \quad , \tag{25}$$

donde x es un vector de entrada, w es un vector n-dimensional de peso ajustable y b es un escalar (inclinación o tendencia). El vector de peso w define la dirección (gradiente) del hiperplano separador $f(x)$ y b (bias) define la distancia del hiperplano desde el origen.

Este hiperplano cumple

$$\mathbf{w}.x + b \geq 1 \qquad si\ y_i = 1 \quad , \tag{26}$$

$$\mathbf{w}.x + b \leq -1 \qquad si\ y_i = -1,\ i = 1, ..., l \quad , \tag{27}$$

o escrita de manera compacta

$$y_i[\mathbf{w}.x + b] \geq 1, \qquad i = 1, ..., l \quad . \tag{28}$$

Ahora, se tienen que maximizar el margen Δ de este hiperplano. Se define el margen Δ como la distancia mínima del hiperplano al punto más cercano. La distancia al hiperplano se expresa como:

$$d = y\frac{|f(x)|}{||w||} \quad , \tag{29}$$

y por lo tanto, si existe un margen Δ, todas las imágenes de entrenamiento cumplen:

$$y_i \frac{|f(x_i)|}{||w||} \geq \Delta, \qquad i = 1, ..., l \quad . \tag{30}$$

Entonces, el problema es encontrar el w que maximice Δ. Existen un número infinito de soluciones, que sólo difieren en la escala w. Con el fin de limitar el número de soluciones, únicamente se consideran las soluciones que estén normalizadas como sigue:

$$\Delta.||w|| = 1 \quad . \tag{31}$$

Estos hiperplanos se conocen como *hiperplanos canónicos*.

Maximizar el margen Δ equivale a minimizar $||w||$. Por lo tanto, el hiperplano óptimo es el que cumple con las restricciones dadas por la Ecuación (28) y minimiza la siguiente expresión:

$$z = \frac{1}{2}||w||^2 \quad . \tag{32}$$

El uso del factor $\frac{1}{2}$ y el exponente de la Ecuación (32) convienen para cálculos posteriores. Este es un problema de optimización con restricciones.

Para resolver este problema de optimización, se puede utilizar el método de multiplicadores de Lagrange para encontrar la solución óptima. Si $\lambda_i \geq 0$ son los multiplicadores de Lagrange, el problema de optimización con restricciones se puede escribir como un problema de optimización sin restricciones:

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2}(\mathbf{w}.\mathbf{w}) - \sum_{i=1}^{l} \lambda_i \{y_i[\mathbf{w}.\mathbf{x}_i + b] - 1\} \quad , \tag{33}$$

donde se tienen que encontrar los puntos de la función $L(\mathbf{w}, b, \lambda)$. La función se debe minimizar con respecto a $\mathbf{w}$ y b; y maximizar con respecto a $\lambda_i \geq 0$.

Consecutivamente, los parámetros de $\mathbf{w}$ y b se escriben en términos de λ_i como sigue. La solución en el punto $(\mathbf{w}^*, b^*, \lambda^*)$ debe cumplir las siguientes condiciones:

$$\frac{\delta L(\mathbf{w}^*, b^*, \lambda^*)}{\delta b} = 0 \quad , \tag{34}$$

$$\frac{\delta L(\mathbf{w}^*, b^*, \lambda^*)}{\delta \mathbf{w}} = 0 \quad . \tag{35}$$

Al resolver estas derivadas parciales se deben satisfacer los siguientes puntos:

1. Los multiplicadores de Lagrange deben cumplir:

$$\sum_{i=0}^{l} \lambda_i^* y_i = 0, \qquad \lambda_i^* \geq 0, i = 1, ..., l \quad . \tag{36}$$

2. El vector $\mathbf{w}^*$ es una combinación lineal de los vectores de entrenamiento:

$$\mathbf{w}^* = \sum_{i=0}^{l} \lambda_i^* y_i \mathbf{x}_i, \qquad \lambda_i^* \geq 0, i = 1, ..., l \quad . \tag{37}$$

El teorema Kuhn-Tucker dice que cualquier λ_i^* es diferente de cero si las restricciones correspondientes en la Ecuación (28) son iguales:

$$\lambda_i^* [y_i(\mathbf{w}.\mathbf{x}_i + b) - 1], \qquad i = 1, ..., l \quad . \tag{38}$$

Los $\mathbf{x}_i$ para los cuales $\lambda_i^* > 0$ son los *vectores de soporte*. Si se reescribe el Lagrangiano como:

$$L(\mathbf{w}, b, \lambda) = \frac{1}{2}(\mathbf{w}.\mathbf{w}) - \sum_{i=1}^{l} \lambda_i y_i \mathbf{x}_i.\mathbf{w} - b \sum_{i=1}^{l} \lambda_i y_i + \sum_{i=1}^{l} \lambda_i \quad , \tag{39}$$

se pueden sustituir (36) y (37) para expresar el Lagrangiano en función de λ_i:

$$L_{dual}(\lambda) = -\frac{1}{2} \sum_{i,j=1}^{l} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i.\mathbf{x}_j) + \sum_{i=1}^{l} \lambda_i \quad . \tag{40}$$

Esta expresión es la formulación dual del problema de optimización. Para resolver este problema dual de optimización, el sistema de la Ecuación (40) se debe maximizar con respecto a los parámetros $(\lambda_1, ..., \lambda_l)$ y sus restricciones:

$$\sum_{i=1}^{l} \lambda_i y_i = 0, \qquad \lambda_i \geq 0, i = 1, ..., l \quad . \tag{41}$$

Es más fácil resolver el problema de optimización dual que la primera formulación, debido a que no se necesitan manejar las restricciones de desigualdad de la Ecuación (28). La mayor ventaja es que los datos aparecen solo como parte de un producto interno y nunca individualmente. Por lo tanto, solo deben calcularse/conocerse los productos internos de los datos, mientras que la forma explícita de los datos pueden quedar implícitos. Este interesante hecho se explota en el método de los kérneles, el cual hace la generalización de las máquinas de soporte vectorial lineales, así como a otros métodos de aprendizaje. Esta generalización se aplica a un espectro amplio de posibles máquinas de soporte vectorial no lineales sin la complejidad computacional adicional.

Finalmente, la función de separación es:

$$f(\mathbf{x}) = \sum_{i=1}^{l} \lambda_i^* y_i (\mathbf{x}.\mathbf{x}_i) + b \quad . \tag{42}$$

Linealmente no separable

La separabilidad lineal no siempre puede ser utilizada. Esto es, incluso cuando los datos sean linealmente separables, algunas veces se puede obtener un margen más grande si algunos de los puntos caen dentro del margen.

Con el fin de generalizar los datos que caen dentro del margen, o que están del lado equivocado de la frontera de decisión, se añaden variables de error positivas $\xi_1, ..., \xi_l$ que suavizan las restricciones de la Ecuación (28) como sigue:

$$y_i[\mathbf{w}.\mathbf{x}_i + b] \geq 1 - \xi_i\,, \qquad \xi_i \geq 0 \quad . \tag{43}$$

Con el fin de encontrar el hiperplano que tenga el margen máximo en este caso, nuevamente se empieza minimizando la Ecuación (32) donde se añade como costo adicional la suma de las variables

$$\frac{1}{2}||\mathbf{w}||^2 + \frac{C}{l}\sum_{i=1}^{l}\xi_i \quad , \tag{44}$$

sujeto a las restricciones de la Ecuación (43). El parámetro C cuantifica el intercambio entre complejidad de las funciones de aprendizaje y la proporción de los ejemplos de entrenamiento permitidos intra-margen o no separables.

Este problema de optimización se puede convertir también a su forma dual. El Lagrangiano dual a maximizarse es:

$$L_{dual}(\lambda) = -\frac{1}{2}\sum_{i,j=1}^{l}\lambda_i\lambda_j y_i y_j(\mathbf{x}_i.\mathbf{x}_j) + \sum_{i=1}^{l}\lambda_i \quad , \tag{45}$$

sujeto a las restricciones:

$$\sum_{i=1}^{l} y_i\lambda_i = 0, \qquad 0 \leq \lambda_i \leq \frac{C}{l}, i = 1, ..., l \quad . \tag{46}$$

2. Máquinas de soporte vectorial basadas en kernel

Supóngase que se mapean los datos $\mathbf{x}$ a un espacio de mayor dimensión H como sigue:

$$R^d \mapsto H \quad ,$$
$$\mathbf{x} \mapsto \phi(\mathbf{x}) \quad .$$

El espacio de mayor dimensión H es, formalmente, un espacio de Hilbert el cual puede

ser de dimensión infinita. En el caso de dimensión finita es suficiente con considerar a H como un espacio Euclidiano.

Ahora, supongamos que existe un kernel tal que:

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}).\phi(\mathbf{y}) \quad ,$$

si se construye un hiperplano en el mapeo implícito al espacio de características de mayor dimensión, este hiperplano corresponde a una función no lineal en el espacio original.

Concretamente, se sustituyen los productos internos por el mapeo no lineal deseado del kernel K como sigue:

$$\mathbf{x}.\mathbf{y} \longleftrightarrow K(\mathbf{x}, \mathbf{y}) \quad . \tag{47}$$

Es decir, los Lagrangianos duales necesitan maximizarse. En el caso de separabilidad en el espacio de características de mayor dimensión tenemos:

$$L_{dual}(\lambda) = -\frac{1}{2}\sum_{i,j=1}^{l} \lambda_i \lambda_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{l} \lambda_i \quad , \tag{48}$$

con restricciones:

$$\sum_{i=1}^{l} \lambda_i y_i = 0, \qquad \lambda_i \geq 0, i = 1, ..., l \quad . \tag{49}$$

Elección del kernel

Los kérneles seleccionados con mayor frecuencia se definen como sigue: para un sistema *polinomial*, se elige el siguiente kernel

$$K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}.\mathbf{x}_i + 1)^p \quad , \tag{50}$$

donde su función de decisión tiene la siguiente forma:

$$f(\mathbf{x}) = sign(\sum_{vectores\,de\,soporte} y_i \lambda_i (\mathbf{x}_i.\mathbf{x} + 1)^p - b) \quad . \tag{51}$$

Con el fin de obtener redes con *Función en Base Radial* (RBF, Radial Basis Function) se utilizan las funciones de decisión:

$$f(\mathbf{x}) = sign(\sum_{vectores\,de\,soporte} y_i K_\gamma(|\mathbf{x} - \mathbf{x}_i|) - b) \quad , \tag{52}$$

y el kernel tiene la siguiente forma:

$$K_\gamma(|\mathbf{x} - \mathbf{x}_i|) = exp\{-\gamma|\mathbf{x} - \mathbf{x}_i|^2\} \quad . \tag{53}$$

El número de vectores de soporte corresponden al número de kérneles RBF utilizados para construir esta red. Un enfoque similar para construir redes neuronales de dos capas es elegir un kernel *sigmoidal*:

$$K(\mathbf{x}, \mathbf{x}_i) = tanh(v(\mathbf{x}.\mathbf{x}_i) + c) \quad , \tag{54}$$

con su correspondiente función de separación:

$$f(\mathbf{x}) = sign(\sum_{vectores\,de\,soporte} \lambda_i tanh(v(\mathbf{x}.\mathbf{x}_i) + c) - b) \tag{55}$$

De nuevo, el número de vectores de soporte correponde al número de neuronas de la primera capa que requiere la red para una generalización óptima. Los pesos de las neuronas de la primer capa son los vectores de soporte $\mathbf{x}_i$, mientras que los pesos de las neuronas de la segunda capa están dados por λ_i.

3. Máquinas de soporte vectorial multiclase

Los sistemas de reconocimiento de patrones multiclase se pueden obtener combinando SVMs de 2-clases. Existen tres métodos para hacer esto. El primer método es el esquema $1 - contra - k$ en el cual se entrenan $k - 1$ clasificadores de 2-clases. Cada máquina se entrena como clasificador para una clase en contra de las clases restantes. El segundo método es el esquema $1 - contra - 1$ para construir un clasificador multiclase; esto es, se construyen $\frac{k(k-1)}{2}$ máquinas de 2-clases. Cada máquina se entrena como clasificador de una clase en contra de otra clase. Con el fin de clasificar los datos de prueba, se realiza la competencia en parejas entre todas las máquinas y, de manera análoga a un torneo de tenis, cada ganador compite contra otro ganador hasta que queda un solo ganador. El ganador final determina la clase de los datos de prueba.

Finalmente, se puede resolver directamente un SVM con $k - clases$ modificando el problema de optimización expresado por la Ecuación (44) a fin de optimizar los márgenes de k hiperplanos $\mathbf{w}_k$ al mismo tiempo. Si se tienen l ejemplos de aprendizaje $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l)$, donde y_i indica una de las k posibles clases:

$$\frac{1}{2}\sum_{m=1}^{k}||\mathbf{w}_m||^2 + \frac{C}{l}\sum_{i=1}^{l}\sum_{m\neq y_i}^{k}\xi_i^m \quad , \tag{56}$$

con restricciones:

$$(\mathbf{w}_{y_i}.\mathbf{x}_i) + b_{y_i} \geq (\mathbf{w}_m.\mathbf{x}_i) + b_m + 2 - \xi_i^m \quad , \tag{57}$$

tal que

$$\xi_i^m \geq 0, \qquad i = 1, ..., l \; m \in \{1, ..., k\} \quad , \tag{58}$$

los cuales proporcionan la solución de la función de decisión

$$f(\mathbf{x}) = arg\,max_k[(\mathbf{w}_i.\mathbf{x}) + b_i], \qquad i = 1, ..., k \quad . \tag{59}$$

De manera similar, si se deriva el Lagrangiano se encontrará su forma dual sustituyendo las condiciones de Kuhn-Tucker. En la práctica, el último método tiene un uso limitado debido a la enorme cantidad de problemas de optimización que se tienen que resolver (el número de variables desconocidas se incrementa linealmente con el número de clases). En suma, este método no tiene un mejor desempeño comparado con los dos primeros.

Método de los vecinos más cercanos

El método de vecinos más cercanos, propuesto por Cover y Hart (1967) se ha convertido en un enfoque no-paramétrico muy eficiente para clasificación. Debido a su teoría mátematica perfecta, este método se puede aplicar con distintas variaciones. El clasificador se convierte en un clasificador Bayesiano si se tienen muchos puntos en la muestra. Pero en la práctica, dada una muestra pequeña, el clasificador Bayesiano falla en la estimación del error de Bayes, especialmente en espacios de mayor dimensión.

Es el método más sencillo de todos los métodos de clasificación, el cual trabaja de la siguiente manera. Se memorizan todos los puntos (datos) de entrenamiento. Con el fin de clasificar un nuevo punto, se calcula la distancia al punto de entrenamiento más cercano. La clase del punto más cercano determina la clasificación del nuevo punto. Regularmente, se utiliza como medida la distancia Euclidiana.

De manera formal, la clasificación de un nuevo vector $\mathbf{x}$, dado un conjunto de entrenamiento $(x^\mu, c^\mu), \mu = 1, ..., P$, se realiza de la siguiente manera:

1. Se calcula la disimilitud del punto de prueba $\mathbf{x}$ a cada uno de los puntos almacenados, $d^\mu = d(\mathbf{x}, \mathbf{x}^\mu)$.

2. Se encuentra el punto de entrenamiento $\mathbf{x}^{\mu^*}$ el cual es el más cercano a $\mathbf{x}$. Esto se realiza encontrando μ^* tal que $d^{\mu^*} < d_\mu$ para toda $\mu = 1, ..., P$.

3. Se asigna la etiqueta de la clase $c(\mathbf{x}) = c^{\mu^*}$.

donde:

c es el número de etiquetas de clases,

P es el conjunto de puntos de entrenamiento,

D es un conjunto de datos de P, tal que $D = \mathbf{x}^\mu, c^\mu$ con $\mu = 1, ..., P$ y

$d(\mathbf{x}, \mathbf{y})$ es una función que mide la disimilitud entre los puntos $\mathbf{x}$ y $\mathbf{y}$.

En caso de existir dos o más puntos equidistantes con diferentes etiquetas de clase, se elige la clase que contega mayor número de puntos. Si no existe este tipo de clase, se utiliza el método k-vecinos más cercanos.

El algoritmo de vecinos más cercanos es muy sencillo, poderoso y, por lo mismo, utilizado en distintas aplicaciones. Sin embargo, hay que tomar en cuenta una consideración: ¿Cómo se debe medir la distancia entre los puntos? Regularmente se utiliza la distancia Euclidiana, como se mencionó anteriormente, o alguna de las distancias que a continuación se listan, dependiendo de la naturaleza de los datos.

- **Euclidiana**: Es la distancia en línea recta entre un punto $\mathbf{x}$ y un punto $\mathbf{y}$.

$$d(\mathbf{x}, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- **Euclidiana cuadrada**: Es igual que la distancia Euclidiana pero no toma en cuenta la raíz cuadrada Como resultado, el cálculo de esta distancia es más rápido que el anterior.

$$d(\mathbf{x}, y) = \sum_{i=1}^{n}(x_i - y_i)^2$$

- **Manhattan**: Esta se calcula viajando de un punto a otro siguiendo el camino de una cuadrícula, es decir, no se permite el viaje en diagonal.

$$d(\mathbf{x}, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- **Chebichev**: Es la distancia máxima entre dos puntos en cualquiera de sus dimensiones. Es apropiada si la diferencia entre los puntos se refleja más en la diferencia de las dimensiones individuales que considerando todas las dimensiones.

$$d(\mathbf{x}, y) = max_i|\mathbf{x}_i - y_i|$$

entre otras.

En la versión más sencilla del algoritmo, explicado anteriormente, es necesario almacenar el conjunto de datos completo para realizar la clasificación. Sin embargo, en general, solo un subconjunto de los datos de entrenamiento determinará la frontera de decisión

Un "outlier" es un punto que se encuentra alejado del resto de los puntos el cual tiene una etiqueta errónea. Si los nuevos puntos se encuentran cercanos a un "outlier", se les asignará la etiqueta de este último. El método de k-vecinos más cercanos es una manera más robusta de clasificar los nuevos puntos al buscar en más que un vecino más cercano.

k-vecinos más cercanos

Como el nombre lo sugiere, la idea es incluir más de un vecino en la decisión acerca de la clase del nuevo punto $\mathbf{x}$. Supóngase que se utiliza la distancia Euclidiana como medida de similitud y se tiene una hiperesfera centrada en el punto $\mathbf{x}$ con radio r. Si se incrementa el radio r hasta que la hiperesfera contenga exactamente k puntos, la clase $c(\mathbf{x})$ estará dada por la clase que contenga más puntos dentro de la hiperesfera.

Si k es muy grande, la clasificación se realizará como en el caso más sencillo (tomando en cuenta todos los datos de entrenamiento). Por lo tanto, se debe elegir $k > 1$ pero sin llegar a tener $k = P$. Lo mejor es encontrar un valor óptimo para k. Con óptimo se refiere a un valor de k el cual dé como resultado el mejor desempeño del clasificador.

k vecinos más cercanos con distancia ponderada

Una mejora al método de k-vecinos más cercanos es ponderar la contribución de cada vecino de acuerdo a la distancia entre él y el nuevo punto a ser clasificado, dando mayor peso a los vecinos más cercanos.

Una vez que se incorpora la distancia ponderada, no hay ningún riesgo en permitir a todos los puntos de entrenamiento que contribuyan en la clasificación del nuevo punto, porque los puntos que se encuentran muy distantes, tendrán muy poco efecto. La única desventaja en considerar todos los puntos es que el clasificador será más lento.

Este método es muy efectivo en muchos problemas prácticos. Es robusto ante el ruido en los datos y muy efectivo cuando se le provee con conjuntos de entrenamiento lo suficientemente grandes. Obsérvese que al tomar el promedio ponderado de los k-vecinos más cercanos, el algoritmo puede evitar el impacto de "outliers".

Método Bayesiano

En esta sección se aborda el método Bayesiano para aprendizaje no supervisado (Duda *et al.*, 2001). Para esto, se asume que θ es una variable aleatoria con una distribución conocida *a priori* $p(\theta)$, y se usan los ejemplos de entrenamiento para calcular la densidad $p(\theta, D)$.

Las suposiciones básicas son:

1. Se conoce el número de clases c.

2. Se conocen las probabilidades *a priori* $p(w_j)$ para cada clase, $j = 1, ..., c$.

3. Se conoce la forma de las densidades de probabilidad $p(\mathbf{x}|w_j, \theta_j)$, pero se desconoce el vector de parámetros $\theta = (\theta_1, ..., \theta_c)^t$.

4. Parte del conocimiento acerca de θ se encuentra en una densidad conocida *a priori* $p(\theta)$.

5. El resto del conocimiento acerca de θ se encuentra en un conjunto D de n muestras $\mathbf{x}_1, ..., \mathbf{x}_n$ tomada independientemente de la densidad.

$$p(\mathbf{x}|\theta) = \sum_{j=1}^{c} p(\mathbf{x}|w_j, \theta_j)p(w_j) \quad . \tag{60}$$

En este punto, se podría ir directamente al cálculo de $p(\theta, D)$. Sin embargo, primero veremos como se utiliza esta densidad para determinar el clasificador Bayesiano. Supóngase que se selecciona un estado de la naturaleza con probabilidad $p(w_i)$ y se selecciona un vector de características $\mathbf{x}$ de acuerdo a la ley de probabilidad $p(\mathbf{x}|w_i, \theta_i)$. Para derivar el clasificador Bayesiano se debe utilizar toda la información disponible para calcular la probabilidad $p(w_i|\mathbf{x})$. Por la fórmula de Bayes tenemos que

$$p(w_i|\mathbf{x}, D) = \frac{p(\mathbf{x}|w_i, D)p(w_i, D)}{\sum_{j=1}^{c} p(\mathbf{x}|w_j, D)p(w_j, D)} \quad . \tag{61}$$

Debido a que la selección del estado de la naturaleza w_i se hizo independientemente de las muestras tomadas previamente, se obtiene

$$p(w_i|\mathbf{x}, D) = \frac{p(\mathbf{x}|w_i, D)p(w_i)}{\sum_{j=1}^{c} p(\mathbf{x}|w_j, D)p(w_j)} \quad . \tag{62}$$

Dentro del método Bayesiano se encuentra la introducción del vector desconocido θ a través de:

$$p(\mathbf{x}|w_i, D) = \int p(\mathbf{x}, \theta|w_i, D)d\theta = \int p(\mathbf{x}, \theta|w_i, D)p(\theta|w_i, D)d\theta \quad . \tag{63}$$

Debido a que la selección de $\mathbf{x}$ es independiente de las muestras, se tiene $p(\mathbf{x}|\theta, w_i, D) = p(\mathbf{x}|w_i, \theta_i)$. De manera similar, por el conocimiento del estado de la naturaleza cuando se selecciona $\mathbf{x}$ se tiene $p(\theta|w_i, D) = p(\theta, D)$ y entonces

$$p(\mathbf{x}|w_i, D) = \int p(\mathbf{x}|w_i, \theta_i)p(\theta|D)d\theta \quad . \tag{64}$$

Esto es, la mejor estimación de $p(\mathbf{x}|w_i)$ se obtiene promediando $p(\mathbf{x}|w_i, \theta_i)$ sobre θ_i. La buena o mala estimación depende de la naturaleza de $p(\theta, D)$.

Aprendizaje del vector de parámetros

Se puede utilizar la fórmula de Bayes para escribir

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)d\theta} \quad , \tag{65}$$

donde la independencia de las muestras produce la probabilidad

$$p(D|\theta) = \prod_{k=1}^{n} p(\mathbf{x}_k|\theta) \quad . \tag{66}$$

Alternativamente, sea D^n el conjunto de n muestras, se puede escribir la expresión (65) de manera recursiva como:

$$p(\theta|D^n) = \frac{p(\mathbf{x}_n|\theta)p(\theta|D^{n-1})}{\int p(\mathbf{x}_n|\theta)p(\theta|D^{n-1})d\theta} \quad . \tag{67}$$

Estas son las fórmulas básicas para el aprendizaje Bayesiano no supervisado. La Ecuación (65) enfatiza la relación entre las soluciones Bayesiana y de máxima probabilidad. Si $p(\theta)$ es uniforme sobre la región donde $p(D|\theta)$ se maximiza; entonces $p(\theta|D)$ se maximiza en el mismo lugar. Si solo ocurre un pico significativo cuando $\theta = \hat{\theta}$ y el pico es puntiagudo, las Ecuaciones (62) y (64) producen

$$p(\mathbf{x}|w_i, D) \simeq p(\mathbf{x}|w_i, \hat{\theta}) \quad , \tag{68}$$

y

$$p(w_i|\mathbf{x}, D) \simeq \frac{p(\mathbf{x}|w_i, \hat{\theta}_i)P(w_i)}{\sum_{j=1}^{c} p(\mathbf{x}|w_j, \hat{\theta}_j)P(w_j)} \quad . \tag{69}$$

Es decir, estas condiciones justifican el uso de la estimación de máxima probabilidad $\hat{\theta}$ como si este fuera el valor verdadero de θ en el diseño del clasificador Bayesiano.

Redes Neuronales

Las redes neuronales están compuestas de *nodos* ligados por conexiones directas, ver Figura 20. Una *conexión* del nodo j al nodo i sirve para propagar la *activación* a_i de j a i.

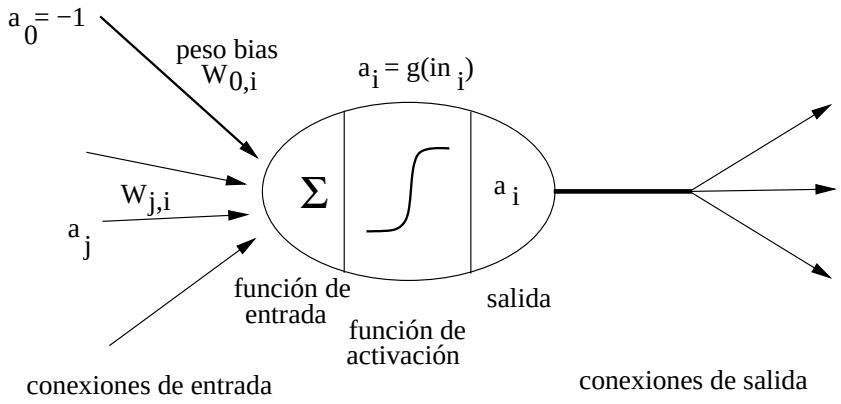


Figura 20: *Modelo matemático de una neurona. La activación de las unidades de salida es* $a_i = g(\sum_{j=0}^{n}, W_{j,i}a_j)$*, donde* a_j *es la activación de salida del nodo* j *y* $W_{j,i}$ *es el peso de la conexión del nodo* j *en este nodo.*

Cada conexión tiene un *peso* numérico asociado $W_{j,i}$, el cual determina la fuerza y signo de la conexión. Cada nodo i calcula la suma de pesos de sus entradas:

$$in_i = \sum_{j=0}^{n} W_{j,i} a_j \quad . \tag{70}$$

Después se aplica una *función de activación* g a esta suma para obtener la salida

$$a_i = g(in_i) = g(\sum_{j=0}^{n} W_{j,i} a_j) \quad . \tag{71}$$

La función de activación g está diseñada para conocer dos cosas. Primero, el nodo estará *activado* (cercano a +1) cuando las entradas dadas sean correctas, e *inactivo* (cercano a 0) cuando las entradas dadas sean incorrectas. Segundo, la activación debe ser no lineal, de otro modo la red neuronal se convierte en una simple función lineal. Existen dos opciones para g como se muestra en la Figura 21 dadas por la función *umbral* y la función *sigmoidal* (también conocida como función logística). La función sigmoidal tiene la ventaja de ser diferenciable; lo cual, como se verá más adelante, es importante para el algoritmo de aprendizaje de pesos. Ambas funciones tienen un umbral en cero; el peso bias $W_{0,i}$ establece el umbral actual para el nodo, en el sentido de que el nodo se activa cuando la suma de los pesos $\sum_{j=1}^{n} W_{j,i} a_j$ (excluyendo al bias de la entrada) excede $W_{0,i}$.

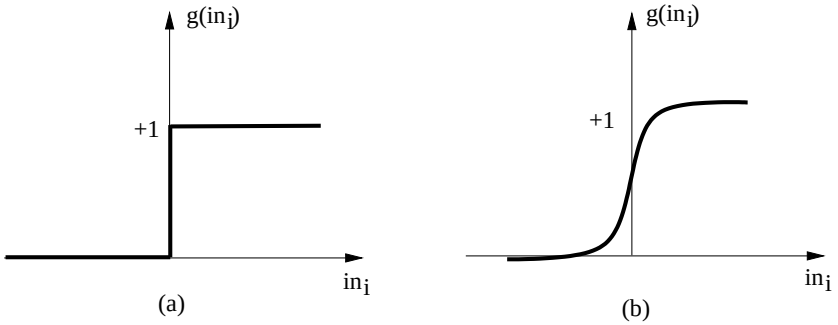


Figura 21: (*a*) *La función de activación* ***umbral****, cuya salida es* 1 *cuando la entrada es positiva y* 0 *en otro caso.* (*b*) *La función* ***sigmoidal*** $\frac{1}{1+e^{-x}}$.

Estructuras de la red

Existen dos categorías principales de estructuras de redes neuronales: la acíclica o *redes de retroalimentación* y las cíclicas o *redes recurrentes*. Una red de retroalimentación se representa en función de su entrada actual; esto es, no tiene otros estados más que los pesos por sí mismos. Las redes recurrentes obtienen sus salidas volviendo a sus propias entradas. Esto significa que la respuesta de la red a una entrada dada depende de su estado inicial, el cual depende de sus entradas previas. Por esto, las redes recurrentes pueden tener memoria a corto plazo. Esto las hace más interesantes como modelos del cerebro pero también vuelve más difícil su comprensión. En esta sección se ahonda en las redes de retroalimentación.

Para entender la suposición de que las redes de retroalimentación se representan en función de sus entradas, considérese la red que se muestra en la Figura 22, la cual tiene dos nodos de entrada, dos nodos *ocultos* y dos nodos de salida (se omite el nodo bias a fin de tener un ejemplo más sencillo).

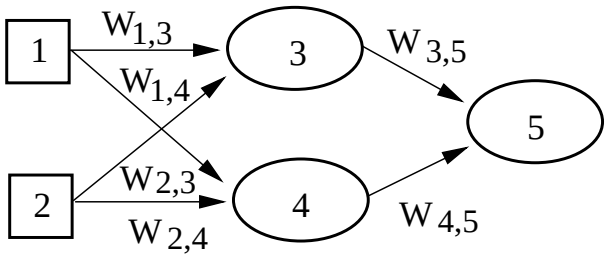


Figura 22: *Una red neuronal con dos entradas, una capa oculta de dos nodos, y una salida.*

Dado un vector de entrada $x = (x_1, x_2)$, la activación de los nodos de entrada son $(a_1, a_2) = (x_1, x_2)$, entonces la red se calcula como sigue:

Es decir, expresando la salida de cada nodo oculto en función de su entrada, se muestra

$$\begin{aligned} a_5 &= g(W_{3,5}a_3 + W_{4,5}a_4) \\ &= g(W_{3,5}g(W_{1,3}a_1) + W_{2,3}a_2) + W_{4,5}g(W_{1,4}a_1 + W_{2,4}a_2) \quad . \end{aligned}$$

que la salida de la red como un todo, a_5, está en función de las entradas de la red. Además, los pesos en la red actúan como parámetros de esta función. Al ajustar los pesos se cambia la función que representa a la red. Esta es la manera en que ocurre el aprendizaje en las redes neuronales.

Las redes de retroalimentación usualmente se ordenan en *capas*, de tal manera que cada nodo recibe la entrada solo de los nodos de la capa inmediatamente anterior. A continuación se profundiza en redes con una sola capa, las cuales no tienen nodos ocultos, y en redes multicapa, las cuales tienen una o más capas de nodos ocultos.

Redes neuronales de retroalimentación de una capa (perceptrones)

Una red con todos sus nodos de entrada conectados directamente a los nodos de salida se conoce como *red neuronal de una capa*, o *red perceptrón*. Dado que cada nodo de salida es independiente de los otros, cada peso afecta sólo a un nodo de salida.

Observando la Ecuación (71) vemos que el perceptrón umbral regresa 1 si y solo si la suma de los pesos de sus entradas es positiva:

$$\sum_{j=0}^{n} W_j x_j > 0 \qquad \acute{o} \qquad \mathbf{W}.\mathbf{x} > 0 \quad . \tag{72}$$

Ahora, la formula $\mathbf{W}.\mathbf{x} = 0$ define un hiperplano en el espacio de entrada, así que el perceptrón regresa 1 si y solo si la entrada se encuentra a uno de los lados del hiperplano. Por esta razón, el perceptrón umbral se conoce como *separador lineal*. En general, los perceptrones umbral pueden representar solamente funciones linealmente separables.

Esto constituye una pequeña fracción de todas las funciones.

A pesar de sus limitaciones, los perceptrones umbral tienen algunas ventajas. En particular, existe un algoritmo de aprendizaje simple que encajará en un perceptrón umbral para cualquier conjunto linealmente separable.

La idea detrás de este algoritmo, y detrás de la mayoría de los algoritmos de aprendizaje de redes neuronales, es ajustar los pesos de la red a fin de minimizar alguna medida de error sobre el conjunto de entrenamiento. Es decir, el aprendizaje se formula como una búsqueda de optimización en el espacio de pesos. La medida "clásica" de error es la *suma de los errores cuadrados*. El error de un ejemplo de entrenamiento con entrada $\mathbf{x}$ y salida verdadera $\mathbf{y}$ se escribe como:

$$E = \frac{1}{2}Err^2 = \frac{1}{2}(\mathbf{y} - h_{\mathbf{w}}(\mathbf{x}))^2 \quad , \tag{73}$$

donde $h_{\mathbf{w}}(\mathbf{x})$ es la salida del perceptrón del ejemplo.

Se puede utilizar el descenso del gradiente para reducir el error calculando las derivadas parciales de E con respecto a cada peso de la siguiente manera:

$$\begin{aligned} \frac{\partial E}{\partial W_j} &= Err \times \frac{\partial Err}{\partial W_j} \\ &= Err \times \frac{\partial}{\partial W_j}(y - g(\sum_{j=0}^{n} W_j x_j)) \\ &= -Err \times g'(in) \times x_j \quad , \end{aligned}$$

donde g' es la derivada de la función de activación. En el algoritmo de descenso del gradiente se pretende reducir E modificando los pesos de la siguiente manera

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j \quad , \tag{74}$$

donde α es la *tasa de aprendizaje*, la cual tiene mucho sentido. Si el error $Err = y - h_w(x)$ es positivo entonces la salida de la red será tan pequeña que los pesos se incrementarán para entradas correctas y disminuirán para entradas incorrectas. Sucede lo opuesto cuando el error es negativo.

El algoritmo completo se muestra en la Tabla II. Este corre los ejemplos de entrenamiento uno a la vez a través de la red, ajustando los pesos después de cada ejemplo para reducir el error. A cada ciclo a través de los ejemplos se le conoce como *época*. Las épocas se repiten hasta que se cumpla algún criterio de paro (regularmente, cuando el cambio en los pesos es muy pequeño).

Tabla II: Algoritmo de aprendizaje descenso de gradiente para perceptrones.

función APRENDIZAJE-PERCEPTRÓN (*ejemplos, red*) **regresa** una hipótesis de red.
  entradas:
    ejemplos, un conjunto de ejemplos, con entrada $\mathbf{x} = x_1, ..., x_n$
      y salida y.
    red, un perceptrón con pesos W_j, $j = 0, ..., n$, y una función de activación g.

  repetir
    para cada e en $ejemplos$ **hacer**
      $in \leftarrow sum_{j=0}^{n} W_j x_{j[e]}$
      $Err \leftarrow y[e] - g)in)$
      $W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j[e]$
  hasta cumplir un criterio de paro.
  regresa HIPÓTESIS DE RED NEURONAL (*red*)

En el algoritmo de aprendizaje descenso de gradiente para perceptrones se supone una función de activación diferenciable g. Para los perceptrones umbral, se omite el factor $g'(in)$ en la modificación de los pesos.

Redes neuronales de retroalimentación multicapa

Ahora consideremos las redes con nodos ocultos. El caso más común es la red que tiene una sola capa oculta, como se ve en la Figura 23. La ventaja de añadir capas ocultas es que se aumenta el espacio de búsqueda que la red puede tener. De hecho, con una sola capa oculta pero suficientemente grande, es posible representar cualquier función continua de las entradas con una precisión arbitraria; además, con dos capas se pueden representar funciones discontinuas. Desafortunadamente, es difícil caracterizar exactamente cuáles funciones se pueden representar y cuáles no a partir de una estructura de red en particular.

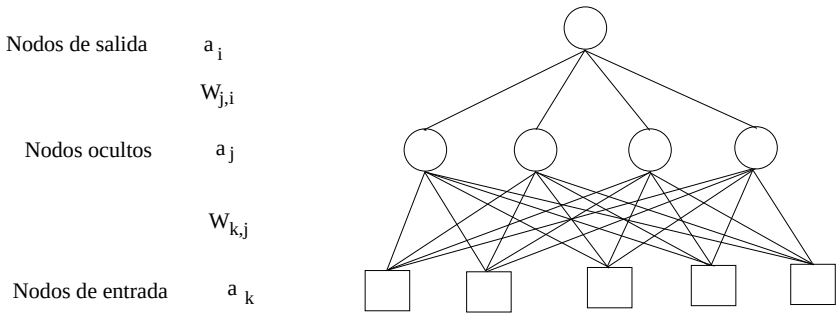


Figura 23: *Una red neuronal multicapa con una capa oculta y 5 nodos de entrada.*

Los algoritmos de aprendizaje para redes multicapa son similares al algoritmo de aprendizaje del perceptrón mostrado en la Tabla II. Una diferencia pequeña es que se pueden tener diferentes salidas, así que se tiene un vector de salida $h_w(x)$ en vez de un solo valor, y cada ejemplo tiene un vector de salida y. Una diferencia grande es que el error $y - h_w$ en la capa de salida hace parecer que el error de las capas ocultas luzca "misterioso" debido a que los datos de entrenamiento no dicen qué valores deberían tener los nodos ocultos. Esto se debe a que se *retropropaga* el error de la capa de salida

a las capas ocultas.

En la capa de salida, la regla de modificación de pesos es igual a la Ecuación (74). Sea Err_i el *i-ésimo* componente del vector de error $y - h_w$. Entonces se define un error modificado $\Delta_i = Err_i \times g'(in_i)$, de tal manera que la regla de modificación de pesos se convierte en

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i \quad . \tag{75}$$

Para modificar las conexiones entre los nodos de entrada y los nodos ocultos, es necesario definir una cantidad análoga al término de error que se aplica a los nodos de salida. Aquí es donde se hace uso de la retropropagación del error. La idea es que el nodo oculto j es "responsable" en alguna medida del error Δ_i en cada uno de los nodos de salida a los cuales está conectado. Esto es, los valores de Δ_i se dividen de acuerdo a la fuerza de la conexión entre el nodo oculto y el nodo de salida, y este se propaga hacia atrás para proveer los valores Δ_j para la capa oculta. La regla de propagación de los valores de Δ es la siguiente:

$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i \quad . \tag{76}$$

Ahora, la regla de modificación para los pesos entre las capas de entrada y las ocultas es casi idéntica a la regla de modificación de la capa de salida:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j \quad . \tag{77}$$

El proceso de retropropagación se resume como sigue:

- Calcular los valores Δ para los nodos de salida utilizando el error observado.
- Empezando con la capa de salida, se repiten los siguientes pasos para cada capa en la red, hasta que se alcance la primer capa oculta:

- Propagar los valores de Δ hacia la capa previa.

- Modificar los pesos entre las dos capas.

El algoritmo se muestra en la Tabla III.

Tabla III: Algoritmo de de retropropagación para aprendizaje en redes multicapa.

función APRENDIZAJE-RETROPROPAGACIÓN (*ejemplos, red*) **regresa** una red neuronal.
  entradas:
    ejemplos: un conjunto de ejemplos con vector de entrada **x**
      y vector de salida **y**.
    red: una red multicapa con L capas, pesos $W_{j,i}$, y una función de activación g.

  repetir
    para cada e **en** *ejemplos* **hacer**
      para cada nodo j en la capa de entrada **hacer** $a_j \leftarrow x_j[e]$
      para $l = 2$ **hasta** L **hacer**
        $in \leftarrow \sum_j W_{j,i} a_j$
        $a_i \leftarrow g(in_i)$
      para cada nodo i en la capa de salida **hacer**
        $\Delta_i \leftarrow g'(in_i) \times (y_i[e] - a_i)$
      para $l = L - 1$ hasta l **hacer**
        para cada nodo j en la capa l **hacer**
          $\Delta_j \leftarrow g'(in_j) \sum_i W_(j,i) \Delta_i$
          para cada nodo i en la capa $l + 1$ **hacer**
            $W_{j,i} \leftarrow W_{j_i} + \alpha \times a_j \times \Delta_i$
  hasta cumplir un criterio de paro.
  regresa HIPÓTESIS DE RED NEURONAL (*red*)

Matemáticamente, el error de un solo ejemplo se define como:

$$E = \frac{1}{2}\sum_i (y_i - a_i)^2 \quad , \tag{78}$$

donde la suma se realiza en los nodos de la capa de salida. Para obtener el gradiente con respecto a un peso específico $W_{j,i}$ en la capa de salida, solo es necesario extender la activación a_i, de tal manera que los otros términos en la sumatoria no se afectan por $W_{j,i}$ con Δ_i como se definió anteriormente. Para obtener el gradiente con respecto a los pesos $W_{k,j}$ que conecta la capa de entrada con la capa oculta, se mantiene la sumatoria completa sobre i debido a que cada valor de salida a_i se puede modificar por los cambios en $W_{k,j}$, así mismo se extiende la activación a_j. A continuación se muestra cómo se propaga el operador de derivación a través de la red:

$$\begin{aligned}
\tfrac{\partial E}{\partial W_{i,j}} &= -(y_i - a_i)\tfrac{\partial a_i}{\partial W_{i,j}} = -(y_i - a_i)\tfrac{\partial g(in_i)}{\partial W_{i,j}} \\
&= -(y_i - a_i)g'(in_i)\tfrac{\partial in_i}{\partial W_{i,j}} = -(y_i - a_i)g'(in_i)\tfrac{\partial}{\partial W_{i,j}}(\textstyle\sum_j W_{j,i}a_j) \\
&= -(y_i - a_i)g'(in_i)a_j = -a_j\Delta_i
\end{aligned}$$

y

$$\begin{aligned}
\tfrac{\partial E}{\partial W_{i,j}} &= -\textstyle\sum_i (y_i - a_i)\tfrac{\partial a_i}{\partial W_{i,j}} = -\sum_i (y_i - a_i)\tfrac{\partial g(in_i)}{\partial W_{i,j}} \\
&= -\textstyle\sum_i (y_i - a_i)g'(in_i)\tfrac{\partial in_i}{\partial W_{i,j}} = -\sum i\Delta_i \tfrac{\partial}{\partial W_{k,j}}(\textstyle\sum_j W_{j,i}a_j) \\
&= -\textstyle\sum_i \Delta_i W_{j,i}\tfrac{\partial a_j}{\partial W_{k_j}} = -\sum_i \Delta_i W_{j,i}\tfrac{\partial g(in_j)}{\partial W_{k,j}} \\
&= -\textstyle\sum_i \Delta_i W_{j,i}g' in_j \tfrac{\partial in_j}{\partial W_{k,j}} \\
&= -\textstyle\sum_i \Delta_i W_{j,i}g' in_j \tfrac{\partial}{\partial W_{k,j}}(\textstyle\sum_k W_{k,j}a_k) \\
&= -\textstyle\sum_i \Delta_i W_{j,i}g' in_j a_k = -a_k\Delta_j \quad .
\end{aligned}$$

Queda claro que el proceso se puede continuar para redes con más de una capa oculta, lo cual justifica el algoritmo general mostrado en la Tabla III.

II.6.2 Validación cruzada

La meta de un clasificador es que pueda predecir eficazmente datos desconocidos (esto es, datos de prueba). Sin embargo, no es útil alcanzar una precisión alta de entrenamiento (clasificadores que predicen datos de entrenamiento de los cuales las etiquetas son conocidas). Por lo tanto, una manera común es separar los datos de entrenamiento en dos partes, de los cuales uno es considerado desconocido durante el entrenamiento del clasificador. Una versión mejorada de este procedimiento es la validación cruzada.

En *validación cruzada de v-conjuntos* (del inglés v-fold cross-validation) primero se divide el conjunto de entrenamiento en v subconjuntos de igual tamaño. Secuencialmente, se prueba un subconjunto con el clasificador entrenado con los $v-1$ subconjuntos restantes. De esta manera, se predice cada instancia del conjunto de entrenamiento. La exactitud de la validación cruzada es el porcentaje de los datos que son clasificados correctamente.

Capítulo III

Reconocimiento de clases de objetos

Recordando el objetivo de esta tesis:

Implementar un sistema que realice el reconocimiento de objetos mediante la búsqueda automática de modelos de clases el cual extraiga información relevante de textura de los objetos en las imágenes digitales mediante el uso de algoritmos genéticos.

En este capítulo se muestra el proceso que se llevó a cabo para cumplir con dicho objetivo. Una vez realizada la implementación del sistema, se presentaron dos problemas a resolver para medir su desempeño: el reconocimiento de expresiones faciales (felicidad, sorpresa, enojo) y el reconocimiento de distintos objetos (edificios, caras, carros, árboles y vacas). Estos problemas se pueden englobar en uno solo: el *reconocimiento de clases de objetos*[3].

Antes de describir el método implementado, se desea puntualizar el uso de los algoritmos genéticos debido a la importancia actual de la evolución.

III.1 Importancia de la evolución

La Computación Genética y Evolutiva (GEC, Genetic and Evolutionary Computation) es un campo de investigación reciente en ciencias de la computación la cual trabaja con sistemas adaptativos y técnicas de optimización inspiradas en las reglas de la evolución natural (Olague *et al.*, 2006). Una de sus metas es dotar a las computadoras con la capacidad de procesamiento de la información comparable a la que se encuentra en la

[3]Para efectos prácticos, una expresión facial puede ser vista como un tipo de objeto.

naturaleza (Holland, 1975; Koza, 1992; Mitchell, 1996; Goldberg, 1989).

En particular, los métodos de GEC se pueden utilizar efectivamente en aquellos campos en donde las tareas requieren de técnicas robustas y flexibles para optimizar el desempeño en diferentes escenarios posibles, los cuales caracterizan problemas del mundo real (GECCO, CEC, EuroGP)[4]. Entre estos campos, la Visión por Computadora y el Procesamiento de Imágenes (CVIU, Computer Vision and Image Understanding) representan uno de los retos en cuanto a complejidad de las tareas que se pueden resolver a fin de tener computadoras con capacidades de percepción como la de los seres humanos, que puedan tener sensibilidad al ambiente, comprensión de los datos sensados, identificación de patrones, toma de acciones apropiada y aprendizaje a través de la experiencia (CVPR, ICCV, ECCV, ICPR)[4].

Las aplicaciones del mundo real de CVIU actualmente incluyen la navegación de robots y vehículos autónomos, inspección, control de calidad, vigilancia, por mencionar unos cuantos. Para llevar a cabo estas tareas de alto nivel, es necesario resolver problemas de bajo nivel como la extracción de características, el modelado 3D y la clasificación de objetos. La CVIU gana relevancia continuamente dentro del gran número de campos de aplicación con técnicas de GEC. Esto debido a la capacidad de explorar espacios de búsqueda enormes.

Uno de los beneficios de estudiar técnicas GEC dentro del marco computacional de CVIU es la maduración de las capacidades del procesamiento de la información basados en problemas del mundo real. Otro de los beneficios es el desarrollo de las técnicas de CVIU con una mejor comprensión de las imágenes de escenas del mundo real.

El uso de algoritmos de evolución en tareas de CVIU del mundo real requiere el conocimiento del dominio de la aplicación y la abstracción del dominio del problema, a través de estructuras que se puedan evolucionar por medio de la selección de representaciones

[4]ver Apéndice A

adecuadas. Sin embargo, la efectividad del diseño en un sistema de evolución necesita las respuestas a las siguientes preguntas: ¿Qué será puesto a evolucionar?, ¿Cuáles son los mecanismos necesarios para que tome lugar la evolución?, ¿Cómo se puede evaluar una estructura que ha sido evolucionada? Las respuestas a estas preguntas dependerán del problema que se quiera resolver.

Por esto, la GEC hace posible emplear a la evolución para resolver problemas en un gran número de aplicaciones. De aquí su utilización en el trabajo de tesis.

III.2 Proceso para el reconocimiento de clases de objetos

Como se mencionó en el capítulo II, el proceso general que se sigue para realizar el reconocimiento de objetos en imágenes consta de 6 etapas (pueden ser menos):

1. Adquisición de las imágenes.
2. Preprocesamiento.
3. Segmentación.
4. Representación de la información.
5. Extracción de características.
6. Reconocimiento del objeto.

Debido a que el sistema implementado no realiza la etapa de segmentación, y a manera de esquematizarlo fácilmente para su uso general, el proceso que se utilizó para el reconocimiento de clases de objetos es el mismo que el aplicado para el reconocimiento de expresiones faciales, ver Figura 7, el cual consta de 3 etapas:

1. Adquisición de los datos.

2. Extracción de características.

3. Clasificación (reconocimiento) de objetos.

III.2.1 Adquisición de los datos

Los datos utilizados[5] se pueden dividir en dos conjuntos: grupo I para el reconocimiento de expresiones faciales y grupo II para el reconocimiento de objetos.

Durante esta etapa se requiere de un detector de objetos con el fin de eliminar información innecesaria. Por esto, se realizó la extracción de la información relevante recortando el área principal del objeto. Esta área se convierte en la nueva imagen. Este proceso se repitió para todas las imágenes a utilizar, ver Figura 24.

Después, se redimensionan las imágenes. En el grupo I, las imágenes quedaron de los siguientes tamaños: 32×32 y 64×64 pixeles. El tamaño de las imágenes del grupo II es de 128×128 pixeles, ver Figura 25.

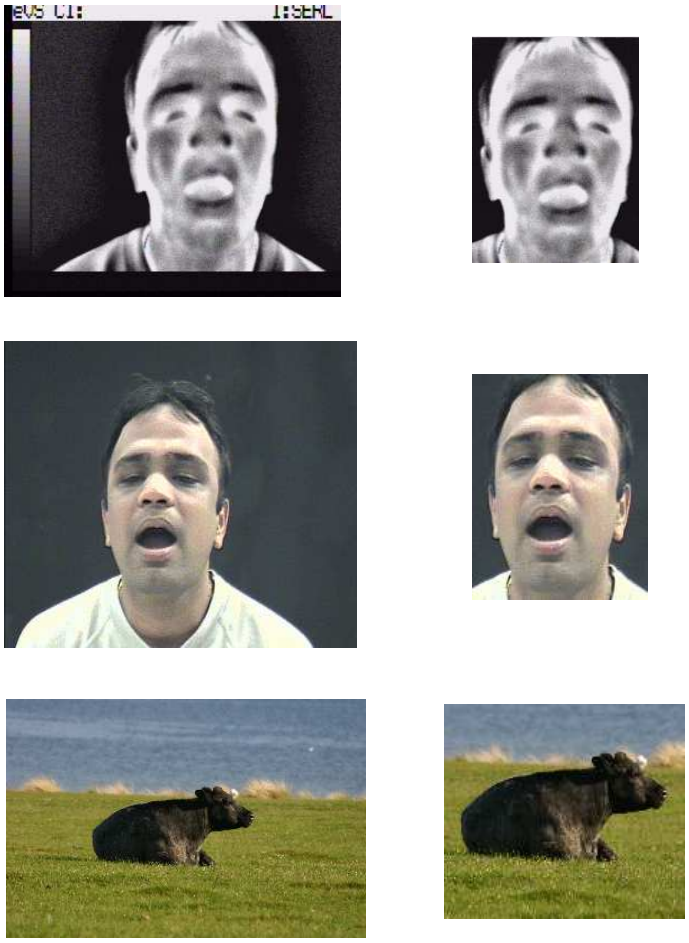
III.2.2 Extracción de características

Para extraer las características de las imágenes utilizadas, se combinaron 3 métodos: algoritmos genéticos (AG), análisis de textura (específicamente las matrices de coocurrencia en niveles de gris, GLCM) y la máquina de soporte vectorial.

Durante esta etapa se implementó un GA el cual busca la mejor selección de regiones de interés (una o más regiones) y su extracción de características necesarias para la etapa de clasificación. Cada posible solución del GA está codificada como una cadena binaria (1s y 0s) y su representación gráfica se muestra en la Figura 26. De esta forma, la primera parte codifica la selección de las regiones de interés de la siguiente manera:

1. Las variables $c_1, ..., c_r$ controlan la activación de las r regiones (1 activa, 0 no activa). Es decir, para encontrar las r regiones de mayor interés, el estado de las

[5]Ver sección IV.2.



(a) (b)

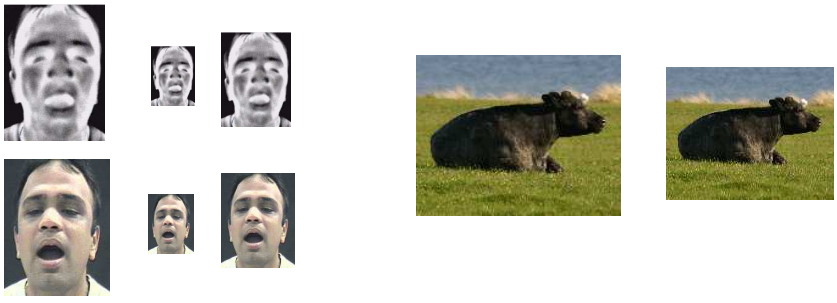
Figura 24: *Ejemplo de las imágenes* (a) antes y (b) después del recorte.

variables (activo o no activo) es variable. Si se desean encontrar *exactamente* r regiones de interés, entonces las r regiones estarán activas $(c_1, ..., c_r = 1)$.

2. Los bloques $w_1, ..., w_r$ definen las características de las regiones a seleccionar. Para

$$w_i = \{x_{w_i}, y_{w_i}, h_{w_i}, w_{w_i}\} \qquad donde\ i = 1, ..., r \quad , \tag{79}$$

(x_{w_i}, y_{w_i}) es el centro y (h_{w_i}, w_{w_i}) son el alto y ancho de la región i.



(a) (b)

Figura 25: *Ejemplo de las imágenes antes y después del redimensionamiento. (a)grupo I y (b)grupo II.*

La segunda parte codifica los parámetros necesarios para la extracción de características de las regiones seleccionadas de la siguiente manera:

1. Los bloques $\pi_{w_1}, ..., \pi_{w_r}$ contienen los parámetros necesarios para calcular la GLCM de cada región definida por w_i de acuerdo a las siguientes variables:

$$\pi_{w_i} = \{V_{w_i}, d_{w_i}, \theta_{w_i}\} \qquad i = 1, ..., r \quad , \tag{80}$$

   donde V es el tamaño de la ventana, d es la distancia y θ es la dirección o ángulo.

2. Se tienen 11 variables de decisión, cada una de las cuales corresponde a un descriptor de textura (ver capítulo II.4). Al igual que las regiones de interés, estas variables pueden estar o no activadas (1 ó 0), lo que determina el tamaño del vector de características γ extraído de cada región.

$$\gamma_{w_i} = \{\beta_{w_i,1}, \beta_{w_i,2}, ..., \beta_{w_i,2t}\} \qquad i = 1...r,\ t = 1..11 \quad . \tag{81}$$

   Este vector contiene la media y la desviación estandar de los descriptores activos de cada región. Debido a lo anterior, el tamaño del vector de características es

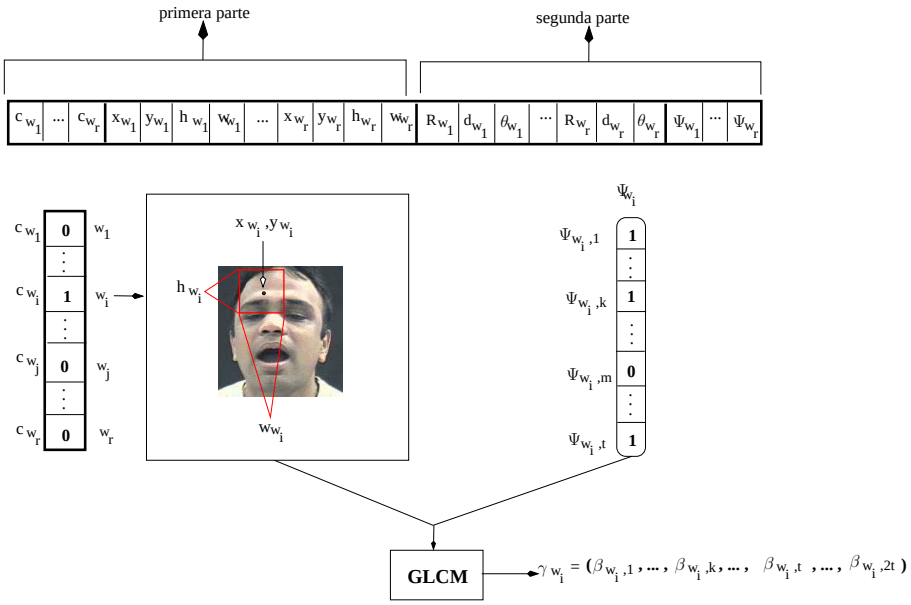


Figura 26: *Codificación e interpretación de una solución posible del GA.*

igual a $2t$, donde t es el número de variables de decisión activadas (como máximo se tendrá $t = 11$, en caso de estar activadas todas las variables de decisión). Con esto, el AG incluye la búsqueda de los mejores descriptores para dichas regiones.

Para el AG implementado, la inicialización de la población se hace de manera aleatoria. Se utiliza la técnica elitista para la selección y el cruzamiento en un punto para la reproducción. El criterio de paro se cumple al alcanzar un número de generaciones dado.

La función de aptitud equivale al porcentaje de clasificación de los objetos del conjunto de imágenes dado, utilizando las características extraídas de las regiones seleccionadas.

La función de aptitud se define de la siguiente manera:

$$f = C(\gamma_1, ..., \gamma_n) \quad . \tag{82}$$

donde:

f es la función de aptitud.

C es el porcentaje de clasificación conocido como aptitud o exactitud(del inglés accuracy).

γ_i es el vector de características de la imagen i.

n es el número de imágenes de entrenamiento.

Los vectores de características se codifican de tal manera que el SVM[6] pueda trabajar con ellos. Esto es, en un archivo se guardan el número de clase del objeto que se encuentra en la imagen, y la media y la desviación estándar de los descriptores de cada región, ver Figura 27.

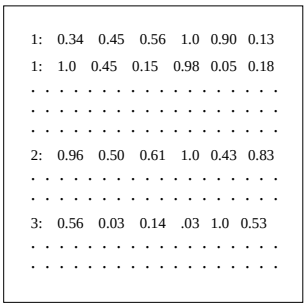


Figura 27: *Archivo que contiene las características extraídas correspondientes a 3 descriptores (2 valores por descriptor (media y desviación estandar) = 6 valores para cada imagen). Cada renglón corresponde a una región de las imágenes de entrenamiento.*

El porcentaje de clasificación del SVM se obtiene por medio de validación cruzada (ver capítulo II.6).

Para el SVM, el kernel utilizado es el de Función en Base Radial, expresado por

[6]Se utiliza el software LIBSVM, este se describe en la sección IV.1.

la Ecuación (52), ya que muestra un mejor desempeño para clasificar problemas no lineales comparado con otros kérneles.

Antes de correr el GA, se calculan los descriptores de las imágenes (tanto de entrenamiento como de prueba) y se guardan en archivos de texto. Para esto, se tiene la siguiente estructura de archivos:

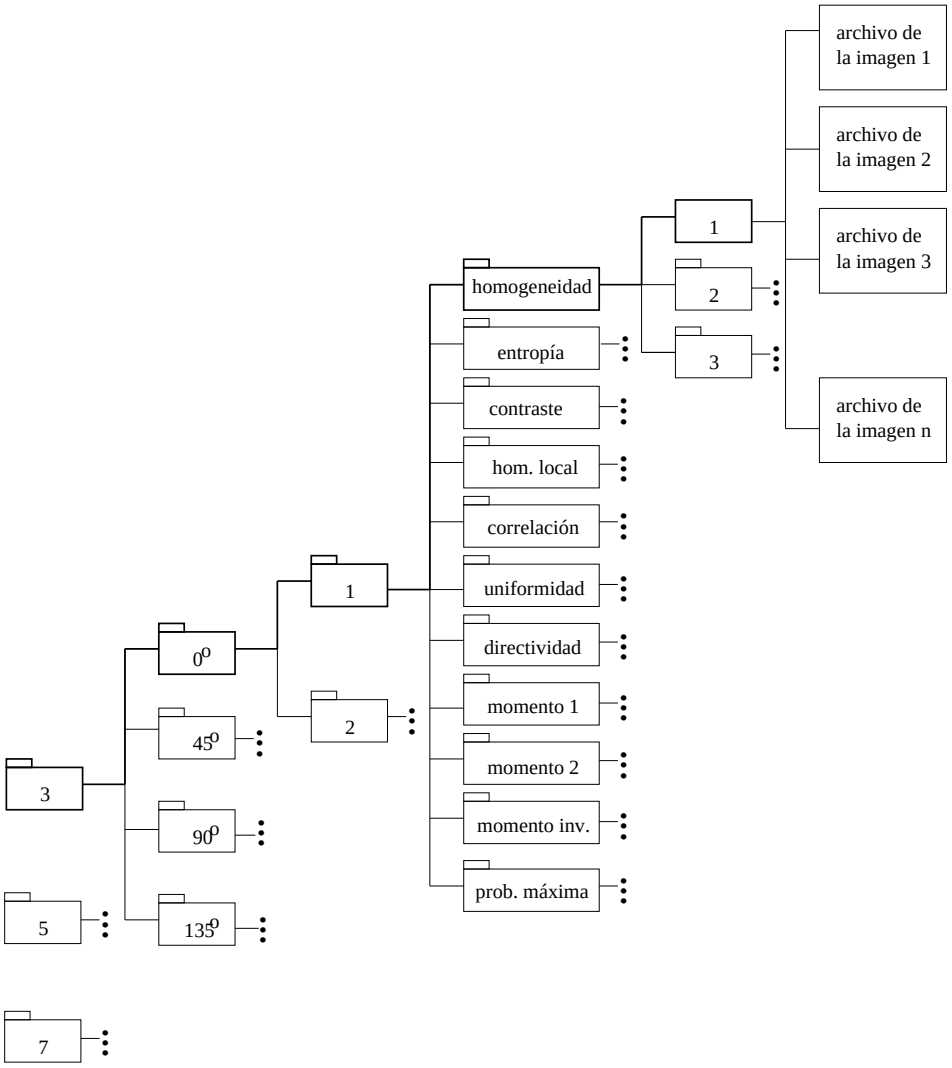


Figura 28: *Estructura de archivos*

La interpretación de esta estructura es la siguiente:

- En el primer nivel se tiene el *tamaño de la ventana*, con valores posibles de 3, 5 y/o 7.
- Dentro de cada directorio de tamaño de la ventana se tiene la *dirección*, ubicada en el segundo nivel del esquema, con valores posibles de 0°, 45°, 90° y 135°.
- Dentro de cada directorio de dirección se tiene la *distancia*, ubicada en el tercer nivel del esquema, con valores posibles de 1 hasta tamaño de ventana -1.
- Dentro de cada directorio de distancia se tiene el *descriptor*, ubicado en el cuarto nivel del esquema, los cuales son: entropía, contraste, homogeneidad, homogeneidad local, correlación, uniformidad, directividad, momento 1, momento 2, momento inverso y probabilidad máxima.
- Dentro de cada directorio de descriptor se tiene la *clase*, ubicada en el quinto nivel del esquema, con valores posibles de 1 a 3 y 1 a 5 dependiendo del número de clases a clasificar (para este trabajo).
- Por último, Dentro de cada directorio de clase se encuentran los archivos que contienen la información extraída de cada imagen, calculada a partir de una selección dada de los parámetros anteriores. A cada imagen le corresponde un archivo en la estructura.

En la Figura 28, se muestra la estructura de los archivos generados a partir de una elección de: *tamaño de ventana* $= 3$, *dirección* $= 0°$, *distancia* $= 1$, *descriptor* $=$ *homogeneidad y clase* $= 1$.

Los archivos del grupo I se cargan en memoria para reducir el tiempo de ejecución del AG al momento de obtener los vectores de características. Sin embargo, en el grupo

II no fue posible hacer esto debido al gran número y tamaño de matrices requerido para almacenarlas.

III.2.3 Clasificación

Una vez que se obtiene la mejor solución del GA, la que mejor clasifique las imágenes de entrenamiento, se decodifican sus variables para seleccionar las regiones y extraer las características de las imágenes de prueba.

Entonces, para asignar una clase al objeto contenido en una imagen de prueba, el SVM codifica las caracteristicas de las imágenes de entrenamiento y las características de las regiones de dicha imagen. Esta da como resultado un número correpondiente a la clase del objeto.

Capítulo IV

Experimentación

Este capítulo se divide en cuatro partes: primero se mencionan las herramientas de trabajo utilizadas para la realización del sistema planteado, luego se describe el conjunto de datos empleado para el entrenamiento y prueba del sistema, después se detallan los experimentos realizados para evaluar dicho sistema y por último se comparan los resultados obtenidos con el trabajo publicado por Winn *et al.* (2005), del cual se obtuvo el conjunto de datos utilizado para el reconocimiento de objetos.

IV.1 Herramientas de trabajo

En esta sección se mencionan las características de la computadora y del software utilizado para la realización de la tesis.

IV.1.1 Unidad central de proceso

- Computadora personal
- Marca: DELL
- Memoria RAM: 512 MB.
- Procesador: Intel Pentium IV 800 MHz.
- Disco duro: 200 GB.
- Monitor: 17 pulgadas.
- Sistema operativo: Linux (distribución Fedora Core 4), Windows XP.

IV.1.2 Software

El GA y algunas funciones del LIBSVM (validación cruzada y clasificación multiclase, principalmente) se adecuaron a VXL debido a las facilidades y poder computacional que proporciona esta última.

GA

Este paquete fue desarrollado por el profesor Kalyanmoy Deb con la asistencia de sus estudiantes (GA, 2001). La implementación utiliza variables binarias y reales. Se pueden utilizar la combinación de variables.

Las características del GA son:

- Selección por torneo.
- Cruzamiento para cadenas binarias: en un punto; para variables reales: cruzamiento binario simulado (SBX).
- Mutación: Por bit para las variables binarias; mutación polinomial para las reales.

VXL 1.3

Es un conjunto de bibliotecas en C++ diseñadas para la implementación e investigación en visión por computadora (VXL, 2003). Están escritas en ANSI/ISO C++ y por su diseño es portátil a muchas plataformas. Debido a que están diseñadas para visión por computadora deben ser *eficientes*, manejando cantidades de datos numéricos muy grandes con poco esfuerzo. Debido a que está orientado a la investigación, debe ser *portátil*, de tal manera que se pueda escribir y correr un programa en cualquier computadora disponible.

Los paquetes se construyeron extrayendo las funcionalidades de dos sistemas: *Ambiente de Procesamiento de Imágenes* (IUE, Image Understanding Environment) y *Target junior* (TargetJr).

Algunas bibliotecas con las que cuenta VXL son:

- *vcl*: Convierte el compilador a uno estándar de C++.
- *vbl (plantilla básica)*: Contiene un conjunto de plantillas generales y algoritmos.
- *vgl (geometría)*: Maneja la geometría de puntos, curvas y otros objetos en dos o más dimensiones.
- *vil (imágenes)*: Cargar, guardar y manipular imágenes.
- *vnl (numéricas)*: Contenedores y algoritmos numéricos (vectores, matrices, etc.).
- *vul (utilidades)*: Contiene clases y funciones útiles en C++, manejo de directorios y archivos, utilidades de cadenas, entre otros.

LIBSVM 1.0

Es un programa escrito en C++ y Java para clasificación de vectores de soporte, regresión y estimación de distribuciones. Soporta la clasificación multiclase (Chang y Lin, 2001).

La meta es ayudar a los usuarios especializados en otros campos de investigación a utilizar los SVMs de una manera fácil. LIBSVM provee una interfaz sencilla donde los usuarios pueden ligarla fácilmente a sus propios programas. Las características principales de LIBSVM son:

1. Diferentes formulaciones de SVMs.
2. Clasificación multiclase.
3. Validación cruzada para el método seleccionado.
4. Estimación de probabilidad.
5. Código fuente en C++ y Java.

IV.2 Conjunto de datos

IV.2.1 Imágenes

Para realizar los experimentos en esta tesis se usaron dos conjuntos de datos los cuales contienen imágenes con un cierto número de clases de expresiones faciales/objetos de cada conjunto de imágenes. El primer conjunto de datos se utiliza para llevar a cabo el reconocimiento de expresiones faciales, el segundo conjunto para el reconocimiento de objetos. Ambos se describen a continuación.

Para el reconocimiento de expresiones faciales

El conjunto de datos OTCBVS (2005) contiene imágenes de 30 personas diferentes. Cada persona asume 3 diferentes poses correspondientes a 3 expresiones faciales distintas. Las imágenes se tomaron desde 12 ángulos distintos. Cada imagen en la base de datos se encuentra en formato RGB con un tamaño de 320×240 pixeles. Para cada una de las expresiones y ángulos de vista se tienen dos imágenes, una correspondiente a la banda infrarroja (térmicas) y la otra al espectro visible.

En algunas imágenes de la base de datos, ciertas personas utilizan lentes. Estas imágenes no se toman en cuenta en los experimentos.

El propósito de la primera parte de la tesis es reconocer las expresiones faciales, primero en imágenes térmicas (ver Figuras 29, 30 y 31) y después en imágenes en el espectro visible (ver Figuras 32, 33 y 34). Para esto, se eligieron 3 clases de expresiones faciales: *sorpresa, felicidad y enojo*. Existen dos conjuntos de datos para cada clase: un conjunto de entrenamiento y un conjunto de prueba. El número de imágenes de cada conjunto para cada tipo de imagen se muestran en las Tablas IV y V.

De acuerdo al propósito de la primera parte de la tesis mencionado anteriormente, se eligieron las vistas frontales que mejor caracterizan cada clase de expresión. De cada imagen, se realizó la extracción del área principal del rostro utilizando una técnica de

localización automática de rostros (Trujillo *et al.*, 2005). Esta área se recortó y redimensionó a un formato en niveles de gris a fin de obtener imágenes de tamaño 32×32 y 64×64 pixeles.

Figura 29: *Imágenes térmicas de la clase* ***sorpresa****.*

Figura 30: *Imágenes térmicas de la clase* ***felicidad****.*

Figura 31: *Imágenes térmicas de la clase* ***enojo***.

Figura 32: *Imágenes visibles de la clase* ***sorpresa****.*

Figura 33: *Imágenes visibles de la clase* ***felicidad****.*

Figura 34: *Imágenes visibles de la clase* ***enojo****.*

Tabla IV: Imágenes térmicas.

Clase	*Conjunto de entrenamiento*	*Conjunto de prueba*
Sorpresa	32	2
Felicidad	25	2
Enojo	32	2

Tabla V: Imágenes visibles.

Clase	*Conjunto de entrenamiento*
Sorpresa	32
Felicidad	25
Enojo	32

Para el reconocimiento de objetos

El conjunto de datos CALTECH (2005) contiene 240 imágenes de las cuales 120 son imágenes segmentadas manualmente que corresponden a las otras 120 imágenes. Estas imágenes contienen objetos en diferentes posiciones, con distintas condiciones de luz y desde diferentes ángulos de vista. Cada imagen en la base de datos se encuentra en formato RGB con un tamaño de 320×213 pixeles. Los objetos pertenecen a siete clases: edificio, árbol, vaca, avión, cara, carro y bicicleta.

Como se señaló, algunas imágenes de la base de datos están segmentadas las cuales no serán tomadas en consideración para los experimentos.

El propósito de la segunda parte de la tesis fue reconocer objetos. Para esto, se eligieron 5 clases de objetos: *edificio*, *cara*, *carro*, *árbol* y *vaca*. Se tienen dos categorias para cada clase: un conjunto de entrenamiento (de CALTECH (2005), ver Figuras 35, 37, 39, 41 y 43) y un conjunto de prueba (del web, ver Figuras 36, 38, 40, 42 y 44). El número de imágenes de cada conjunto se muestra en la Tabla VI.

Figura 35: *Imágenes visibles de entrenamiento de la clase* ***edificio***.

Figura 36: *Imágenes visibles de prueba de la clase* ***edificio***.

Figura 37: *Imágenes visibles de entrenamiento de la clase* ***cara****.*

Figura 38: *Imágenes visibles de prueba de la clase* ***cara****.*

Figura 39: *Imágenes visibles de entrenamiento de la clase* ***carro****.*

Figura 40: *Imágenes visibles de prueba de la clase* ***carro****.*

Figura 41: *Imágenes visibles de entrenamiento de la clase* ***árbol****.*

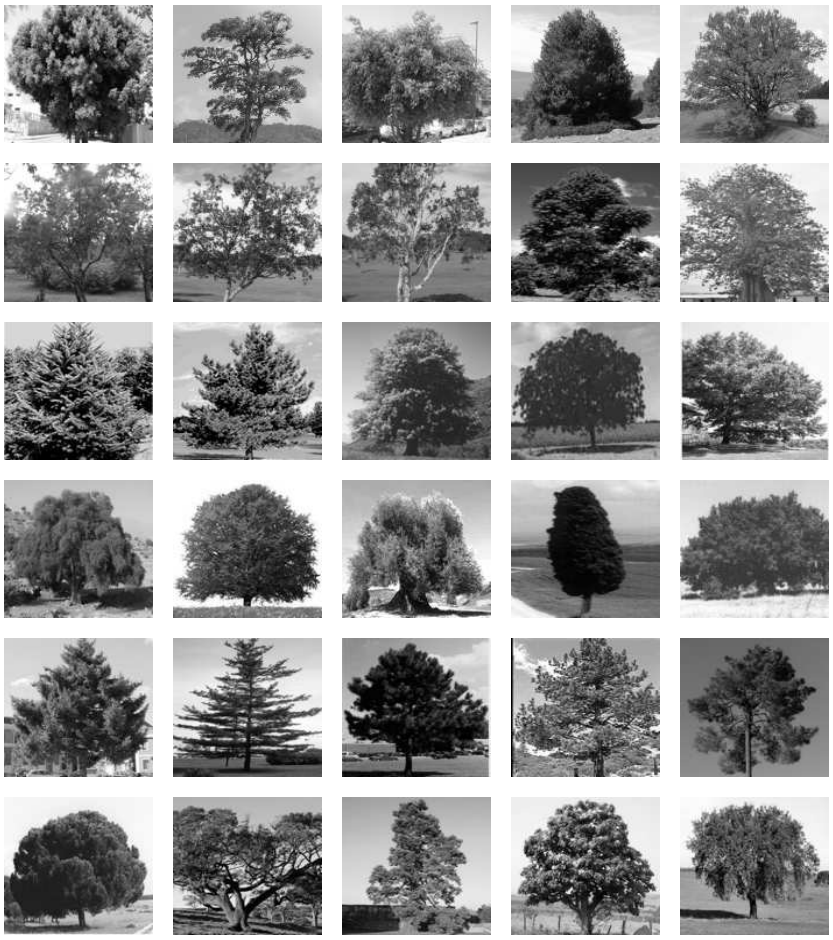


Figura 42: *Imágenes visibles de prueba de la clase* ***árbol****.*

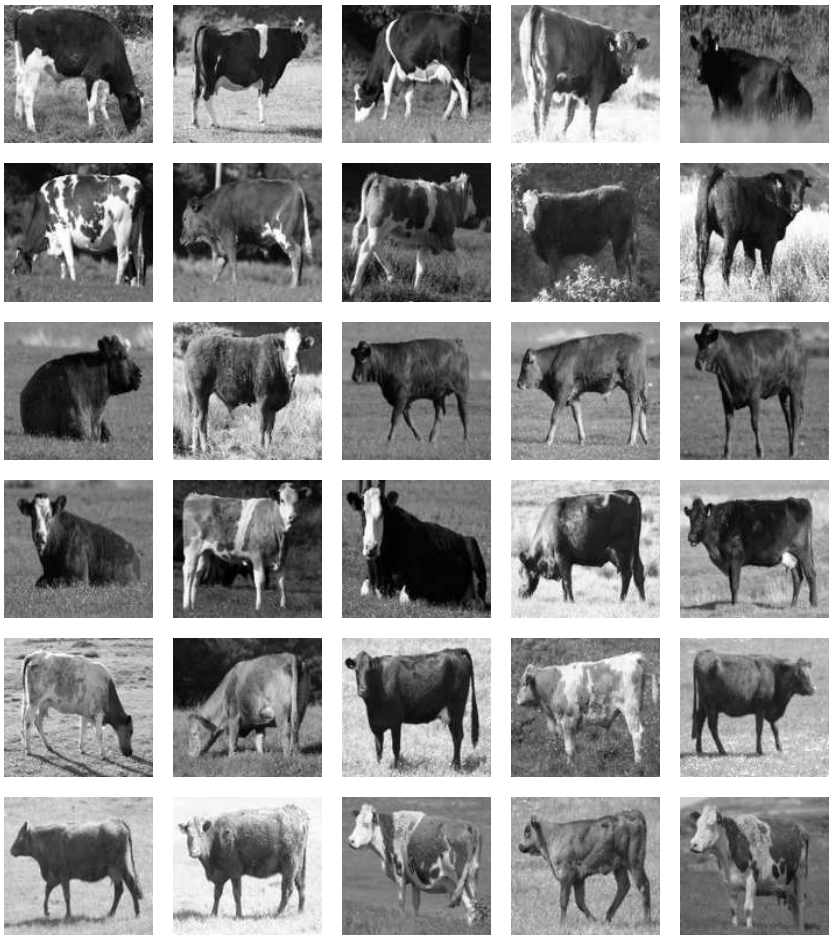


Figura 43: *Imágenes visibles de entrenamiento de la clase* ***vaca****.*

Figura 44: *Imágenes visibles de prueba de la clase* ***vaca****.*

Tabla VI: Imágenes de objetos.

Clase	*Conjunto de entrenamiento*	*Conjunto de prueba*
Edificio	30	50
Cara	30	50
Carro	30	50
Árbol	30	30
Vaca	30	30

En este caso, las imágenes se redimensionaron a un formato en niveles de gris de tamaño 128×128 pixeles.

Ejemplos

En este punto, es útil presentar algunos ejemplos del conjunto de imágenes de objetos (del grupo II) debido a que las clases de objetos difieren en detalle y tamaño. Estas diferencias afectan el desempeño de los métodos aplicados, de ahí la importancia de su mención.

El fondo de todas las imágenes es cambiante pues corresponden a fotografías tomadas en el mundo real. Además, como ya se señaló anteriormente, los objetos se encuentran en distintas posiciones, condiciones de luz y ángulos de vista.

En las imágenes de carros, estos se pueden presentar desde diferentes ángulos de vista y estar parcialmente ocluídos. Incluso, una imagen puede contener más de un carro, como se muestra en la Fig. 45.

Figura 45: *Ejemplos de imágenes que contienen carros los cuales presentan diferentes problemas (distintos ángulos de vista, oclusión, etc.)*

En las imágenes de edificios y vacas, estas se pueden presentar también desde diferentes ángulos de vista aunque no tan variados como en las imágenes de carros. En algunas imágenes, los edificios se encuentran parcialmente ocluídos, ver Figura 46. Por ejemplo, en el caso de las vacas, algunas imágenes no contemplan el cuerpo completo o se presenta oclusión por sí mismas, dependiendo de la posición en la que se encuentren.

Figura 46: *Ejemplos de imágenes de edificios.*

Por último, en las imágenes de las caras y árboles solo se presentan cambios en las condiciones de luz. En general, se encuentran centrados en la imagen, ver Figura 47.

Figura 47: *Ejemplos de imágenes de caras.*

IV.2.2 Cuestionarios

Con el fin de comparar los resultados obtenidos en el reconocimiento automático de expresiones de esta tesis, se diseñaron 2 cuestionarios. Estos cuestionarios evalúan el porcentaje de reconocimiento realizado por humanos. En el primer cuestionario se presentan 15 imágenes térmicas con alguna de las 3 expresiones faciales elegidas (felicidad (1), sorpresa(2), enojo(3)) y una casilla en blanco para escribir el número de clase que le corresponde. En el segundo cuestionario se presentan 15 imágenes en el espectro visible y su casilla en blanco correspondiente, ver Figuras 48-49 y 50-51.

Figura 48: *Cuestionario para reconocer expresiones faciales en imágenes térmicas (parte 1).*

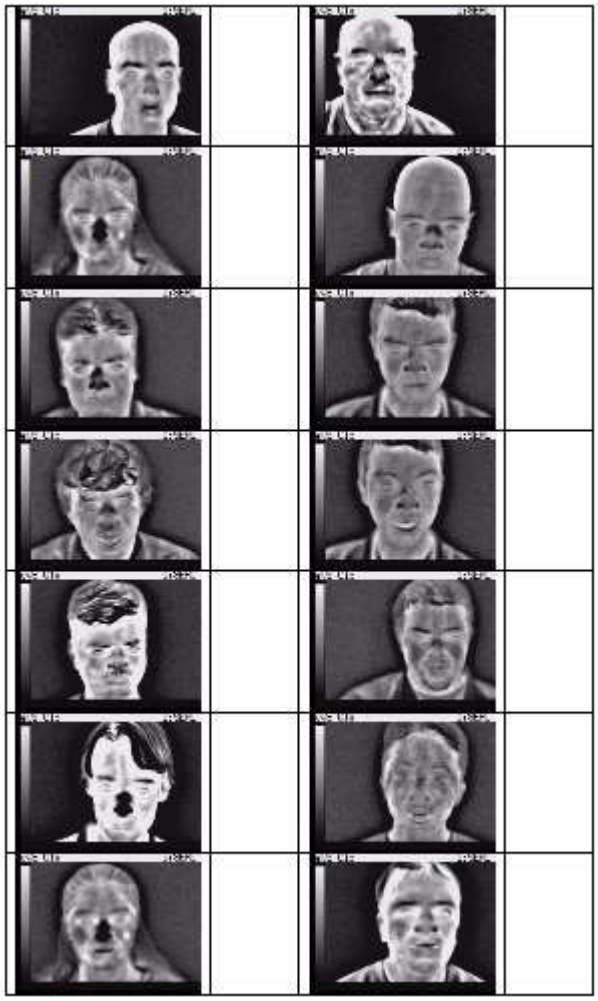


Figura 49: *Cuestionario para reconocer expresiones faciales en imágenes térmicas (parte 2).*

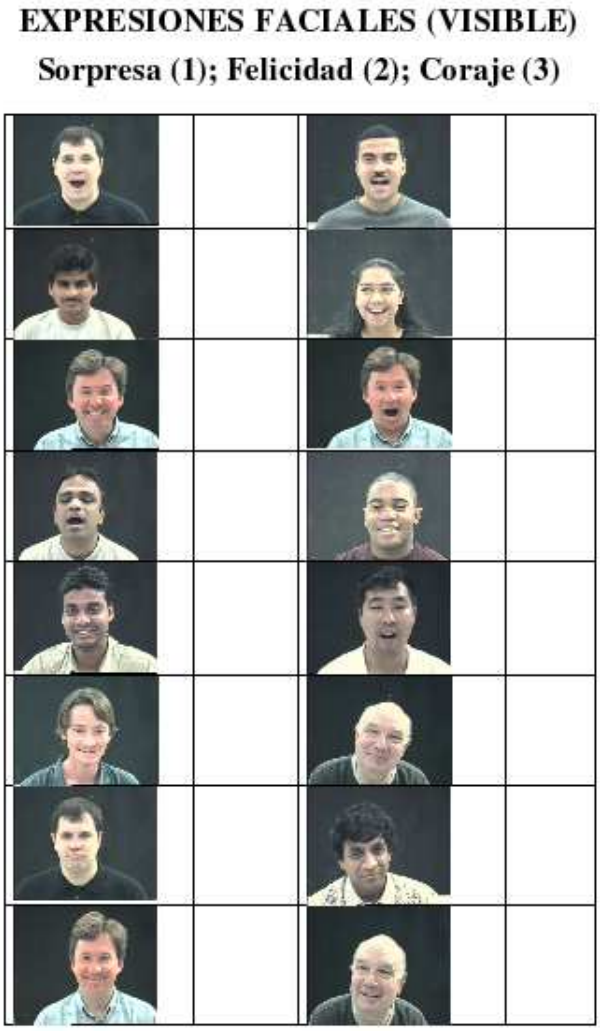


Figura 50: *Cuestionario para reconocer expresiones faciales en imágenes visibles (parte 1).*

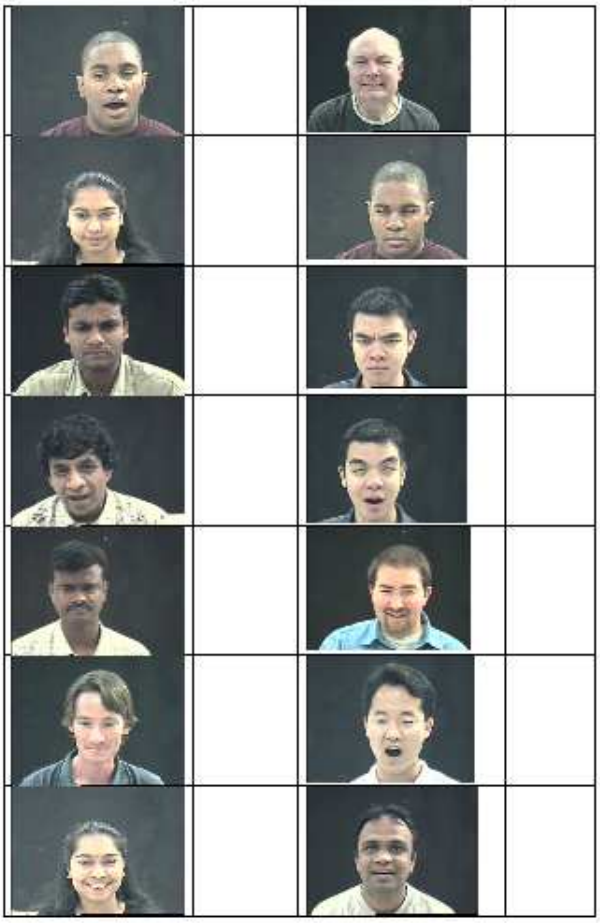


Figura 51: *Cuestionario para reconocer expresiones faciales en imágenes visibles (parte 2).*

Estos cuestionarios se aplicaron a 50 personas. Los resultados de los cuestionarios de las imágenes térmicas y visibles se muestran en las Tablas VII y VIII respectivamente. A estas tablas se les conoce como matrices de confusión en la cual la casilla de intersección de dos clases es su porcentaje de clasificación. Por ejemplo, en la tabla VIII, del 100% de las imágenes que tienen como expresión facial *felicidad*; el 16% fue clasificada como sorpresa, el 2% como enojo y el 82% fue clasificado correctamente como felicidad.

Tabla VII: Matriz de confusión de imágenes térmicas, con respecto a la clasificación realizada por 50 personas (61% de clasificación).

	Felicidad	*Sorpresa*	*Enojo*
Felicidad	**48%**	26%	29%
Sorpresa	33%	**56%**	11%
Enojo	14%	7%	**79%**

Tabla VIII: Matriz de confusión de imágenes visibles, con respecto a la clasificación realizada por 50 personas (77% de clasificación).

	Felicidad	*Sorpresa*	*Enojo*
Felicidad	**82%**	16%	2%
Sorpresa	21%	**78%**	1%
Enojo	25%	5%	**70%**

IV.3 Experimentos

En esta sección se muestran los experimentos realizados y los resultados obtenidos, de los cuales se hace un análisis comparándolo con trabajos relacionados.

IV.3.1 Experimento I

Con las imágenes térmicas del grupo I de tamaño 32×32 pixeles para 3 clases (felicidad, sorpresa, enojo). Las Figuras 52 y 53 muestran la decodificación e interpretación gráfica de la solución obtenida para este caso.

Los parámetros para el AG fueron: 80% cruzamiento, 20% mutación, 100 generaciones y 80 individuos. Como se puede ver, el porcentaje de entrenamiento es muy bajo (40.07%), esto se debe a que el tamaño de las imágenes no permite obtener la suficiente información de textura. Por lo tanto, el porcentaje de clasificación también es bajo.

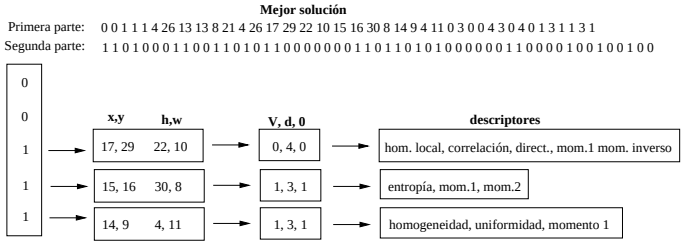


Figura 52: *Mejor solución para imágenes térmicas de tamaño* 32×32. *Porcentaje de clasificación para* ***entrenamiento****:* 40.07%, ***prueba****:* 50%.

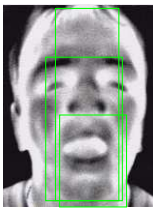


Figura 53: *Imagen de un rostro que muestra las regiones encontradas por la solución vista en la Figura 52.*

IV.3.2 Experimento II

Con las imágenes térmicas del grupo I de tamaño 64×64 pixeles para 3 clases (felicidad, sorpresa, enojo) (ver Figuras 54 y 55).

Los parámetros para el AG fueron: 85% cruzamiento, 15% mutación, 100 generaciones y 80 individuos.

Se aumentó el tamaño de las imágenes de tamaño 32×32 (experimento I) a 64×64 para considerar más información de textura. Como se puede ver, el porcentaje de entrenamiento aumentó, en comparación con el experimento I, aunque sigue siendo bajo (59.55%), Aún así, estos resultados son comparables con la clasificación hecha por humanos (61%), ver Tabla VII. Debido a que el procentaje de clasificación en las imágenes de entrenamiento no fue el esperado, no se aplicó esta solución para imágenes de prueba.

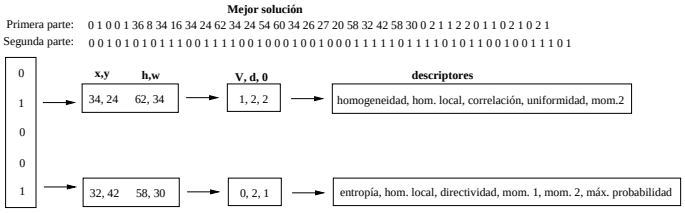


Figura 54: *Mejor solución para imágenes térmicas de tamaño* 64×64. *Porcentaje de clasificación para* ***entrenamiento****:* 59.55%.

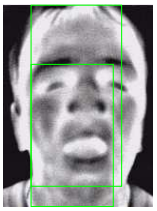


Figura 55: *Imagen de un rostro que muestra las regiones encontradas por la solución vista en la Figura 54.*

IV.3.3 Experimento III

Con las imágenes visibles del grupo I de tamaño 64×64 pixeles para 3 clases (felicidad, sorpresa, enojo) (ver Figuras 56 y 57).

Los parámetros para el AG fueron: 90% cruzamiento, 10% mutación, 200 generaciones y 100 individuos.

Se cambió el tipo de imágenes, de térmicas a visibles, conservando el mismo tamaño de las imágenes del experimento II, con el fin de considerar más información de textura. En este caso, el porcentaje de entrenamiento aumentó, en comparación con el experimento II, aunque muy poco (62.47%). En la clasificación hecha por humanos se obtuvo un (77%), ver Tabla VIII. También, debido a que el procentaje de clasificación en las imágenes de entrenamiento no fue el esperado, no se aplicó esta solución para imágenes de prueba.

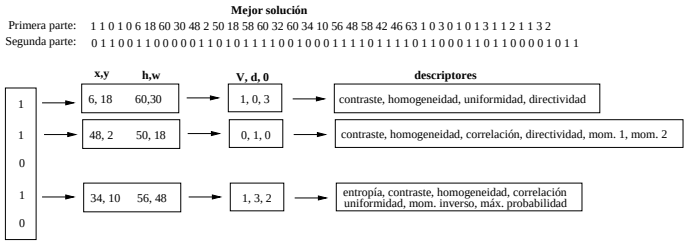


Figura 56: *Mejor solución para imágenes visibles de tamaño* 64×64. *Porcentaje de clasificación para* ***entrenamiento****:* 62.47%.

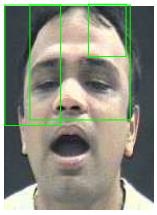


Figura 57: *Imagen de un rostro que muestra las regiones encontradas por la solución vista en la Figura 56.*

IV.3.4 Experimento IV

Con las imágenes del grupo II de tamaño 128 × 128 pixeles para 3 clases (edificios, caras, carros).

Los parámetros para el AG fueron: 80% cruzamiento, 15% mutación, 80 generaciones y 80 individuos.

Como segundo problema se propuso hacer el reconocimiento de distintas clases de objetos. En este caso, se obtuvo una mejoría considerable con respecto a los resultados obtenidos en los experimentos anteriores. Esto se debe a que la diferencia de textura existente entre una clase de objeto y otra es mayor para este nuevo caso con respecto a las expresiones faciales. La Figura 58 muestra la aptitud de los individuos a través de las generaciones.

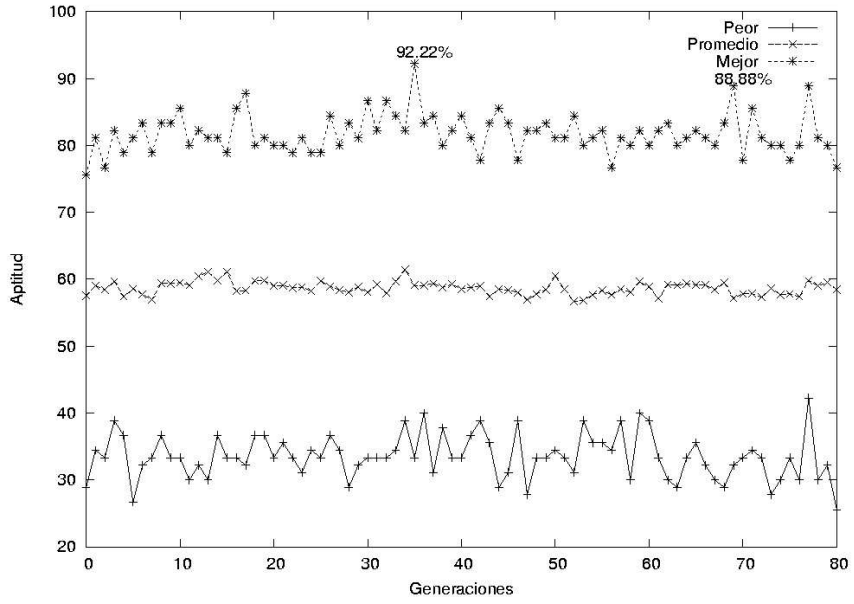


Figura 58: *La gráfica muestra los individuos peores, promedios y mejores de acuerdo a la aptitud de cada uno.*

Individuo 92.22%

En este caso, el porcentaje de entrenamiento es muy bueno (92.22%). Pero el porcentaje de prueba es relativamente bueno (73%), esto se debe a que algunas de las imágenes de prueba contienen características diferentes en los objetos comparados a las imágenes de entrenamiento. Se puede inferir que la ubicación de la región se localiza en la parte inferior de la imagen, debido a que los carros en las imágenes se encuentran en su mayoría en esa parte; además de que cubre una parte significativa de los objetos de las otras imágenes, por esto la mejoría en la clasificación. La matriz de confusión correspondiente es:

Tabla IX: Matriz de confusión de imágenes de objetos para **prueba**: 73%.

	Edificio	*Cara*	*Carro*
Edificio	**68%**	20%	12%
Cara	18%	**78%**	4%
Carro	14%	14%	**72%**

y la solución se decodifica como se ve en las Figuras 59 y 60.

Individuo 88.88%

En este caso, se toma un individuo con porcentaje de entrenamiento de 88.88% para probar su nivel de clasificación. El porcentaje de prueba es alto (80%), debido a que se hizo una selección de las imágenes de prueba, es decir, se tomaron en cuenta solo las que son más parecidas a las imágenes que se tienen para el entrenamiento. Al igual que en la decodificación del individuo anterior, se puede inferir que la ubicación de la región se localiza en la parte inferior de la imagen, debido a que los carros en las imágenes se encuentran en su mayoría en esa parte; además de que cubre una parte significativa de

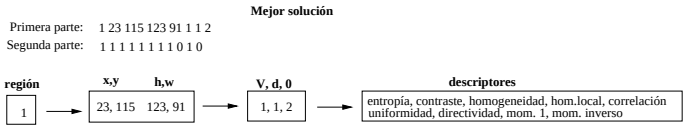


Figura 59: *Mejor solución para imágenes con 3 clases de objetos de tamaño* 128×128.

Figura 60: *Imagen de una clase de objeto (edificio) que muestra la región encontrada por la solución vista en la Figura 59.*

los objetos de las otras imágenes, por esto la mejoría en la clasificación. La matriz de confusión correspondiente es:

Tabla X: Matriz de confusión de imágenes de objetos para **prueba**: 80%.

	$Edificio$	$Cara$	$Carro$
Edificio	**85%**	11%	4%
Cara	6%	**80%**	14%
Carro	12%	12%	**76%**

y la solución se decodifica como se ve en las Figuras 61 y 62.

IV.3.5 Experimento V

Con las imágenes del grupo II de tamaño 128×128 pixeles para 5 clases (edificios, caras, carros, árboles y vacas).

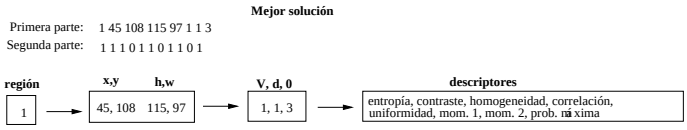


Figura 61: *Mejor solución para imágenes con 3 clases de objetos de tamaño* 128×128.

Figura 62: *Imagen de una clase de objeto (edificio) que muestra la región encontrada por la solución vista en la Figura 61.*

Los parámetros para el AG fueron: 80% cruzamiento, 15% mutación, 80 generaciones y 80 individuos.

Debido a los buenos resultados obtenidos con el expermento IV, se aumentó el número de clases. Es decir, se dejaron las tres primeras clases del experimento anterior (edificios, caras y carros) y se añadieron otras dos (árboles y vacas).

Individuo 83.33%

En este caso, se toma un individuo con porcentaje de entrenamiento de 83.33% para probar su nivel de clasificación. El porcentaje de prueba es relativamente alto (77%). En este caso, se puede inferir que la ubicación de la región se localiza en la parte superior de la imagen, debido a que los árboles en las imágenes se encuentran en su mayoría en esa parte; además de que cubre una parte significativa de los objetos de las otras imágenes, por esto la mejoría en la clasificación. La matriz de confusión correspondiente es:

Tabla XI: Matriz de confusión de imágenes de objetos para **prueba**: 77%.

	Edificio	*Cara*	*Carro*	*Árbol*	*Vaca*
Edificio	**74%**	8%	1%	8%	0%
Cara	6%	**80%**	4%	2%	8%
Carro	26%	0%	**68%**	0%	6%
Árbol	7%	0%	3%	**87%**	3%
Vaca	4%	4%	4%	0%	**48%**

y la solución se decodifica como sigue:

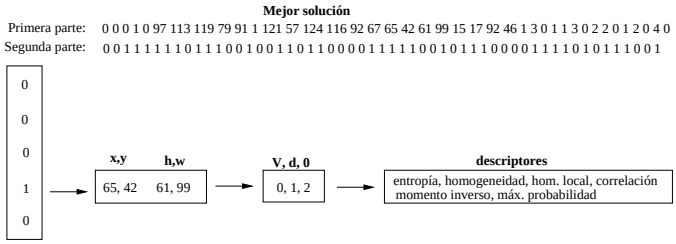


Figura 63: *Mejor solución para imágenes con 5 clases de objetos de tamaño* 128×128.

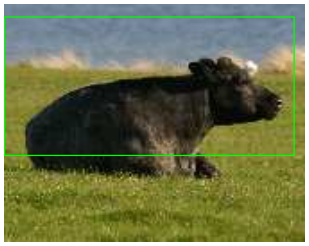


Figura 64: *Imagen de una clase de objeto (vaca) que muestra la región encontrada por la solución vista en la Figura 63.*

IV.4 Resultados

Los resultados obtenidos para el caso de reconocimiento de clases de objetos se pueden comparar con el trabajo realizado por Winn *et al.* (2005) debido a que la investigación realizada por Winn *et al.* (2005) es un trabajo reciente en la resolución de este tipo de problemas. Por esto, se utiliza la misma base de datos. El método utilizado en dicho trabajo consiste en clasificar una región de acuerdo a las proporciones de las distintas palabras visuales. Las palabras visuales y las proporciones de cada objeto se aprenden a partir de un conjunto de imágenes de entrenamiento segmentadas a mano. Se utilizan dos métodos para evaluar la clasificación: vecinos más cercanos y el modelo Gaussiano.

Winn *et al.* (2005) obtuvieron un porcentaje de clasificación de 93% en promedio utilizando las imágenes segmentadas y un porcentaje de clasificación de 76% en promedio eligiendo la región a mano. Este último es el que se compara a nuestro trabajo.

En nuetro caso, se obtuvo un porcentaje de clasificación de 88.88% en la etapa de entrenamiento y un porcentaje de clasificación de 80% en la etapa de prueba. Cabe señalar varios puntos que son importantes en esta tesis a comparación del trabajo anteriormente descrito:

- No se utilizan imágenes segmentadas para hacer el entrenamiento y la clasificación.
- La selección de la región se realiza de manera automática por medio del AG.
- Las imágenes que se utilizan en la etapa de prueba no son parte del conjunto de datos CALTECH (2005), estas fueron obtenidas del web.

Esto demuestra que nuestro sistema tiene un mejor desempeño que el realizado por Winn *et al.* (2005), aún cuando no se tienen las restricciones de la segmentación de las imágenes y que las imágenes de prueba, al no pertenecer al conjunto de datos provisto,

contienen mayor diferencia entre las utilizadas para el entrenamiento. La Tabla XI muestra la comparación de nuestro método y el utilizado por Winn *et al.* (2005).

Tabla XII: Comparación del porcentaje de clasificación.

	Win *et al.*			GA-SVM	
	NN $k = 2000$	NN $k = 216$	Gaussiano	Objetos (3) 88.88%	Objetos (5) 83.33%
Sel. de características	a mano	a mano	a mano	automático	automático
Clasificación	76.3%	78.5%	77.4%	80.0%	77.0%

Capítulo V

Conclusiones

En este capítulo final se presentan las conclusiones del trabajo realizado. Estas se presentan en dos partes, aquellas que se relacionan con el problema de reconocimiento de clases de objetos y aquellas relacionadas al sistema implementado. Finalmente, terminamos describiendo el trabajo que se puede realizar a futuro.

V.1 Conclusiones del reconocimiento de objetos

El problema de reconocimiento de objetos es muy amplio. Desafortunadamente, aún cuando existen diferentes métodos que muestran buenos resultados, no existe un método que pueda manejar el problema del reconocimiento de objetos en el mundo real. Toda la investigación realizada permite solo un número limitado de clases de objetos debido a la poca disponibilidad de los conjuntos de datos. Una aplicación en el mundo real requiere enormes cantidades de datos de entrenamiento los cuales consumen mucho tiempo de recolección. Además, los modelos tienen que ser fáciles de manipular a fin de hacer inferencias de manera más directa.

V.2 Conclusiones del sistema implementado

Para el reconocimiento de expresiones faciales no se obtuvieron los resultados esperados debido a la poca información de textura, aún cuando se aumentó el tamaño de las imágenes, que hiciera diferenciar entre una expresión y otra. Además, las bases de datos de imágenes utilizadas en la literatura para resolver estos problemas son cuidadosamente elegidas, es decir, presentan diferencias mínimas en variación de luz y posición de los

rostros, lo que hace más sencillo su reconocimiento. Sin embargo, se puede decir que las regiones seleccionadas pueden o no estar directamente relacionadas a nuestra noción intuitiva de localización de diferencias, tal como los ojos y la boca.

Para el reconocimiento de distintos objetos, la clasificación en las imágenes de entrenamiento fue significativamente alto, al igual que el porcentaje de clasificación en las imágenes de prueba. En este caso, también existen diferentes tipos de objetos que pertenecen a una clase en particular, los cuales presentan variaciones que no se proporcionan al clasificador al momento del entrenamiento. Aún así, el resultado obtenido es muy bueno.

Existen algunas desventajas. El tiempo de ejecución del AG es grande (semanas). Por otra parte, mientras se aumenta el número de clases, se necesita más entrenamiento pues no se obtiene una solución general a partir de pocas clases. Por ejemplo, una estrategia para detectar caras no detectará aviones.

V.3 Trabajo a futuro

La continuación de este trabajo es grande. Por ejemplo, se puede enfocar en el reconocimiento de objetos como parte de las tareas realizadas por un robot móvil. Tiene otras aplicaciones como la recuperación de imágenes, búsqueda en el web, entre otras, las cuales se mencionan en la introducción de este trabajo.

Lo que se puede agregar al sistema implementado es aumentar el número de clases a reconocer, tener un conjunto de datos más grande, el cual se utiliza tanto para entrenamiento como para la prueba del sistema. Otra idea es usar los descriptores más utilizados en la literatura y por el GA (contraste, homogeneidad y correlación). También se pueden añadir otras características, no solo basadas en textura, por ejemplo las estructurales.

Bibliografía

Bolon, P., J. M. Chassery, J. P. Cocquerez, D. Demigny, C. Graffigne, A. Montanvert, S. Philipp, R. Zéboudj, y J. Zérubia 1995. "Analyse d'images: filtrage et segmentation". Masson, Paris. 488 pp.

Box, G. E. P. 1957. "Evolutionary operation: A method for increasing industrial productivity". Applied Statistics, 6(2):81-101 p.

Bremermann, H. J. 1958. "The evolution of intelligence: The nervous system as a model of its environment". Technical report no. 1, Department of Mathematics, University of Washington, Seattle.

Burgues, C. J. C. 1998. "A tutorial on support vector machines for pattern recognition". Knowledge Discovery and Data Mining, 2:121-167 p.

CALTECH 2005. "Caltech categories". `http://www.vision.caltech.edu/html-files/archive.html`. Abril, 2006.

Chang, C.-C. y C.-J. Lin 2001. "Libsvm: a library for support vector machines". `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. Enero, 2006.

Cover, T. M. y P. E. Hart 1967. "Nearest neighbor pattern classification". IEEE Transactions on Information Theory, IT-13(1):21-27 p.

Draper, B. A. y K. Baek 2000. "Unsupervised learning of biologically plausible object recognition strategies". Biologically Motivated Computer Vision, 1811:238-247 p.

Duda, R. O., P. E. Hart, y D. G. Stork 2001. "Pattern classification". John Wiley and Sons, Inc., New York. 654 pp.

Edelman, S. H. 1997. "Computational theories of object recognition". Trends in Cognitive Sciences, 1(8):296-304 p.

Fogel, L. J. 1962. "Autonomous automata". Industrial Research, 4:14-19 p.

GA 2001. "Genetic algorithm". `http://www.iitk.ac.in/kangal/codes/sga/sga.tar`. Septiembre, 2005.

Goldberg, D. E. 1989. "Genetic algorithms in search, optimization and machine learning". Addison-Wesley Publishing Company, Inc., Reading, Massachusetts. 432 pp.

Gónzalez, R. C. y R. E. Woods 1996. "Tratamiento digital de imágenes". Addison-Wesley/Díaz de Santos, Estados Unidos. 773 pp.

Haralick, R. M. 1979. "Statistical and structural approaches to texture". Proceedings of the IEEE, 67(5):786-804 p.

Haralick, R. M., K. Shanmugam, y I. Dinstein 1973. "Textural features for image classification". IEEE Transactions on Systems, Man and Cybernetics, SMC, 3(6):610-621 p.

Holland, J. H. 1975. "Adaptation in natural and artifical systems". University of Michigan Press, Ann Arbor, Michigan, USA. 211 pp.

Hsu, C.-W., C.-C. Chang, y C.-J. Lin 2003. "A practical guide to support vector classification". Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. 1-12 p.

INAOE 2004. "Visión por computadora". `http://ccc.inaoep.mx/~labvision/vcomp.htm`. Agosto, 2005.

Jain, A. K., R. P. W. Duin, y J. Mao 2000. "Statistical pattern recognition: A review". IEEE Transactions on Pattern Analysys and Machine Intelligence, 22(1):4-37 p.

Koza, J. R. 1992. "Genetic programming: on the programming of computers by means of natural selection". MIT Press, Cambridge, Massachusetts. 840 pp.

Malick, J. y P. Perona 1990. "Preattentive texture discrimination with early vision mechanisms". Journal of the Optical Society of America, 7(5):923-932 p.

Materka, A. y M. Strzelecki 1998. "Texture analysis methods - a review". Reporte técnico , Technical University of Lodz, Institute of Electronics, Brussels. 1-33 p.

Mitchell, M. 1996. "An introduction to genetic algorithms". MIT Press, Cambridge, Massachusetts. 224 pp.

Olague, G., S. Cagnoni, y E. Lutton 2006. "Introduction to the special issue on evolutionary computer vision and image understanding". Pattern Recognition Letters, 27(11):1161-1163 p.

OTCBVS 2005. "Object tracking and classification in and beyond the visible spectrum". `http://www.cse.ohio-state.edu/OTCBVS/05/`. Septiembre, 2005.

Parker, J. R. 1997. "Algorithms for image processing and computer vision". John Wiley and Sons, Inc., New York. 417 pp.

Russell, S. J. y P. Norving 1995. "Artificial intelligence: a modern approach". Prentice Hall, Inc., New Jersey. 932 pp.

Russell, S. J. y P. Norving 2003. "Artificial intelligence: a modern approach". Prentice Hall, Inc., New Jersey, segunda edition. 1080 pp.

Smart, W. D. y M. Zhang 2004. "Applying online gradient descent search to genetic programming for object recognition". En: "The Australasian Workshop on Data Mining and Web Intelligence", Dunedin, New Zealand, Enero, 2006. 133-138 p.

Suetens, P., P. Fua, y A. J. Hanson 1992. "Computational strategies for object recognition". ACM Computing Surveys, 24(1):5-61 p.

Trujillo, L. y G. Olague 2006. "Synthesis of interest point detectors through genetic programming". En: "GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation", Seattle, Washington, USA, Julio, 2006. ACM Press. 887-894 p.

Trujillo, L., G. Olague, R. Hammoud, y B. Hernández 2005. "Automatic feature localization in thermal images for facial expression recognition". IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 3(40):14 pp.

Tuceryan, M. y A. K. Jain 1998. "The handbook of pattern recognition and computer vision". World Scientific Publishing Co., Singapore, segunda edition. 1044 pp.

Van Gool, L. J., P. Dewaele, y A. Oosterlinck 1986. "Texture analysis anno 1983". Computer Vision, Graphics and Image Processing, 29(3):336-357 p.

VXL 2000-2003. "C++ libraries for computer vision research and implementation". `http://vxl.sourceforge.net/`. Septiembre, 2005.

Wechsler, H. 1981. "Texture analysis - a survey". En: "Signal Processing", volume 2. North Holland Publishing Company. 271-282 p.

Wikipedia 2004. "Wikipedia the free encyclopedia". `http://en.wikipedia.org/wiki/Face_Recognition/`. Agosto, 2005.

Winkeler, J. F. y B. S. Manjunath 1997. "Genetic programming for object detection". En: "Proc. Genetic Programming 1997 Conference", Stanford, CA, USA, Julio, 1997. 330-335 p.

Winn, J., A. Criminisi, y T. Minka 2005. "Object categorization by learned universal visual dictionary". En: "ICCV 2005, Tenth IEEE International Conference on Computer Vision", volume 2. Microsoft Research, Cambridge, UK, Octubre, 2005. 1800-1807 p.

Apédice A

Lista de conferencias en Visión por Computadora y Algoritmos de Evolución

- **CEC**: *Congress on Evolutionary Computation* (El congreso más importante dentro de IEEE).
- **CVPR**: *Computer Vision and Pattern Recognition* (Conferencia americana organizada por IEEE con alcance alrededor del mundo).
- **ECCV**: *European Conference on Computer Vision* (La conferencia europea más grande de este tipo).
- **EuroGP**: *European Conference on Genetic Programming* (El evento más importante orientado a Programación Genética).
- **GECCO**: *Genetic and Evolutionary Computation Conference* (La conferencia más importante en este campo).
- **ICCV**: *International Conference on Computer Vision* (El evento más grande en la disciplina de Visión por Computadora).
- **ICPR**: *International Conference on Pattern Recognition* (La conferencia semestral más grande en Reconocimiento de Patrones).