UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: Selección de características. Su aplicación a clasificación de texturas.
Autores: María Lucía Violini
Director: Javier Oscar Giacomantone
Carrera: Licenciatura en Sistemas

Resumen

Los métodos de selección de características son utilizados dentro de la etapa de reducción de dimensión de sistemas de reconocimiento de patrones. Estos métodos se utilizan para obtener el subconjunto de características más relevantes del conjunto completo de características, dicho subconjunto será aquel que maximice una función criterio determinada. En esta tesina se estudian métodos de selección tanto óptimos como sub-óptimos y se implementa una librería de funciones que contiene los métodos estudiados. Además se realiza el estudio de los principales descriptores estadísticos de textura de primer y segundo orden que se obtienen a partir de imágenes digitales. Los mismos son implementados conformando una librería de funciones que permite obtener descriptores de textura a partir de una imagen digital. Por último se plantea un modelo que permite aplicar métodos de selección de características a clasificación de texturas. Se realiza la implementación de dicho modelo dando lugar a una librería de funciones que se utiliza posteriormente para realizar pruebas. Los resultados obtenidos muestran que los métodos de selección de características mejoran el rendimiento de los sistemas de reconocimiento de patrones.

Palabras Claves

- Reconocimiento de Patrones.
- Selección de Características.
- Métodos de Selección de Características.
- Texturas.
- Descriptores Estadísticos de Textura.
- Clasificación de Texturas.

Trabajos Realizados

- Se estudiaron métodos de selección óptimos y sub-óptimos.
- Se estudiaron los principales descriptores de textura de primer y segundo orden.
- Se planteó un modelo de reconocimiento de patrones aplicado a clasificación de texturas.
- Se implementaron tres librerías de funciones para: (1) selección, (2) texturas y (3) pruebas.
- Se realizaron experimentos con dichas librerías y se compararon los resultados obtenidos.

Conclusiones

Luego del estudio de diferentes métodos de selección de características y en base a los resultados obtenidos en las pruebas realizadas, se observa que los métodos de selección de características mejoran el rendimiento del clasificador en los sistemas de reconocimiento de patrones. El método de selección de características sub-óptimo *SFFS* muestra un buen desempeño en general, igual o cercano al desempeño de los métodos óptimos la mayoría de las veces.

Trabajos Futuros

Como trabajo futuro se propone realizar el estudio de otros métodos de selección de características.
Otro aspecto para un trabajo de investigación futuro es el análisis de diferentes funciones criterio y su influencia sobre los métodos estudiados.

Fecha de la presentación: Noviembre de 2014

Universidad Nacional de La Plata

Facultad de Informática

Selección de Características.
Su aplicación a Clasificación de Texturas.

Tesina de Licenciatura en Sistemas

Alumna

María Lucía Violini

Director

Javier Oscar Giacomantone

A mi Familia.

En especial, a mis Padres.

Índice general

Resumen

Los métodos de selección de características son utilizados dentro de la etapa de reducción de dimensión de sistemas de reconocimiento de patrones. Estos métodos se utilizan para obtener el subconjunto de características más relevantes del conjunto completo de características, dicho subconjunto será aquel que maximice una función criterio determinada. En esta tesina se estudian métodos de selección tanto óptimos como sub-óptimos y se implementa una librería de funciones que contiene los métodos estudiados.

Además se realiza el estudio de los principales descriptores estadísticos de textura de primer y segundo orden que se obtienen a partir de imágenes digitales. Los mismos son implementados conformando una librería de funciones que permite obtener descriptores de textura a partir de una imagen digital.

Por último se plantea un modelo que permite aplicar métodos de selección de características a clasificación de texturas. Se realiza la implementación de dicho modelo dando lugar a una librería de funciones que se utiliza posteriormente para realizar pruebas.

Los resultados obtenidos muestran que los métodos de selección de características mejoran el rendimiento de los sistemas de reconocimiento de patrones.

Palabras Claves: Reconocimiento de Patrones, Selección de Características, Métodos de Selección de Características, Texturas, Descriptores Estadísticos de Textura, Clasificación de Texturas.

Capítulo 1

Introducción

Un sistema de reconocimiento de patrones destinado a clasificar texturas presentes en imágenes digitales requiere una etapa que se encargue de generar características a partir de las imágenes para poder formar los patrones. Dichas características pueden ser obtenidas con descriptores estadísticos de textura de primer y segundo orden.

Los descriptores estadísticos de primer orden se calculan a partir del histograma de niveles de gris de la imagen. Entre este tipo de descriptores se encuentran: media, desviación estándar, asimetría, kurtosis, energía y entropía. Los descriptores estadísticos de segundo orden se calculan a partir de las matrices de co-ocurrencia de los niveles de gris de la imagen. Entre estos descriptores se encuentran: energía, contraste, correlación, homogeneidad y entropía.

Cuando el número de descriptores aumenta es necesaria una etapa de reducción de dimensión dentro del sistema de reconocimiento de patrones. Esta etapa puede utilizar métodos de extracción y/o selección de características.

Un método de selección de características combina un algoritmo de búsqueda y una función criterio con el objetivo de encontrar el "mejor" subconjunto de características. Los métodos de selección se encargan de preservar la semántica de las variables originales, facilitando la interpretación por parte del experto en un área de conocimiento en particular.

La función criterio se usa para comparar subconjuntos y determinar qué subconjun-

to es mejor que otro. El "mejor" subconjunto de características, también denominado subconjunto óptimo, es aquel que maximiza la función criterio en cuestión.

Existen diferentes métodos de selección de características, de acuerdo al algoritmo de búsqueda que utilicen pueden ser: óptimos o sub-óptimos. Un método es óptimo cuando utiliza un algoritmo de búsqueda óptimo, el cual garantiza encontrar el "mejor" subconjunto de características. Un método es sub-óptimo cuando utiliza un algoritmo de búsqueda sub-óptimo, el cual no siempre encuentra el "mejor" subconjunto de características pudiendo proporcionar una solución sub-óptima. Dentro de los algoritmos de búsqueda óptimos se destacan: *Búsqueda Exhaustiva (BE)* y *Branch and Bound (BB)*. Y en los algoritmos de búsqueda sub-óptimos se distinguen: *Sequential Forward Selection (SFS)*, *Sequential Backward Selection (SBS)* y *Sequential Forward Floating Selection (SFFS)*, donde los dos primeros son de búsqueda secuencial y el último es de búsqueda secuencial flotante.

Los métodos de selección de características no sólo pueden ser aplicados a sistemas de clasificación de texturas sino que también se pueden utilizar en otras situaciones. Un ejemplo concreto es la utilización de métodos de selección de características sub-óptimos embebidos en técnicas de agrupamiento espectral, lo que permite la selección adecuada del conjunto de autovectores de la matriz de afinidad [1] [2]. En el XIX Congreso Argentino de Ciencias de la Computación (CACIC) se presentó un método de agrupamiento espectral que incorpora una etapa de selección sub-óptima de características [3].

A continuación se detalla la estructura de la tesina:

- En el Capítulo 2 se aborda el problema de selección de características, se estudia la función criterio y los métodos de selección de características tanto óptimos como sub-óptimos. En la última sección del Capítulo 2 se presenta la librería de funciones de selección implementada, denominada *libSelección*.

- En el Capítulo 3 se analiza el concepto de textura y se estudian diferentes descriptores estadísticos de texturas de primer y segundo orden. En la última sección del Capítulo 3 se presenta la librería de funciones de texturas implementada, denomi-

nada *libTexturas*.

- En el Capítulo 4 se plantea la situación de seleccionar características de texturas y se presenta un modelo de reconocimiento de patrones aplicado a clasificación de texturas. En la última sección del Capítulo 4 se presenta la librería de funciones para la realización de pruebas implementada, denominada *libSelTex*.

- En el Capítulo 5 se muestran diferentes experimentos realizados y resultados obtenidos. Se presentan casos de prueba para cada una de las tres librerías implementadas: *libSelección*, *libTexturas* y *libSelTex*. Luego se hacen comparaciones de los métodos de selección en base a los resultados.

- En el Capítulo 6 se exponen las conclusiones a las que se ha llegado y se planea el trabajo a futuro que de continuidad a la investigación realizada en esta tesina.

Capítulo 2

Selección de Características

2.1. El problema de selección

Un modelo general de un sistema de reconocimiento de patrones está compuesto por un conjunto de etapas de procesamiento denominadas *Sensor*, *Reductor de Dimensión* y *Clasificador*, como se muestra en la figura 2.1.

La primera etapa proporciona una representación de los elementos del universo a ser clasificados. La segunda etapa se encarga, a partir del patrón de representación, de extraer la información discriminatoria eliminando la información redundante e irrelevante. Y por último, la tercera etapa asigna una etiqueta a cada patrón que permite identificarlo como perteneciente a una determinada clase [4].

Figura 2.1: Modelo general de un sistema de reconocimiento de patrones.

La etapa de reducción de dimensión es fundamental en los sistemas de reconocimiento de patrones [5]. Su propósito principal es reducir la dimensión del problema removiendo las características que no son relevantes para la clasificación del patrón [6]. Reducir el número de características mejora el rendimiento de los sistemas de reconocimiento de patrones en general, ya que permite: minimizar el costo de extracción de medidas, disminuir los

costos de almacenamiento, reducir el costo computacional de clasificación y mejorar el rendimiento del clasificador.

Para llevar a cabo la etapa de reducción de dimensión pueden utilizarse métodos de extracción y/o selección de características [7][8]. Los métodos de extracción de características son aquellos que permiten obtener un subconjunto de características a partir de la transformación o combinación de las características originales. En cambio, los métodos de selección de características permiten obtener un subconjunto de características del conjunto inicial de características, manteniendo el significado físico de las mismas [9].

El problema de selección de características consiste básicamente en encontrar el "mejor" subconjunto de d características de un conjunto inicial más numeroso de D características, $d < D$. Para ello, se deberá utilizar una función criterio que determine cuál es el "mejor" subconjunto y un algoritmo de búsqueda que permita encontrarlo [10].

Es importante poder determinar cuáles son las "mejores" características de acuerdo al problema de reconocimiento en cuestión, ya que en reconocimiento de patrones ocurre lo que se conoce como *Maldición de la Dimensionalidad* [11][12][13][14]. Este efecto indica que el resultado de una clasificación no mejora necesariamente con un incremento del número de características. Para tamaños pequeños de muestra, el resultado del clasificador mejora al agregar nuevas características hasta alcanzar un máximo, para luego decaer.

Formalmente, el problema de selección de características puede enunciarse de la siguiente manera [15]: dado un conjunto Y de D características, seleccionar un subconjunto $X \subset Y$ de tamaño d que optimice una cierta función criterio $J(X)$. Donde la función criterio J se maximice para el "mejor" subconjunto de características.

$$J(X) = \max_{Z \subset Y, |Z|=d} J(Z)$$

2.2. Función criterio

Seleccionar características consiste en obtener el subconjunto formado por las características más relevantes del conjunto original de características, de acuerdo a una función criterio. El objetivo de seleccionar características es encontrar el subconjunto con el valor

máximo para una función criterio determinada. Por lo tanto, la función criterio se utiliza para medir la calidad de los subconjuntos.

La función criterio podría ser la tasa de acierto de un clasificador o una medida estadística del grado de separación entre las clases correspondientes a los datos analizados, la elección de la función deberá hacerse dependiendo del problema en cuestión.

Por ejemplo, una medida que puede ser utilizada como función criterio en ciertos casos es la *Distancia Mahalanobis* [15]. La misma sirve para medir la distancia de un punto x de la media de una distribución con matriz de covarianza Σ. Bajo la suposición de que las dos distribuciones de clase tienen la misma matriz de covarianza, la *Distancia Mahalanobis* entre las medias respectivas de esas dos distribuciones mide la separabilidad de las dos clases. Se define la *Distancia Mahalanobis* como: $(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$.

2.3. Métodos de selección

Un método de selección de características es aquel que determina la función criterio y el algoritmo de búsqueda a utilizarse. La función criterio deberá ser elegida de acuerdo a la naturaleza del problema que se está analizando, y el algoritmo de búsqueda se deberá elegir considerando un compromiso entre optimización y complejidad que resulte viable [16][17].

Existen diversos algoritmos de búsqueda que pueden ser utilizados para realizar selección de características. En la figura 2.2 se muestra una adaptación de la taxonomía de algoritmos de selección de características presentada por Jain y Zongker [15].

Una primera distinción que puede observarse en la taxonomía se produce entre los algoritmos de selección de características basados en *reconocimiento estadístico de patrones* y los que usan *redes neuronales artificiales*.

Otra distinción posible se da entre algoritmos *óptimos* (aquellos que garantizan encontrar la solución óptima) y algoritmos *sub-óptimos* (no pueden garantizarlo, pudiendo resultar en una solución sub-óptima).

También se produce una división entre algoritmos con *una solución* (en los cuales sólo un subconjunto de características es almacenado y modificado por el algoritmo) y

algoritmos con *muchas soluciones* (mantienen una población de subconjuntos).

Por último se observa una separación entre algoritmos *determinísticos* (producen siempre el mismo subconjunto para un problema determinado) y *estocásticos* (tienen un elemento aleatorio que podría producir subconjuntos diferentes en cada ejecución).

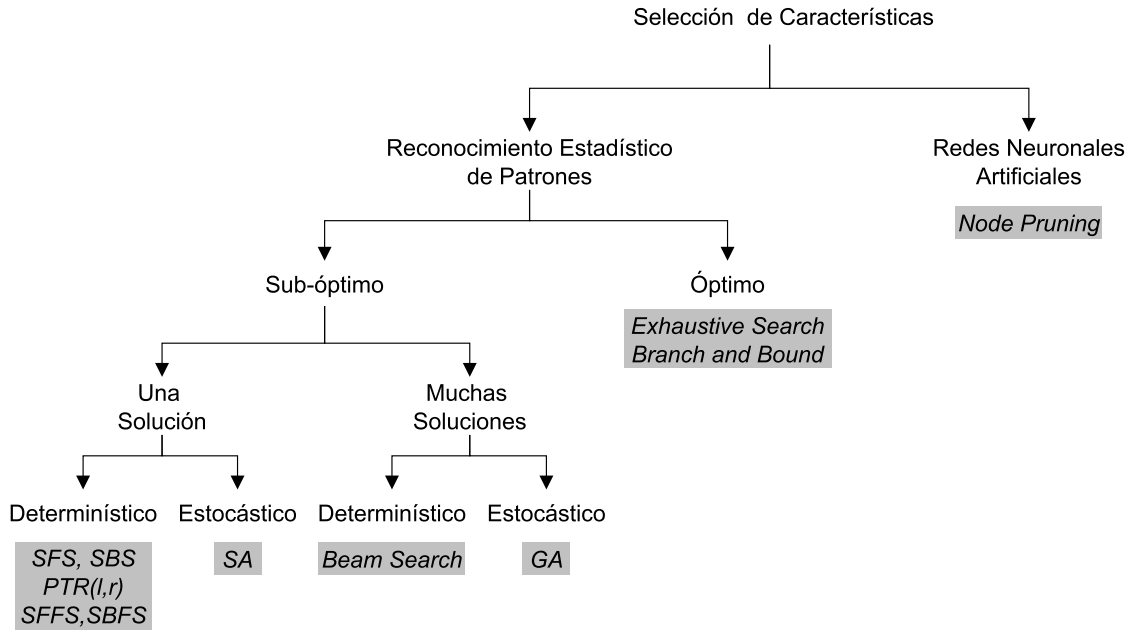


Figura 2.2: Una taxonomía de algoritmos de selección de características.

2.3.1. Métodos de selección óptimos

Un método de selección de características óptimo es aquel que utiliza un algoritmo de búsqueda óptimo para hallar el "mejor" subconjunto de características, de acuerdo a una función criterio determinada. Los algoritmos de búsqueda óptimos garantizan encontrar dicho subconjunto, pero poseen gran costo computacional.

Entre estos algoritmos se encuentran: *Búsqueda Exhaustiva* y *Branch and Bound*. El algoritmo *Búsqueda Exhaustiva* evalúa siempre todos los subconjuntos posibles al contrario del algoritmo *Branch and Bound* que sólo lo hace en el peor de los casos, ya que la mayoría de las veces logra detener su búsqueda antes de evaluar todos los subconjuntos.

Búsqueda Exhaustiva (BE)

El algoritmo *Búsqueda Exhaustiva* [18] es capaz de encontrar el "mejor" subconjunto de d características. Para eso examina todos los posibles subconjuntos de d características a partir de las D características originales, haciendo que la búsqueda crezca exponencialmente. Por lo tanto, es un algoritmo inviable en muchos casos [19].

La cantidad de subconjuntos que serán evaluados estará dada por: $\binom{D}{d} = \frac{D!}{(D-d)!\,d!}$, donde $d < D$.

El número posible de combinaciones $\binom{D}{d}$ se convierte en prohibitivo incluso para valores moderados de d y D. Por ejemplo: $\binom{12}{6}$ es 924, mientras que, $\binom{24}{12}$ es 2704156. Como la cantidad de subconjuntos a ser considerados crece muy rápidamente con la cantidad de características, el algoritmo *Búsqueda Exhaustiva* se vuelve muy costoso computacionalmente en la mayoría de los casos.

Generalmente se emplea el algoritmo *Búsqueda Exhaustiva* cuando se requiere la utilización de un algoritmo de búsqueda óptimo y la función criterio que va a utilizarse no cumple con la propiedad de monotonicidad. No podría aplicarse en ese caso el algoritmo *Branch and Bound*, aunque también proporciona la solución óptima y tiene un mejor desempeño, ya que necesita que la función criterio sea monótona.

Branch and Bound (BB)

El algoritmo *Branch and Bound* [20] es capaz de encontrar el "mejor" subconjunto de d características. Para lograrlo evita realizar una búsqueda exhaustiva, pero requiere que la función criterio sea monótona. La complejidad de este algoritmo en el peor de los casos es exponencial.

El algoritmo utiliza un árbol de búsqueda en el cual las hojas representan todos los subconjuntos posibles de tamaño d. La utilización de una función criterio monótona permite que algunas ramas internas del árbol sean descartadas sin perjudicar la búsqueda del subconjunto óptimo, es decir que no es necesario recorrer todo el árbol para encontrar la solución óptima.

La clave del algoritmo es la propiedad de monotonicidad de la función criterio $J(\cdot)$.

Dados dos subconjuntos X_1 y X_2, si $X_1 \subset X_2$ entonces $J(X_1) < J(X_2)$ [18]. Esto significa que un subconjunto de características nunca es mejor que un conjunto de características más grande que lo contenga.

Existen diversas versiones del algoritmo *Branch and Bound* [21][22][23][24][25][26][27]. Cada una de ellas hace modificaciones o adaptaciones en base a la primera versión del algoritmo, denominada *Branch and Bound*.

El algoritmo *Branch and Bound* selecciona un subconjunto de tamaño d de un conjunto original de tamaño D. La selección se realiza recorriendo un árbol de búsqueda. La raíz del árbol representa el conjunto original Y. Los otros nodos representan posibles subconjuntos de Y de tamaño d. A continuación, mediante un ejemplo, se explicará el procedimiento que realiza el algoritmo para seleccionar el subconjunto óptimo.

Dadas dos clases que poseen distribución normal. Con vectores de media μ_1 y μ_2 respectivamente y matriz de covarianza igual para ambas clases $\Sigma_1 = \Sigma_2 = \Sigma$.

$$\mu_1 = \begin{pmatrix} 2 \\ 3 \\ 3 \\ -2 \end{pmatrix} \mu_2 = \begin{pmatrix} -1 \\ 2 \\ 1 \\ -6 \end{pmatrix} \Sigma = \begin{pmatrix} 4 & 1 & -3 & 0 \\ 1 & 2 & -1 & 0 \\ -3 & -1 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

Se desea determinar, a partir del conjunto completo de características $Y = \{y_1, y_2, y_3, y_4\}$, el "mejor" subconjunto de 2 características. Se considera como función criterio la *Distancia Mahalanobis*. Se aplica el algoritmo *Branch and Bound*, el cual determina que la solución óptima en este caso está dada por el subconjunto $X = \{x_1, x_3\}$ con $J(X) = 26,3333$.

La figura 2.3 muestra el árbol de solución del algoritmo *Branch and Bound* para el caso anteriormente planteado, donde $Y = \{1, 2, 3, 4\}$, $D = 4$ y $d = 2$. Un subconjunto de un nodo está formado por el subconjunto del nodo padre con una, y solamente una, característica removida. El rótulo de las aristas representa la característica removida. El subconjunto de características y el valor de $J(\cdot)$ están indicados próximo al nodo correspondiente. Como una característica es removida por nivel en el camino de la raíz a una hoja, el árbol posee $D - d + 1$ niveles. El número k de un nivel representa el número de características removidas de Y. Un subconjunto X_k corresponde a un nodo de nivel k.

Por ejemplo, en la figura 2.3, $X_1 = \{1, 3, 4\}$ para el nodo 4 y $X_2 = \{1, 2\}$ para el nodo 3 . La elección de las características que deben ser removidas de cada nodo se hace de acuerdo a algunas reglas que serán presentadas a continuación.

Normalmente el algoritmo *Branch and Bound* se implementa de manera que el árbol es construido conforme es recorrido. La principal etapa de ejecución de este algoritmo se denomina etapa de expansión. En dicha etapa se visita un nodo, se analiza el subconjunto correspondiente y se llama a la expansión de los nodos hijos.

El recorrido se realiza a partir de la raíz, de arriba hacia abajo y de derecha a izquierda. La numeración interna de los nodos muestra el camino que sigue el recorrido. Durante la expansión, el valor de $J(X_k)$ es comparado con el valor de un límite B. Cuando el nodo analizado es una hoja, si $J(X_{D-d}) > B$ entonces el valor de B es actualizado $B = J(X_{D-d})$. Por lo tanto, B almacena el mayor valor encontrado en una hoja hasta el momento. Por ejemplo, en la figura 2.3, se realiza esa actualización en los nodos 3 y 5. Como la función criterio es monótona, el valor de $J(\cdot)$ nunca aumenta en el pasaje de un nodo a su sucesor. Por lo tanto, si $J(X_k) \leq B$ no hay motivo para que la búsqueda continúe en el subárbol cuya raíz es el nodo correspondiente al subconjunto X_k, pues ningún valor mayor que B será encontrado en ese subárbol. En consecuencia, esa rama de árbol puede ser podada y eliminada de la búsqueda. Así, el algoritmo *Branch and Bound* consigue encontrar el subconjunto óptimo sin precisar evaluar todos los subconjuntos. Siempre que el valor de B es actualizado, el subconjunto correspondiente X_{D-d} debe ser almacenado como X'. Cuando el algoritmo concluye la búsqueda por todo el árbol, el subconjunto óptimo es X' y $J(X') = B$.

Sea $F = (f_1, f_2, ..., f_{D-d})$ la secuencia ordenada de las características removidas en el camino de la raíz hasta una hoja. Por ejemplo, en la figura 2.3, en el camino desde la raíz hasta el nodo 5, $F = (2, 4)$. Para garantizar que no haya repetición de subconjuntos en el árbol se debe seguir la siguiente regla: $f_1 < f_2 < ... < f_{D-d}$.

Sea q_k el número de sucesores de un nodo de nivel k. A partir de la ecuación anterior se concluye que el número más alto de la primera característica que puede ser removida es $f_1 = d + 1$. Por lo tanto, la raíz debe tener $q_0 = d + 1$ sucesores. Para los demás nodos el número de sucesores es determinado en la expansión del nodo padre. Los nodos sucesores

deben recibir una numeración p de izquierda a derecha. Por ejemplo, en la figura 2.3, para los sucesores del nodo 1, $p = 1$ para el nodo 7, $p = 2$ para el nodo 4 y $p = 3$ para el nodo 2. Así, $q_{k+1} = q_k - p + 1$. Siguiendo esas reglas se representan todos los subconjuntos de tamaño d sin repeticiones en las hojas.

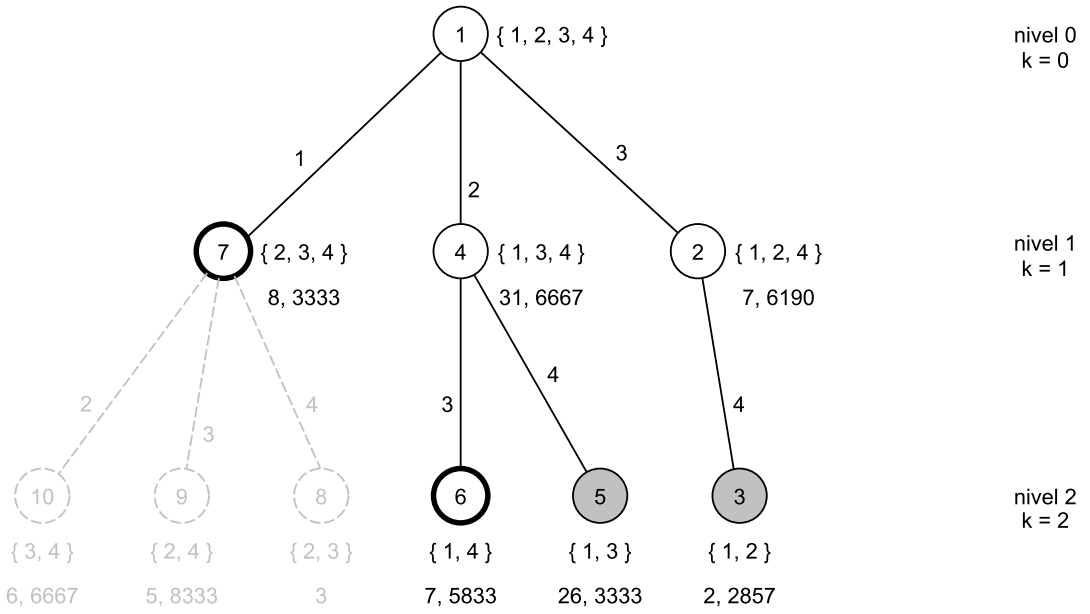


Figura 2.3: Árbol de solución para el algoritmo *Branch and Bound*, con $d = 2$ y $D = 4$. La numeración interna de los nodos indica el camino en el cual es realizado el recorrido. El subconjunto de características y el valor de $J(\cdot)$ están indicados próximos al nodo correspondiente. El rótulo de cada arista indica la característica que fue removida en el pasaje de un nodo del nivel k a un nodo del nivel $k + 1$. Los nodos con fondo sombreado indican que el límite fue actualizado. Los nodos con borde grueso indican que fue encontrado $J(\cdot) < B$, y si el nivel del nodo es $k < 2$ indican poda. Los nodos con borde punteado indican que fueron eliminados por podas.

Otra versión del algoritmo *Branch and Bound* es la versión llamada *Branch and Bound ordenado*. En este algoritmo las características son ordenadas de acuerdo al valor de la función criterio para maximizar las chances de tener nodos eliminados por podas. A continuación, mediante el mismo ejemplo que se utilizó anteriormente, se explicará el procedimiento que realiza el algoritmo para seleccionar el subconjunto óptimo.

Cuanto menor es el valor de $J(\cdot)$ para los nodos más a la izquierda del árbol mayor es el número de nodos eliminados por podas, ya que el número de ramificaciones aumenta de

derecha a izquierda. Además de eso, cuanto menor es el valor de $J(\cdot)$ en cualquier nodo mayor es la probabilidad de poda. Para conseguir valores más bajos en la izquierda del árbol se puede hacer un cambio en el orden en que las características son removidas. El *Branch and Bound ordenado* utiliza esa estrategia reordenando las características en cada expansión de un nodo. La figura 2.4 muestra el mismo problema presentado en la figura 2.3, pero utilizando un árbol de búsqueda *Branch and Bound ordenado*.

Para poder seguir las reglas presentadas en el caso del algoritmo *Branch and Bound* y para que la reordenación sea posible, las características deben ser representadas como variables. El conjunto de todas las características se representa ahora como una secuencia ordenada $Y' = (y_1, y_2, ..., y_D)$, siendo que $1 \leq y_i \leq D$ y $y_i \neq y_j$ si $i \neq j$. Las características removidas en el camino de la raíz hasta una hoja son $F' = y_{f_1}, y_{f_2}, ..., y_{f_{D-d}}$. Así, la ecuación presentada anteriormente, $f_1 < f_2 < ... < f_{D-d}$, aún es válida para la construcción del árbol. Para cada nodo hay un conjunto de características disponibles para remover, $T = y_a, y_{a+1}, ..., y_D$, donde a se determina de acuerdo a la ecuación anterior. Por ejemplo, en la figura 2.4, $a = 1$ para el nodo 1, $a = 4$ para el nodo 2, $a = 3$ para el nodo 4 y $a = 2$ para el nodo 7. Esos valores de a son coincidentes con el menor número de las características que son removidas de los nodos de la figura 2.3. En la expansión de un nodo los elementos de Y' pertenecientes a T son ordenados de manera creciente de acuerdo con el valor de $J(X_k \setminus \{y_i\})$ correspondiente. En el ejemplo, presentado en la figura 2.4, los valores calculados en la expansión de la raíz son:

$$J(Y \setminus \{y_1\}) = J(\{1,2,3,4\} \setminus \{1\}) = J(\{2,3,4\}) = 8,3333$$
$$J(Y \setminus \{y_2\}) = J(\{1,2,3,4\} \setminus \{2\}) = J(\{1,3,4\}) = 31,6667$$
$$J(Y \setminus \{y_3\}) = J(\{1,2,3,4\} \setminus \{3\}) = J(\{1,2,4\}) = 7,6190$$
$$J(Y \setminus \{y_4\}) = J(\{1,2,3,4\} \setminus \{4\}) = J(\{1,2,3\}) = 28.$$

La ordenación resultante es $Y' = (3,1,4,2)$. Las características elegidas para ser removidas de la raíz son 3, 1 y 4, en ese orden. En este caso, el subconjunto obtenido con la eliminación de la característica 2 de la raíz no es aprovechado en el árbol a pesar de haber sido evaluado. En la expansión del nodo 2 el valor calculado es:

$$J(X_1 \setminus \{y_4\}) = J(\{1,2,3\} \setminus \{2\}) = J(\{1,3\}) = 26,3333.$$

Se observa una cantidad mayor de nodos eliminados por podas en el árbol de la figura 2.4 que aquellos observados en la figura 2.3. Además, la ordenación aumenta la probabilidad de encontrar más rápidamente valores más altos para B. Sin embargo, en cada expansión se realiza un mayor número de llamadas a la función criterio. De todos modos, el algoritmo *Branch and Bound ordenado* presenta eficiencia superior al algoritmo *Branch and Bound* en la mayoría de los casos.

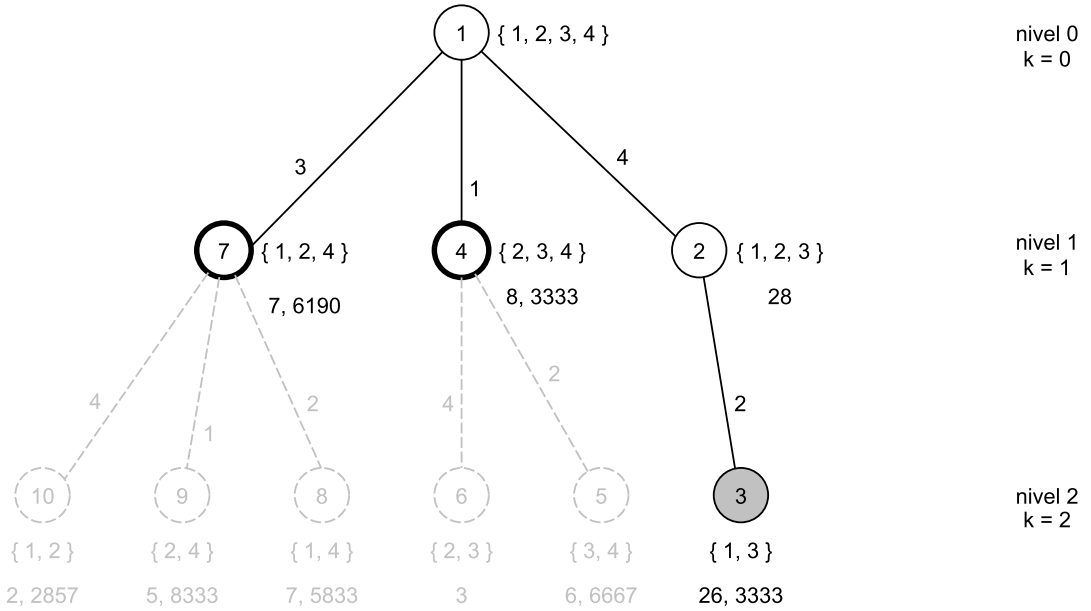


Figura 2.4: Árbol de solución para el algoritmo *Branch and Bound ordenado*, con $d = 2$ y $D = 4$. La numeración interna de los nodos indica el camino en el cual es realizado el recorrido. El subconjunto de características y el valor de $J(\cdot)$ están indicados próximos al nodo correspondiente. El rótulo de cada arista indica la característica que fue removida en el pasaje de un nodo del nivel k a un nodo del nivel $k+1$. Los nodos con fondo sombreado indican que el límite fue actualizado. Los nodos con borde grueso indican que fue encontrado $J(\cdot) < B$, y si el nivel del nodo es $k < 2$ indican poda. Los nodos con borde punteado indican que fueron eliminados por podas.

En conclusión, el algoritmo *Branch and Bound* (ya sea en su primera versión u ordenado), a pesar de ser más eficiente en muchos casos que el algoritmo *Búsqueda Exhaustiva*, es costoso computacionalmente. Por lo tanto, sólo es utilizado cuando se desea obtener realmente una solución óptima. En otro caso puede resultar más conveniente utilizar algún algoritmo sub-óptimo que brinde buenos resultados a menor costo.

2.3.2. Métodos de selección sub-óptimos

Un método de selección de características sub-óptimo es aquel que utiliza un algoritmo de búsqueda sub-óptimo para intentar hallar el "mejor" subconjunto de características, de acuerdo a una función criterio determinada. Los algoritmos de búsqueda sub-óptimos no garantizan encontrar dicho subconjunto, en algunos casos pueden terminar proporcionando un subconjunto sub-óptimo de características.

Entre estos algoritmos se encuentran: los algoritmos de búsqueda secuencial, como por ejemplo *Sequential Forward Selection* y *Sequential Backward Selection*, y los algoritmos de búsqueda secuencial flotante, como *Sequential Forward Floating Selection* [28].

Sequential Forward Selection (SFS)

Sequential Forward Selection [29] es un algoritmo de búsqueda secuencial hacia adelante sub-óptimo. Parte del conjunto vacío y adiciona sucesivamente la característica más significativa en relación al subconjunto obtenido en la etapa anterior, por eso se dice que la búsqueda se realiza "hacia adelante".

Este algoritmo selecciona un subconjunto de tamaño d de un conjunto original de tamaño D, donde $d < D$. La selección se realiza mediante una búsqueda secuencial del "mejor" subconjunto. *SFS* parte del conjunto vacío de características. Se comienza seleccionando la "mejor" característica individual (aquella que produce el máximo valor de $J(\cdot)$). En cada una de las iteraciones siguientes se agrega la "mejor" característica (entendiendo por "mejor" característica a aquella que en combinación con las otras características ya seleccionadas maximiza la función criterio). Una vez que una característica es agregada al subconjunto ya no puede ser removida del mismo. El algoritmo finaliza cuando el subconjunto seleccionado posee d características. A continuación, mediante un ejemplo, se mostrará el procedimiento que realiza el algoritmo para buscar el subconjunto óptimo.

Dadas dos clases que poseen distribución normal. Con vectores de media μ_1 y μ_2 respectivamente y matriz de covarianza igual para ambas clases $\Sigma_1 = \Sigma_2 = \Sigma$.

$$\mu_1 = \begin{pmatrix} 2 \\ 3 \\ 3 \\ -2 \end{pmatrix} \mu_2 = \begin{pmatrix} -1 \\ 2 \\ 1 \\ -6 \end{pmatrix} \Sigma = \begin{pmatrix} 4 & 1 & -3 & 0 \\ 1 & 2 & -1 & 0 \\ -3 & -1 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

Se desea determinar, a partir del conjunto completo de características $Y = \{y_1, y_2, y_3, y_4\}$, el "mejor" subconjunto de 2 características. Se considera como función criterio la *Distancia Mahalanobis*. Se aplica el algoritmo *Sequential Forward Selection*, el cual proporciona como resultado el subconjunto $X = \{x_1, x_4\}$ con $J(X) = 7{,}58$.

La figura 2.5 muestra un gráfico de solución del algoritmo *Sequential Forward Selection* para el caso anteriormente planteado, donde $Y = \{1, 2, 3, 4\}$, $D = 4$ y $d = 2$. Como el algoritmo parte del conjunto vacío de características, el recuadro raíz está formado por el subconjunto vacío. Un subconjunto en un recuadro está formado por el subconjunto del recuadro padre con una, y solamente una, característica agregada. El rótulo de las aristas representa la característica agregada. El valor de $J(\cdot)$ está indicado próximo al recuadro correspondiente. Como una característica es agregada en cada iteración el procedimiento se repite d veces, habrá entonces d iteraciones o etapas (además de la etapa 0 que corresponde a la inicialización). La elección de las características que deben ser agregadas en cada etapa se hace de acuerdo al valor de la función criterio, será agregada al subconjunto la característica que maximice el valor de $J(\cdot)$. Por ejemplo, en la figura 2.5, en la etapa 1 se agrega la característica 4 porque produce el valor 5,33 que es el máximo de los valores producidos en esa etapa. En la etapa 2 se agrega la característica 1 ya que ésta, combinada con las demás características ya seleccionadas, maximiza la función criterio produciendo un valor de 7,58. Después de la etapa 2 el subconjunto seleccionado $\{1, 4\}$ posee el tamaño deseado (2 características), por lo tanto el algoritmo finaliza y devuelve dicho subconjunto como solución óptima (cabe destacar que no necesariamente será ésta la verdadera solución óptima).

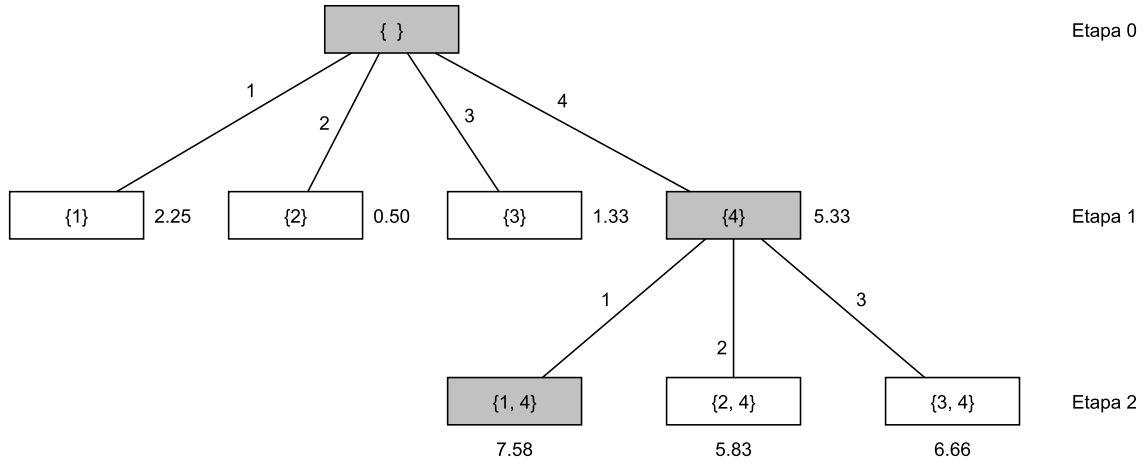


Figura 2.5: Gráfico de solución para el algoritmo *Sequential Forward Selection*, con $d = 2$ y $D = 4$. Dentro de cada recuadro se indica el subconjunto de características que está siendo evaluado. El valor de $J(\cdot)$ está indicado próximo al recuadro correspondiente. La etapa 0 corresponde a la inicialización, se inicializa el subconjunto en vacío. El rótulo de cada arista indica la característica que fue agregada al subconjunto seleccionado hasta el momento, en el pasaje de una etapa a la siguiente. Los recuadros con fondo sombreado indican que el subconjunto fue seleccionado por proporcionar el máximo valor de $J(\cdot)$ en la etapa.

Sequential Backward Selection (SBS)

Sequential Backward Selection [30] es un algoritmo de búsqueda secuencial hacia atrás sub-óptimo. Realiza el proceso inverso al *SFS*. Parte del conjunto completo de características y remueve sucesivamente las características menos significativas en relación al subconjunto obtenido en la etapa anterior, por eso se dice que la búsqueda se realiza "hacia atrás".

Este algoritmo selecciona un subconjunto de tamaño d de un conjunto original de tamaño D, donde $d < D$. La selección se realiza mediante una búsqueda secuencial del "mejor" subconjunto. *SBS* parte del conjunto completo de características. En cada una de las iteraciones se remueve la "peor" característica (entendiendo por "peor" característica a aquella que siendo removida del subconjunto seleccionado maximiza la función criterio). Una vez que una característica es removida del subconjunto ya no puede volver a formar parte del mismo. El algoritmo finaliza cuando el subconjunto seleccionado posee d características. A continuación, mediante un ejemplo, se mostrará el procedimiento que

realiza el algoritmo para buscar el subconjunto óptimo.

Dadas dos clases que poseen distribución normal. Con vectores de media μ_1 y μ_2 respectivamente y matriz de covarianza igual para ambas clases $\Sigma_1 = \Sigma_2 = \Sigma$.

$$\mu_1 = \begin{pmatrix} 2 \\ 3 \\ 3 \\ -2 \end{pmatrix} \mu_2 = \begin{pmatrix} -1 \\ 2 \\ 1 \\ -6 \end{pmatrix} \Sigma = \begin{pmatrix} 4 & 1 & -3 & 0 \\ 1 & 2 & -1 & 0 \\ -3 & -1 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

Se desea determinar, a partir del conjunto completo de características $Y = \{y_1, y_2, y_3, y_4\}$, el "mejor" subconjunto de 2 características. Se considera como función criterio la *Distancia Mahalanobis*. Se aplica el algoritmo *Sequential Backward Selection*, el cual proporciona como resultado el subconjunto $X = \{x_1, x_3\}$ con $J(X) = 26{,}33$.

La figura 2.6 muestra un gráfico de solución del algoritmo *Sequential Backward Selection* para el caso anteriormente planteado, donde $Y = \{1, 2, 3, 4\}$, $D = 4$ y $d = 2$. Como el algoritmo parte del conjunto completo de características, el recuadro raíz está formado por el subconjunto que incluye las cuatro características iniciales. Un subconjunto en un recuadro está formado por el subconjunto del recuadro padre con una, y solamente una, característica removida. El rótulo de las aristas representa la característica removida. El valor de $J(\cdot)$ está indicado próximo al recuadro correspondiente. Como una característica es removida en cada iteración el procedimiento se repite $D-d$ veces, habrá entonces $D-d$ iteraciones o etapas (además de la etapa 0 que corresponde a la inicialización). La elección de las características que deben ser removidas en cada etapa se hace de acuerdo al valor de la función criterio, será removida del subconjunto la característica menos significativa. Por ejemplo, en la figura 2.6, en la etapa 1 se remueve la característica 2 porque produce el valor 31,66 que es el máximo de los valores producidos en esa etapa. En la etapa 2 se remueve la característica 4 ya que ésta, siendo eliminada del subconjunto seleccionado, maximiza la función criterio produciendo un valor de 26,33. Después de la etapa 2 el subconjunto seleccionado $\{1, 3\}$ posee el tamaño deseado (2 características), por lo tanto el algoritmo finaliza y devuelve dicho subconjunto como solución óptima (en este caso el subconjunto resultante es óptimo, pero cabe destacar que el algoritmo podría haber proporcionado una solución sub-óptima).

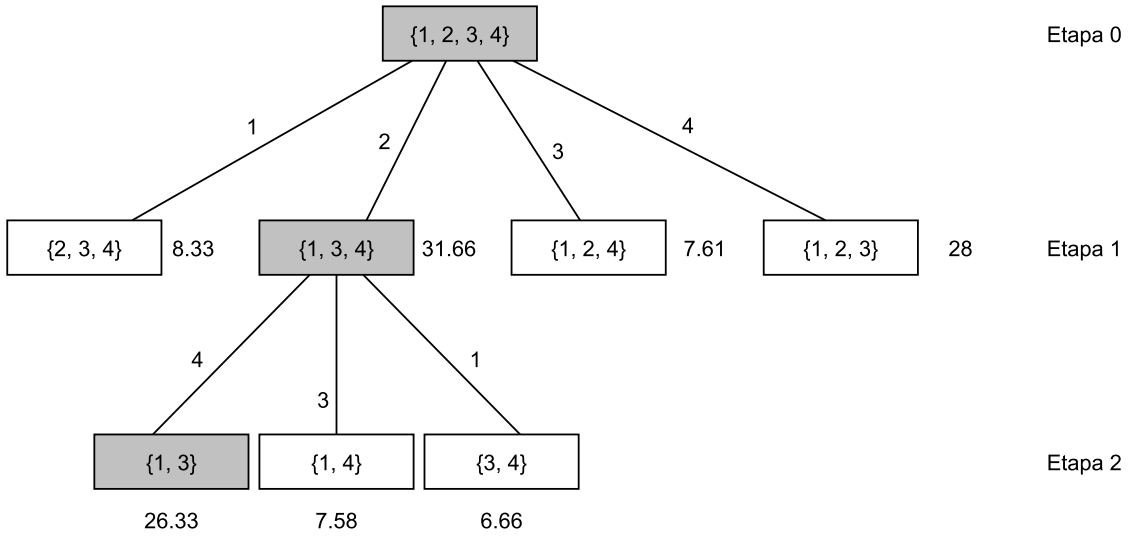


Figura 2.6: Gráfico de solución para el algoritmo *Sequential Backward Selection*, con $d = 2$ y $D = 4$. Dentro de cada recuadro se indica el subconjunto de características que está siendo evaluado. El valor de $J(\cdot)$ está indicado próximo al recuadro correspondiente. La etapa 0 corresponde a la inicialización, se inicializa el subconjunto con las D características originales. El rótulo de cada arista indica la característica que fue removida del subconjunto seleccionado hasta el momento, en el pasaje de una etapa a la siguiente. Los recuadros con fondo sombreado indican que el subconjunto fue seleccionado por proporcionar el máximo valor de $J(\cdot)$ en la etapa.

Sequential Forward Floating Selection (SFFS)

SFS y *SBS* son métodos rápidos y simples, pero que fácilmente resultan en una solución sub-óptima. Ambos métodos sufren el problema llamado *Efecto de Anidamiento* [31]. Esto significa que en el caso de *SFS* las características seleccionadas no pueden ser descartadas más tarde, y en el caso de *SBS* las características descartadas no pueden ser nuevamente seleccionadas. Esto resulta en subconjuntos de características anidados sin posibilidad de realizar correcciones en pasos posteriores, haciendo que el resultado final generalmente sea lejano al óptimo.

Una manera de abordar este problema es combinar *SFS* y *SBS* durante la ejecución, haciendo adiciones o eliminaciones de características. El algoritmo *Plus-l Take-Away-r (PTA(l,r))* [32] [33] realiza la adición de l características y a continuación la eliminación de r características sucesivamente, tal que $l \neq r$. Consiste en aplicar *SFS* l veces seguidas por

r pasos de *SBS*, repitiendo este ciclo fijo hasta que se alcanza el número de características requerido. Si $l > r$, el conjunto inicial debe ser el conjunto vacío. Si $l < r$, el conjunto inicial debe ser el conjunto Y. *SFS* es equivalente a *PTA(1,0)*, y *SBS* es equivalente a *PTA(0,1)*. Este algoritmo tiene mayor probabilidad de encontrar la solución óptima o una solución próxima a la óptima en relación a las versiones más simples de búsqueda secuencial, ya que permite un retroceso fijado definido por los valores de l o r. Por lo tanto, el algoritmo posee dos parámetros que precisan ser previamente definidos.

Para que el cambio de dirección de la búsqueda sea realizado automáticamente, sin uso de los parámetros, fueron propuestos los algoritmos de búsqueda secuencial flotante. Estos algoritmos consideran inclusión y exclusión de características controladas por el valor del criterio mismo. El retroceso en estos algoritmos es controlado dinámicamente, por lo tanto no es necesario ningún ajuste de parámetros.

Sequential Forward Floating Selection [31] [33] es un algoritmo que selecciona un subconjunto de tamaño d de un conjunto original de tamaño D, donde $d < D$. La selección se realiza mediante una búsqueda secuencial flotante del "mejor" subconjunto. *SFFS* parte del conjunto vacío de características. Consiste básicamente en aplicar después de cada inclusión de característica (iteración hacia adelante) una determinada cantidad de exclusiones condicionales de características (vueltas hacia atrás). En cada iteración una característica es adicionada y cero o más características son removidas, en tanto fueran encontrados subconjuntos mejores de los que fueron obtenidos hasta entonces. El algoritmo finaliza cuando el subconjunto seleccionado posee d características. A continuación, en la figura 2.7, se describe *SFFS* algorítmicamente. Para ello, se hacen primero algunas suposiciones.

Dado el conjunto completo de D características $Y = \{y_j \mid j = 1, ..., D\}$, se desean seleccionar k características ($k < D$) para formar el conjunto $X_k = \{x_j \mid j = 1, ..., k, \;\; x_j \in Y\}$ que optimice la función criterio $J(X_k)$ correspondiente.

Inicialización

$X_0 := \varnothing$

$k := 0$ (se puede comenzar con $k = 2$ por aplicar *SFS* dos veces)

Paso 1 - Inclusión

$x^+ := \arg \max_{x \in Y - X_k} J(X_k + x)$ (x^+ es la característica más significativa con respecto a X_k)

$X_{k+1} := X_k + x^+$

$k := k + 1$

Paso 2 - Exclusión condicional

$x^- := \arg \max_{x \in X_k} J(X_k - x)$ (x^- es la característica menos significativa en X_k)

if $J(X_k - \{x^-\}) > J(X_{k-1})$ then

$X_{k-1} := X_k - x^-$

$k := k - 1$

ir a Paso 2

else

ir a Paso 1

Terminación

Parar cuando k es igual al número de características requerido

Figura 2.7: Algoritmo *SFFS*

Como contrapartida del algoritmo *Sequential Forward Floating Selection* existe un algoritmo llamado *Sequential Backward Floating Selection (SBFS)* [31] [33]. Éste parte del conjunto completo de características y realiza una eliminación y cero o más adiciones en cada iteración.

SBFS es un algoritmo de búsqueda secuencial flotante que excluye características por medio de la aplicación del procedimiento *SBS* básico. Comienza por el conjunto completo de características. Realiza una serie de inclusiones condicionales sucesivas de la característica más significativa entre las características disponibles, si esto produce una mejora.

2.4. Librería de funciones de selección *libSelección*

Es una librería de funciones de selección de características implementada en Matlab [34], que incluye algoritmos de selección óptimos y sub-óptimos y puede ser utilizada con diferentes tipos de datos. En la figura 2.8 se puede ver cómo está conformada esta librería.

libSelección contiene las siguientes funciones que implementan métodos de selección óptimos:

- `be.m`: *Búsqueda Exhaustiva*
- `bb.m`: *Branch and Bound*

Además contiene las siguientes funciones que implementan métodos de selección sub-óptimos:

- `sfs.m`: *Sequential Forward Selection*
- `sbs.m`: *Sequential Backward Selection*
- `sffs.m`: *Sequential Forward Floating Selection*

También, *libSelección* contiene la implementación de una función criterio:

- `mah.m`: *Distancia Mahalanobis*

Se incluyen en *libSelección* una serie de scripts que permiten realizar la carga de diferentes datos para probar las funciones. Estos son:

- `cargaSel1.m`
- `cargaSel2.m`
- `cargaSel3.m`
- `cargaSel4.m`
- `cargaSel5.m`
- `cargaSel6.m`

- `cargaSel7.m`

En todos los casos los datos que se cargan corresponden a: un vector de media para la clase 1, un vector de media para la clase 2, una matriz de covarianza asociada a las dos clases y la cantidad de características a seleccionar. En los casos 1, 2, 3 y 4 se utilizan datos artificiales. En los casos 5, 6 y 7 los datos fueron calculados a partir de los patrones de muestra contenidos en la base de datos *Iris Plants* [35].

Para realizar una prueba rápida de cómo funciona la librería se puede utilizar el script de prueba que incluye la misma:

- `probarSel.m`

Mediante la invocación a este script se realiza la ejecución de un caso de prueba. En primer lugar se cargan los datos, luego se aplican los métodos de selección de características sobre dichos datos y finalmente se observan los resultados.

Si se quieren realizar otras pruebas sólo basta con cambiar dentro del script de prueba la invocación al script de carga de datos, para así probar los resultados de los métodos sobre datos distintos.

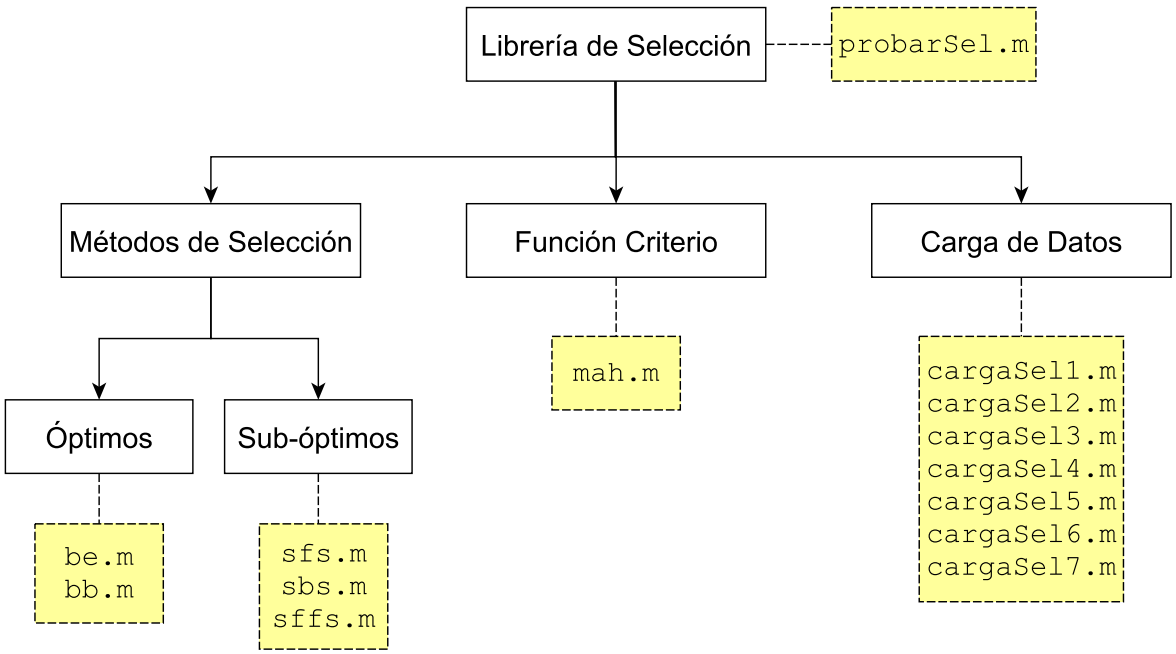


Figura 2.8: Librería de funciones de selección *libSelección*.

Capítulo 3

Texturas

3.1. El concepto de textura

La textura es una característica importante de toda imagen, ya que es una fuente valiosa de información para el análisis y comprensión de dicha imagen. Se utiliza principalmente en segmentación, identificación de objetos o regiones de interés y obtención de forma [36][37][38]. El uso de la textura para identificar una imagen proviene de la habilidad innata de los humanos para reconocer diferencias texturales. Por medio de la visión y el tacto, el ser humano es capaz de distinguir en forma intuitiva diversos tipos de textura.

La textura puede ser evaluada cualitativamente como fina o suave, gruesa, rizada o rugosa, alineada o regular, desordenada o irregular y ondulada. En la figura 3.1 se pueden diferenciar algunos tipos de textura.

La textura es una propiedad de todas las superficies, por ejemplo: los granos en la madera o la trama de un tejido textil. Contiene información importante acerca de la colocación estructural de la superficie y su relación con el entorno. Aunque es muy fácil para el observador humano reconocerla y describirla en términos empíricos, este concepto no posee una definición formal [39].

La idea de textura se basa en la distribución espacial de tonos de gris. Se puede definir como el arreglo de píxeles en una región de interés que son percibidos por el sistema visual humano como similares a distintos materiales.

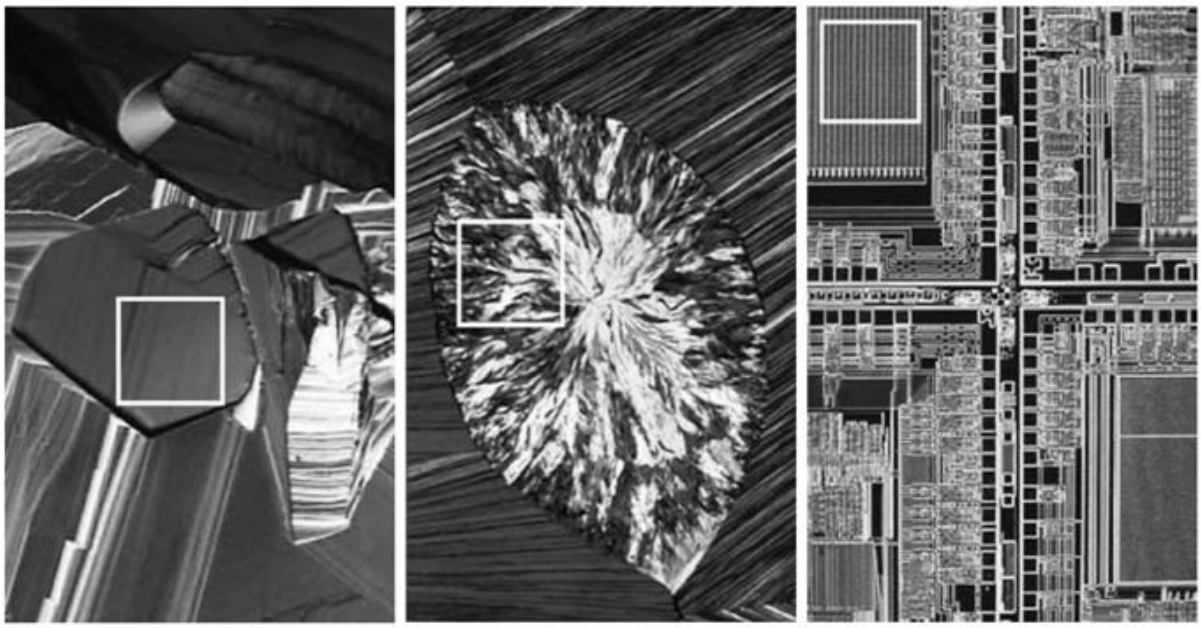


Figura 3.1: Texturas dentro del recuadro blanco, de izquierda a derecha: suave (superconductor), rugosa (colesterol humano) y regular (microprocesador) [40].

Debido a que la textura de una imagen brinda información útil para propósitos de discriminación es importante desarrollar características o descriptores de textura [39][41]. Los descriptores de textura permiten la cuantificación del contenido de la textura presente en la imagen o en una región de ella. Estos descriptores proporcionan una medida de propiedades tales como suavidad, grosor y regularidad. Existen diferentes métodos que permiten generar descriptores de textura [42], entre ellos se distinguen:

- *Métodos estadísticos*: Los métodos estadísticos se basan en el histograma de la imagen (primer orden) y en las matrices de co-ocurrencia de la imagen (segundo orden) [43][44][45]. Estos métodos serán estudiados más adelante, en la sección 3.2.

- *Métodos estructurales*: Los métodos estructurales se caracterizan porque en la definición de textura están presentes "elementos de la textura" o primitivas. El método de análisis normalmente depende de las propiedades geométricas de los elementos de la textura. Una vez que esos elementos de la textura se identifican en la imagen, hay dos enfoques principales para analizar la textura. Uno calcula las propiedades estadísticas de los elementos de la textura extraídos y utiliza éstos como descriptores de la textura. El otro enfoque intenta extraer la regla de ubicación que describe

la textura, para ello puede incluir métodos sintácticos o geométricos de análisis de textura. En la figura 3.2 se observan tres texturas distintas y sus respectivas primitivas, puede observarse en cada caso una primitiva diferente. El inconveniente que poseen los métodos estructurales es que no todas las texturas tienen un elemento que se repite.

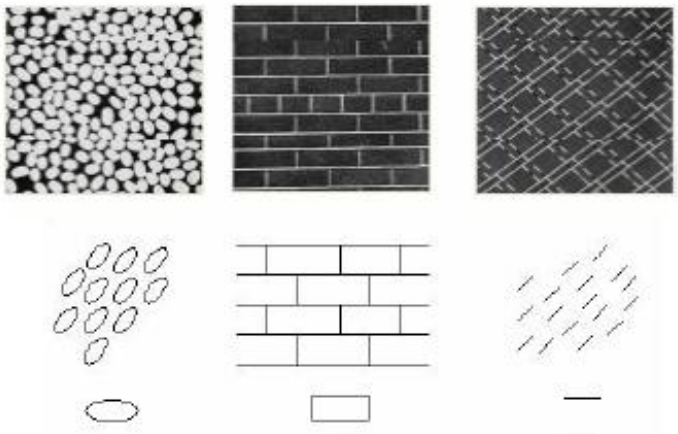


Figura 3.2: Texturas distintas junto a sus respectivas primitivas.

- *Métodos espectrales*: Los métodos espectrales utilizan la transformada de Fourier y son usados primariamente para detectar la periodicidad global presente en la imagen. En la figura 3.3 se observa un patrón de textura y el espectro de su transformada, el cual tiene un sólo pico cada 90º debido a la periodicidad del patrón. El inconveniente que presentan los métodos espectrales se observa cuando su implementación se ve restringida por la poca información que se puede extraer cuando las texturas son heterogéneas o de carácter aleatorio.

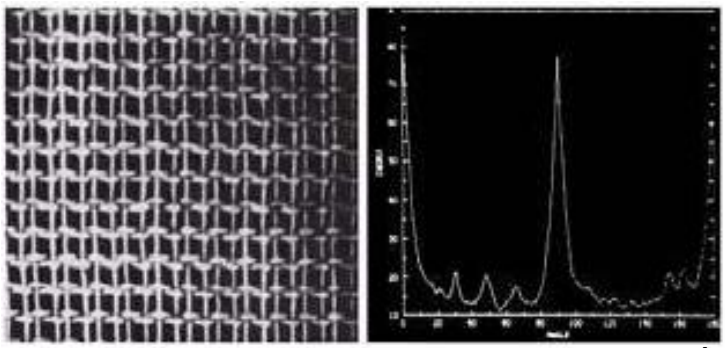


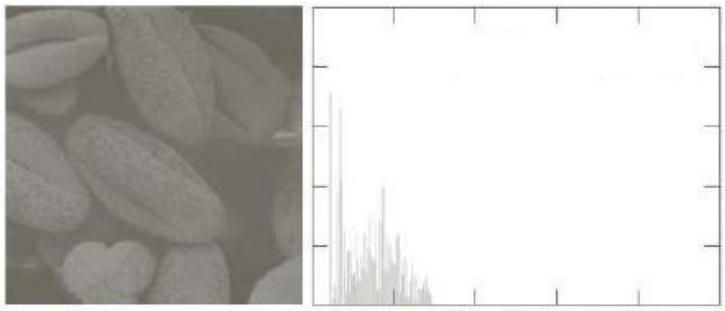
Figura 3.3: Una textura junto al espectro de su transformada [40].

3.2. Descriptores estadísticos

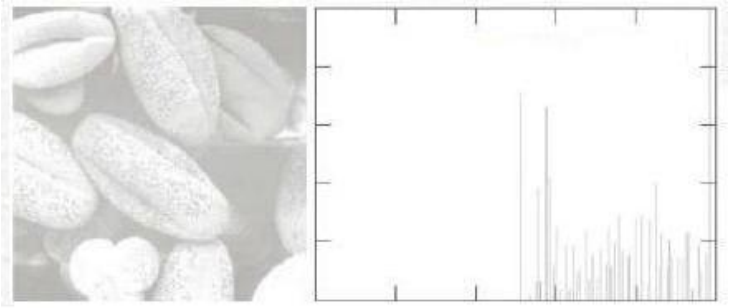
Los métodos estadísticos utilizan los rasgos estadísticos de la textura de la imagen, la textura es modelada como una función de los niveles de gris. Desde el punto de vista estadístico se observa que una textura puede definirse por un conjunto de datos estadísticos extraídos de un gran conjunto de propiedades locales de la imagen. Según Haralick [46], la textura de una imagen de niveles de gris está caracterizada por dos dimensiones. La primera dimensión comprende los parámetros de la distribución de los valores de gris que generan un procedimiento estadístico simple llamado *de primer orden*, donde están incluidos los momentos estadísticos correspondientes al histograma de los niveles de gris de la región y la entropía media. Estos rasgos de textura poseen la limitación de no contener información respecto a la posición relativa que tienen los píxeles entre sí. Una alternativa para obtener este tipo de información en el proceso de análisis de textura es considerar no solamente la distribución de las intensidades, sino también las posiciones de los píxeles con iguales valores de intensidad o cercanamente igual. Es por eso que la segunda dimensión comprende las relaciones y dependencias que existen entre los niveles de gris vecinos o entre regiones de niveles de gris de la imagen, lo cual genera otro procedimiento estadístico llamado *de segundo orden*. Esta segunda dimensión es la base primaria para el desarrollo de los métodos de análisis de textura que utilizan la matriz de co-ocurrencia de niveles de gris [39].

3.2.1. Descriptores estadísticos de primer orden

Los descriptores estadísticos de primer orden se basan en el histograma de la imagen. En el histograma los niveles de intensidad de gris están representadas a lo largo del eje x y suelen ir de 0 a 255, mientras que el número de ocurrencias para cada nivel de intensidad de gris se representa en el eje y. En las figuras 3.4 y 3.5 se muestran distintos ejemplos; puede verse una misma imagen pero con diferente luz y contraste en cada caso, y cómo esto se refleja en sus respectivos histogramas. A continuación se da una definición formal del histograma de una imagen y de algunas medidas que pueden obtenerse a partir del mismo.

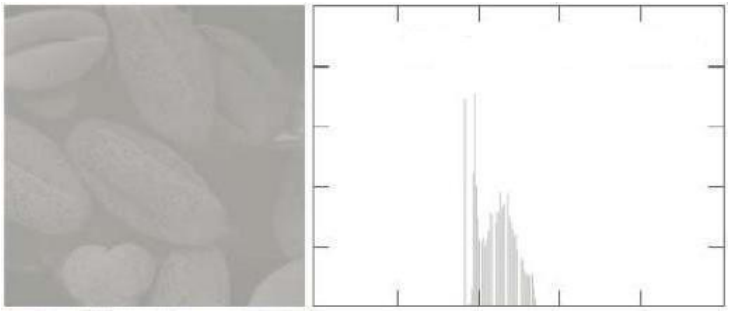


(a) Histograma de una imagen oscura.

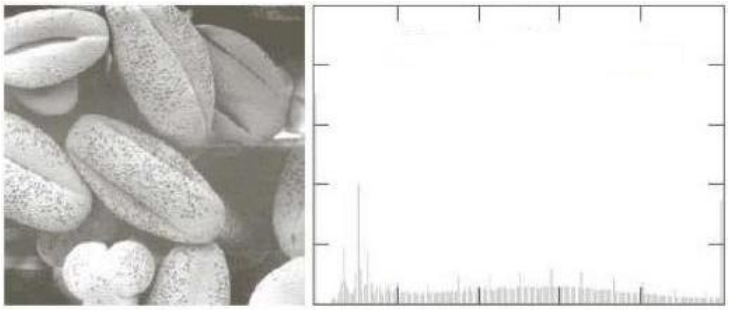


(b) Histograma de una imagen clara.

Figura 3.4: Histogramas de imágenes con distinta luz.



(a) Histograma de una imagen con bajo contraste.



(b) Histograma de una imagen con alto contraste.

Figura 3.5: Histogramas de imágenes con distinto contraste.

Sea la matriz I la imagen analizada. El valor del elemento $I(m,n)$ corresponde al valor del píxel en la m-ésima fila y n-ésima columna de la imagen, siendo que $m = 0, 1, ..., M-1$ y $n = 0, 1, ..., N-1$. De esta manera, si G es el número de niveles de intensidad de gris de la imagen, $I(m,n) = 0, 1, ..., G-1$. El histograma $h(\cdot)$ de la imagen I se define por:

$$h(i) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(i, I(m,n)), \text{ donde } i = 0, 1, ..., G-1,$$

$$\delta(i,j) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j. \end{cases}$$

El valor de $h(i)$ corresponde al número de ocurrencias del nivel de intensidad de gris i en la imagen. La función de probabilidad $p(i)$, que representa la probabilidad de ocurrencia del nivel de intensidad de gris i, se obtiene dividiendo $h(i)$ por el número total píxeles de la imagen:

$$p(i) = h(i)/(N \cdot M).$$

Existen diversas medidas que pueden ser extraídas del histograma, algunas de las más utilizadas son las siguientes:

- *Media*: Representa el nivel de intensidad de gris medio en el histograma.

$$\mu = \sum_{i=0}^{G-1} i \cdot p(i)$$

- *Desviación Estándar*: Representa el nivel de contraste medio en el histograma.

$$\sigma = \sqrt{\sigma^2}$$

  Donde:

$$\sigma^2 = \sum_{i=0}^{G-1} (i-\mu)^2 \cdot p(i)$$

- *Asimetría*: Corresponde al grado de simetría. Si $\gamma_1 = 0$, el histograma es simétrico en relación a la media. Caso contrario, la distribución se concentra a la izquierda o a la derecha de la media.

$$\gamma_1 = \frac{1}{\sigma^3} \cdot \sum_{i=0}^{G-1} (i-\mu)^3 \cdot p(i)$$

- *Kurtosis*: Se relaciona con el "aplanamiento" del histograma. Si $\gamma_2 = 3$, el aplanamiento es el mismo al de una distribución normal. Si $\gamma_2 > 3$, el histograma es más alto que una normal. Si $\gamma_2 < 3$, el histograma es más bajo que una normal.

$$\gamma_2 = (\frac{1}{\sigma^4} \cdot \sum_{i=0}^{G-1} (i-\mu)^4 \cdot p(i))$$

- *Energía*: Mide la presencia de valores altos, en relación a los demás valores, en el histograma.

$$E = \sum_{i=0}^{G-1} (p(i))^2$$

- *Entropía*: Mide la uniformidad del histograma.

$$H = -\sum_{i=0}^{G-1} p(i) \cdot log_2(p(i))$$

3.2.2. Descriptores estadísticos de segundo orden

Los descriptores estadísticos de segundo orden se basan en las matrices de co-ocurrencia de la imagen. La matriz de co-ocurrencia de niveles de gris (Gray Level Co-occurrence Matrix - GLCM) [39] es un método que se utiliza para medir la textura. Esta matriz es un histograma de dos dimensiones de los niveles de gris para un par de píxeles, el pixel de referencia y el pixel vecino.

Los descriptores estadísticos de segundo orden asumen que la información de la textura en una imagen está contenida en la relación espacial que los niveles de gris tienen entre ellos. Las relaciones están especificadas en las matrices de co-ocurrencia espaciales (o de niveles de gris) que son calculadas entre los píxeles vecinos, para una dirección especifica o para todas las direcciones, en una ventana móvil dentro de la imagen.

La relación espacial entre un pixel de referencia y su vecino puede ser en cualquiera de las ocho direcciones posibles, como se observa en la figura 3.6. Por lo general se calculan sólo cuatro direcciones y las restantes se obtienen con la matriz simétrica. Para lograr

una relación "espacialmente invariante" se eligen las direcciones 0°, 45°, 90° y 135° y se promedian.

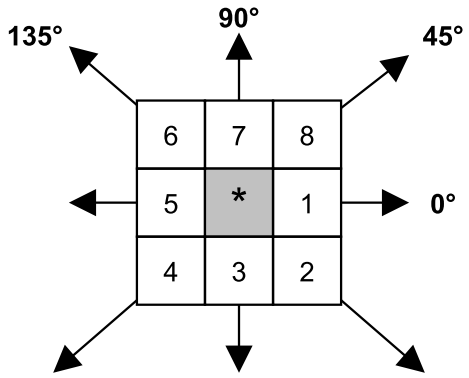
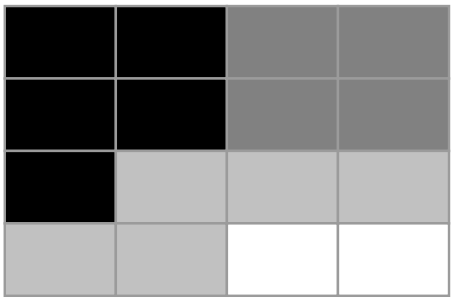


Figura 3.6: Información espacial del pixel * [39].

En la figura 3.7 se representa una imagen de 4 x 4 píxeles junto a los valores correspondientes a los niveles de gris de dicha imagen, se distinguen cuatro valores de niveles de gris: 0, 1, 2 y 3.



0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Figura 3.7: Imagen de 4 x 4 píxeles con cuatro valores de niveles de gris (0, 1, 2 y 3) [39].

La forma general de cualquier matriz de co-ocurrencia para una imagen con valores de niveles de gris de 0 a 3 puede verse en la figura 3.8.

Niveles de Gris	0	1	2	3
0	#(0,0)	#(0,1)	#(0,2)	#(0,3)
1	#(1,0)	#(1,1)	#(1,2)	#(1,3)
2	#(2,0)	#(2,1)	#(2,2)	#(2,3)
3	#(3,0)	#(3,1)	#(3,2)	#(3,3)

Figura 3.8: Forma general de cualquier matriz de co-ocurrencia para una imagen con valores de niveles de gris de 0 a 3, $\sharp(i,j)$ representa el número de veces que niveles de gris i y j han sido vecinos [39].

En la figura 3.9 pueden verse las cuatro matrices de co-ocurrencia simétricas obtenidas para la imagen de la figura 3.7. A continuación se da una definición formal de las matrices de co-ocurrencia y de algunas medidas que pueden obtenerse a partir de las mismas.

0°

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

(a) Matriz simétrica 0°

45°

4	1	0	0
1	2	2	0
2	2	4	1
0	0	1	0

(b) Matriz simétrica 45°

90°

6	0	2	0
0	4	2	0
2	2	2	2
0	0	2	0

(c) Matriz simétrica 90°

135°

2	1	3	0
1	2	1	0
3	1	0	2
0	0	2	0

(d) Matriz simétrica 135°

Figura 3.9: Matrices de co-ocurrencia simétricas de la imagen de la figura 3.7 [39].

Siendo G el número de niveles de intensidad de gris de la imagen, cada matriz de co-ocurrencia tiene tamaño G x G. La comparación de los píxeles se hace de acuerdo con un desplazamiento horizontal d_x y vertical d_y. Sea $C_{d_x d_y}$ una matriz de co-ocurrencia para los desplazamientos d_x y d_y. El elemento $C_{d_x d_y}(i,j)$ corresponde al número de ocurrencias de píxeles de nivel de intensidad de gris j situados a un desplazamiento horizontal d_x y vertical d_y de píxeles de nivel de intensidad de gris i. O sea, la ocurrencia es considerada sólo cuando $I(m,n) = i$ y $I(m+d_y, n+d_x) = j$. La generación de matrices de co-ocurrencia se define por:

$$C_{d_x d_y}(i,j) = \sum_{m=-min(0,d_y)}^{M-1-max(0,d_y)} \sum_{n=-min(0,d_x)}^{N-1-max(0,d_x)} \delta(i, I(m,n)) \cdot \delta(j, I(m+d_y, n+d_x)).$$

Las funciones $min(\cdot)$ y $max(\cdot)$, presentes en los límites de las sumatorias, son necesarias para garantizar que los píxeles $I(m,n)$ y $I(m+d_y, n+d_x)$ pertenezcan a la imagen. Las características extraídas de las matrices con desplazamientos opuestos son iguales o muy próximas, por eso se realiza la suma de esas matrices para obtener una matriz simétrica $(C_{d_x d_y} + C_{-d_x -d_y})$. Dicha matriz simétrica debe ser normalizada para obtener la probabilidad de ocurrencia de pares de píxeles para cada posicionamiento. La matriz normalizada $R_{d_x d_y}$ se obtiene por:

$$R_{d_x d_y}(i,j) = \frac{C_{d_x d_y}(i,j)}{\sum_{m=0}^{G-1} \sum_{n=0}^{G-1} C_{d_x d_y}(m,n)}.$$

Existen diversas medidas que pueden ser extraídas de la matriz de co-ocurrencia. Las siguientes fórmulas son válidas para matrices de co-ocurrencia simétricas [47]:

- *Energía*: Brinda una idea de la suavidad que posee la textura.

$$C_1 = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (R_{d_x d_y}(i,j))^2$$

- *Contraste*: Proporciona información acerca de las variaciones bruscas de tono en la imagen.

$$C_2 = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} (i-j)^2 \cdot R_{d_x d_y}(i,j)$$

- *Correlación*: Mide la similitud entre píxeles vecinos.

$$C_3 = \frac{\sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i-\mu)\cdot(j-\mu)\cdot R_{d_x d_y}(i,j)}{\sigma^2}$$

  Donde:

$$\mu = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1} i \cdot R_{d_x d_y}(i,j)$$

$$\sigma^2 = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}(i-\mu)^2 \cdot R_{d_x d_y}(i,j)$$

- *Homogeneidad*: Provee información sobre la regularidad de la textura.

$$C_4 = \sum_{i=0}^{G-1}\sum_{j=0}^{G-1}\frac{R_{d_x d_y}(i,j)}{1+(i-j)^2}$$

- *Entropía*: Es una medida de la aleatoriedad contenida en la matriz de co-ocurrencia.

$$C_5 = -\sum_{i=0}^{G-1}\sum_{j=0}^{G-1} R_{d_x d_y}(i,j)\cdot log_2(R_{d_x d_y}(i,j))$$

3.3. Librería de funciones de texturas *libTexturas*

Es una librería de funciones implementada en Matlab [34] que permite obtener a partir de una imagen digital los principales descriptores estadísticos de textura de primer y segundo orden. En la figura 3.10 se puede ver cómo está conformada esta librería.

libTexturas contiene las siguientes funciones que implementan descriptores estadísticos de primer orden:

- `histograma.m`: Calcula el histograma de la imagen de entrada.
- `asimetria1.m`: Asimetría.
- `desviacion1.m`: Desviación Estándar.
- `energia1.m`: Energía.
- `entropia1.m`: Entropía.

- `kurtosis1.m`: Kurtosis.
- `media1.m`: Media.
- `varianza1.m`: Varianza.
- `descriptores1.m`: Proporciona los descriptores estadísticos de primer orden de la imagen de entrada.

Además, *libTexturas* contiene las siguientes funciones que implementan descriptores estadísticos de segundo orden:

- `matrizcooc.m`: Calcula las cuatro matrices de co-ocurrencia (simétricas y normalizadas, con distancia de 1 pixel, en las direcciones 0°, 45°, 90° y 135°) de la imagen de entrada.
- `contraste2.m`: Contraste.
- `correlacion2.m`: Correlación.
- `energia2.m`: Energía.
- `entropia2.m`: Entropía.
- `homogeneidad2.m`: Homogeneidad.
- `media2.m`: Media.
- `varianza2.m`: Varianza.
- `descriptores2.m`: Proporciona los descriptores estadísticos de segundo orden de la imagen de entrada.

Se incluyen en *libTexturas* una serie de scripts que permiten realizar la carga de diferentes datos para probar las funciones. Estos son:

- `cargaTex1.m`
- `cargaTex2.m`

- `cargaTex3.m`
- `cargaTex4.m`
- `cargaTex5.m`
- `cargaTex6.m`
- `cargaTex7.m`
- `cargaTex8.m`

En todos los casos los datos que se cargan corresponden a: la matriz que representa una imagen y la cantidad de niveles de intensidad de gris de dicha imagen. En los casos 1, 2, 3 y 4 las imágenes utilizadas fueron extraídas de la Base de Datos de imágenes de texturas *Brodatz* [48]. Y en los casos 5, 6, 7 y 8 las imágenes utilizadas fueron extraídas de una Base de Datos de texturas propia.

Para realizar una prueba rápida de cómo funciona la librería se puede utilizar el script de prueba que incluye la misma:

- `probarTex.m`

Mediante la invocación a este script se realiza la ejecución de un caso de prueba. En primer lugar se cargan los datos, luego se generan los descriptores estadísticos sobre dichos datos y finalmente se observan los resultados.

Si se quieren realizar otras pruebas sólo basta con cambiar dentro del script de prueba la invocación al script de carga de datos, para así probar los resultados de los descriptores sobre datos distintos.

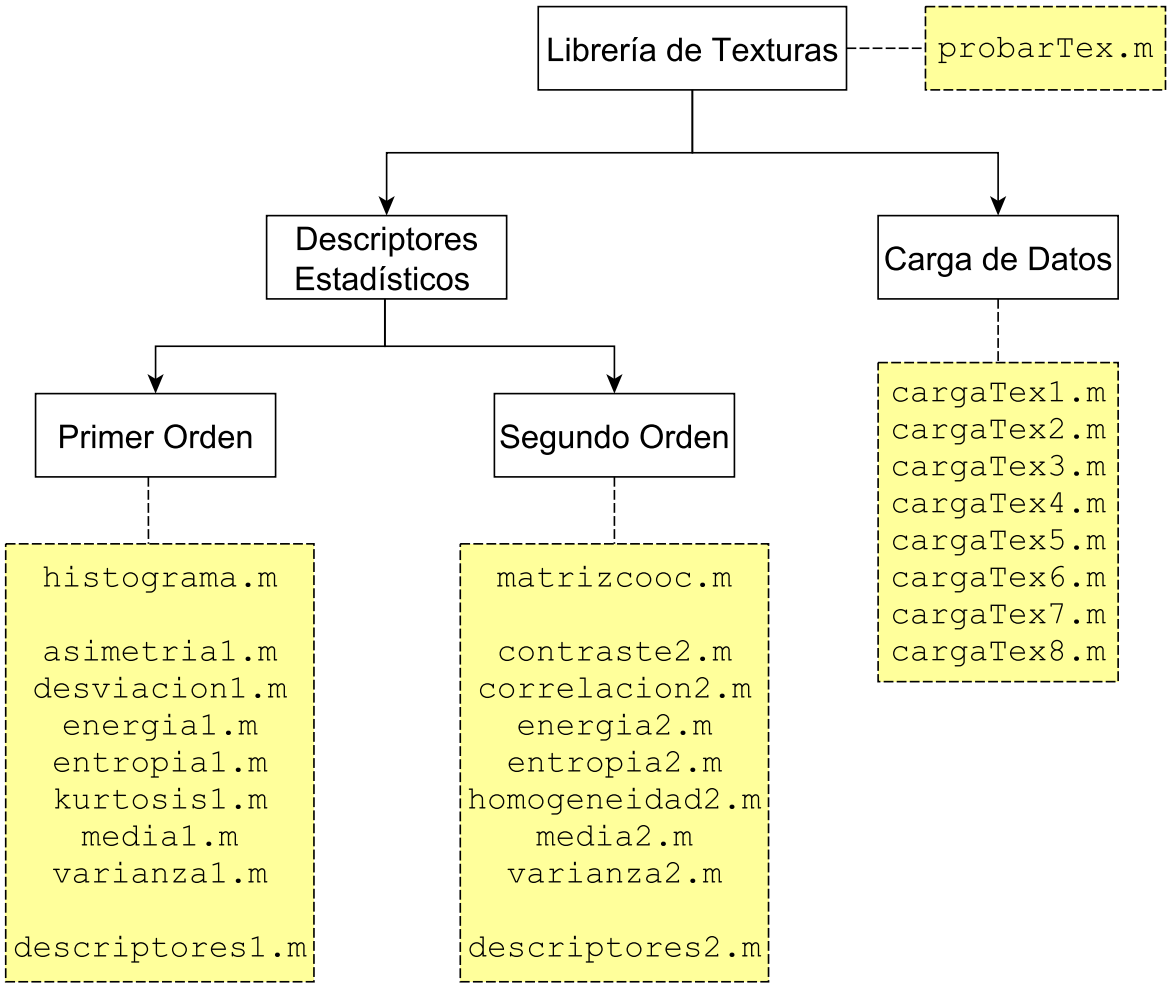


Figura 3.10: Librería de funciones de texturas *libTexturas*.

Capítulo 4

Selección de Características de Textura

4.1. Seleccionar características de textura

Un modelo general de un sistema de reconocimiento de patrones está compuesto por un conjunto de etapas de procesamiento denominadas *Sensor*, *Reductor de Dimensión* y *Clasificador*, como se mencionó anteriormente en la sección 2.1.

Para que la información de una imagen digital pueda ser utilizada en un sistema de clasificación automática se requiere de una etapa adicional denominada *Generación de Características o Descriptores* [49]. Una alternativa para caracterizar los objetos de interés en una imagen digital es mediante descriptores de textura [50]. La textura de un objeto en la imagen está asociada a la distribución de los niveles de gris en dicha región. Un abordaje posible para la generación de descriptores de textura es mediante descriptores estadísticos de primer y segundo orden, obtenidos a partir de las características de la región de interés [39].

A medida que el número de descriptores aumenta también crece la complejidad computacional del sistema, y la posibilidad de tener descriptores correlacionados que no contribuyan a la capacidad de discriminación entre objetos diferentes. En un sistema de reconocimiento de patrones la etapa de reducción de dimensión tiene un impacto funda-

mental en el costo de adquisición, el rendimiento y la complejidad del sistema [7]. En particular los métodos de selección de características, usados en la etapa de reducción de dimensión, permiten determinar un subconjunto de d características de un conjunto inicial más numeroso D optimizando un determinado criterio J. En el rendimiento ideal de un sistema de clasificación, considerando la probabilidad de error del mismo, tiene influencia determinante el compromiso existente en la determinación del número de características óptimo en su diseño.

4.2. Modelado de un sistema de RP aplicado a clasificación de texturas

Se propone un modelo de un sistema de reconocimiento de patrones (RP) aplicado a clasificación de texturas, que tiene como propósito combinar en su desarrollo los temas estudiados en las capítulos 2 y 3. Para este modelo resulta necesario generar características de textura, por lo cual se obtienen descriptores estadísticos de primer y segundo orden a partir de una imagen. Además es útil en este modelo la realización de una etapa de reducción de dimensión, por lo que se aplican métodos de selección de características a los patrones originales.

El modelo propuesto consta de cinco etapas:

- *Carga de datos de entrada*: Se cargan dos imágenes/clases a procesar.
- *Generación de características de textura*: A partir de dos imágenes/clases se obtienen los patrones de muestra correspondientes a cada imagen/clase. Para ello se generan 10 regiones de interés de cada imagen. De cada región de interés se calculan los descriptores estadísticos de primer orden (media, desviación, asimetría, kurtosis, energía y entropía) y de segundo orden (energía, contraste, correlación, homogeneidad y entropía), los cuales conforman un patrón de muestra. Se obtiene un total de 20 patrones de muestra, 10 correspondientes a la primera clase y 10 a la segunda, donde cada patrón está formado por 11 características.

- *Selección de características*: A partir de los patrones de muestra obtenidos en la etapa anterior se calculan los datos de entrada necesarios para los métodos de selección implementados: media de la primera clase, media de la segunda clase y matriz de covarianza asociada a ambas clases. Luego se seleccionan las mejores características utilizando un método de selección.

- *Clasificación*: Se realiza la clasificación de los patrones, considerando sólo las características seleccionadas. Para lograr esto se toma una mitad de los patrones de la primera clase como patrones de muestra y la otra mitad como patrones de entrenamiento, lo mismo se hace con la segunda clase. De cada patrón se consideran sólo las características seleccionadas. Luego se realiza la clasificación de los patrones de muestra utilizando el método de los k vecinos más próximos.

- *Almacenamiento de resultados*: Los resultados obtenidos se almacenan en el archivo de texto "`resultados.txt`" de la siguiente manera: el nombre de cada imagen utilizada, el nombre del método de selección de características utilizado, las características seleccionadas y la clasificación obtenida.

4.3. Librería para la realización de pruebas *libSelTex*

Es una librería de funciones implementada en Matlab [34] para la realización de pruebas, que representa las etapas básicas del modelo de reconocimiento de patrones aplicado a clasificación de texturas. En la figura 4.1 se puede ver cómo está conformada esta librería.

libSelTex contiene una función que permite ejecutar todas las etapas del modelo planteado:

- `ejecutarSelTex.m`

Para llevar a cabo cada una de las etapas del modelo, *libSelTex* contiene las siguientes funciones:

1. *CARGA DE DATOS*

   Scripts que permiten realizar la carga de diferentes datos:

- `cargaSelTex1.m`
- `cargaSelTex2.m`
- `cargaSelTex3.m`
- `cargaSelTex4.m`
- `cargaSelTex5.m`
- `cargaSelTex6.m`
- `cargaSelTex7.m`
- `cargaSelTex8.m`

En todos los casos los datos que se cargan corresponden a: los nombres de dos imágenes junto a la cantidad de niveles de intensidad de gris de dichas imágenes y el método de selección de características a utilizar junto a la cantidad de características a seleccionar. En los casos 1, 2, 3 y 4 las imágenes utilizadas fueron extraídas de la Base de Datos de imágenes de texturas *Brodatz* [48]. Y en los casos 5, 6, 7 y 8 las imágenes utilizadas fueron extraídas de una Base de Datos de texturas propia.

2. *TEXTURAS*

Funciones que implementan descriptores estadísticos de primer orden:

- `histograma.m`: Calcula el histograma de la imagen de entrada.
- `asimetria1.m`: Asimetría.
- `desviacion1.m`: Desviación Estándar.
- `energia1.m`: Energía.
- `entropia1.m`: Entropía.
- `kurtosis1.m`: Kurtosis.
- `media1.m`: Media.
- `varianza1.m`: Varianza.
- `descriptores1.m`: Proporciona los descriptores estadísticos de primer orden de la imagen de entrada.

Funciones que implementan descriptores estadísticos de segundo orden:

- `matrizcooc.m`: Calcula las cuatro matrices de co-ocurrencia (simétricas y normalizadas, con distancia de 1 pixel, en las direcciones 0°, 45°, 90° y 135°) de la imagen de entrada.
- `contraste2.m`: Contraste.
- `correlacion2.m`: Correlación.
- `energia2.m`: Energía.
- `entropia2.m`: Entropía.
- `homogeneidad2.m`: Homogeneidad.
- `media2.m`: Media.
- `varianza2.m`: Varianza.
- `descriptores2.m`: Proporciona los descriptores estadísticos de segundo orden de la imagen de entrada.

Función para obtener muestras:

- `muestreo.m`: Proporciona patrones de muestra a partir de una imagen.

3. *SELECCIÓN*

Funciones que implementan métodos de selección óptimos:

- `be.m`: *Búsqueda Exhaustiva*

Funciones que implementan métodos de selección sub-óptimos:

- `sfs.m`: *Sequential Forward Selection*
- `sbs.m`: *Sequential Backward Selection*
- `sffs.m`: *Sequential Forward Floating Selection*

Función criterio:

- `mah.m`: *Distancia Mahalanobis*

Función para calcular datos:

- `calcularDatos.m`: A partir de los patrones de muestra calcula los datos de entrada necesarios para los métodos de selección.

4. *CLASIFICACIÓN*

   Función para clasificar:

   - `clasificar.m`: Realiza la clasificación de los patrones de muestra utilizando el método de los k vecinos más próximos.

5. *ALMACENAMIENTO*

   Función para almacenar los resultados:

   - `guardar.m`: Guarda en un archivo de texto los resultados obtenidos.

   Archivo de resultados:

   - `resultados.txt`: Archivo de texto que almacena los resultados obtenidos.

Para realizar una prueba rápida de cómo funciona la librería se puede utilizar el script de prueba que incluye la misma:

- `probarSelTex.m`

Mediante la invocación a este script se realiza la ejecución de un caso de prueba. En primer lugar se cargan los datos, luego se aplican todas las etapas del modelo de reconocimiento de patrones sobre dichos datos y finalmente se observan los resultados.

Si se quieren realizar otras pruebas sólo basta con cambiar dentro del script de prueba la invocación al script de carga de datos, para así probar los resultados que se obtienen con distintos datos de entrada.

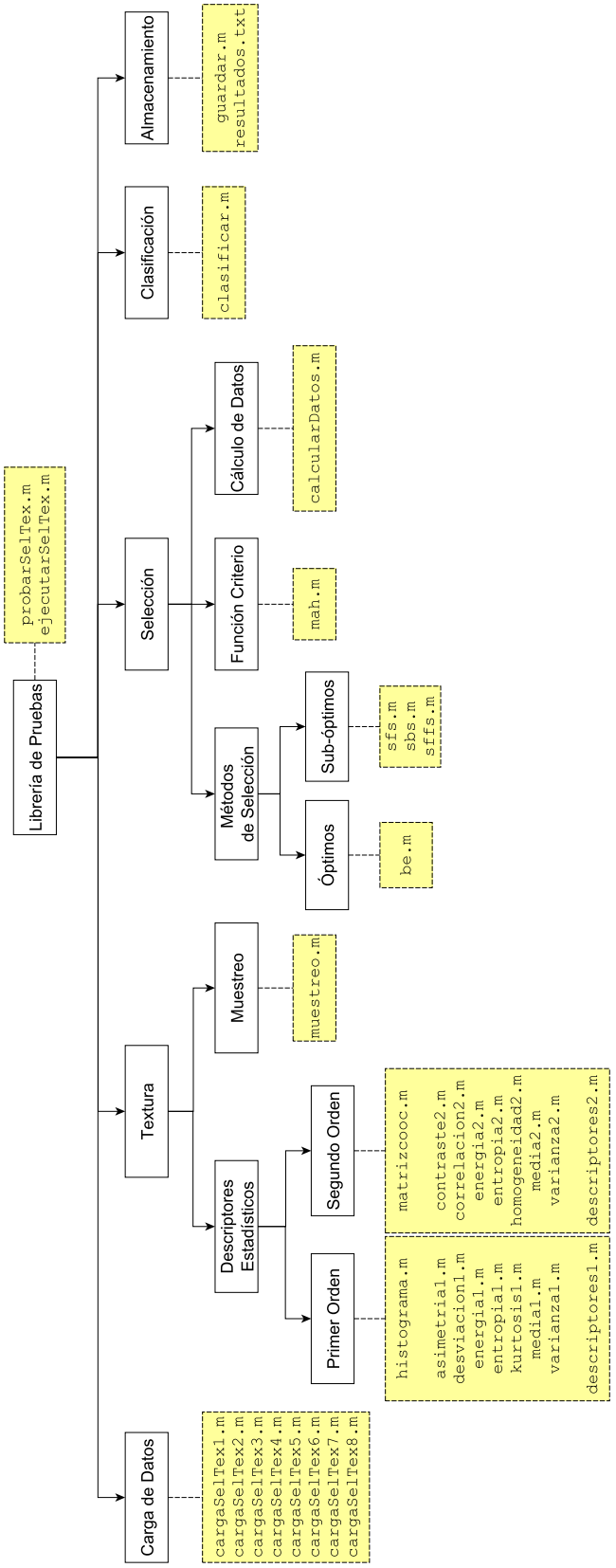


Figura 4.1: Librería para la realización de pruebas *libSelTex*.

Capítulo 5

Experimentos y Resultados

5.1. Usando *libSelección*

Los experimentos que se muestran a continuación fueron realizados utilizando la librería de funciones *libSelección* implementada, la cual fue presentada en la sección 2.4.

5.1.1. Con datos de entrada artificiales

A continuación se presentan cuatro casos de prueba. Cada uno de ellos muestra las características que se obtienen con cada método de selección al procesar los datos de entrada y aplicar las funciones de la librería *libSelección*.

En todos los casos los datos de entrada son: un vector de media para la clase 1, un vector de media para la clase 2, una matriz de covarianza asociada a las dos clases y la cantidad de características a seleccionar. Corresponden a los scripts de carga de datos 1, 2, 3 y 4 que posee la librería *libSelección*.

En la figura 5.1 se observan los valores resultantes respectivamente.

Método de Selección	Características Seleccionadas
BE	1,3
BB	1,3
SFS	4,1
SBS	1,3
SFFS	1,3

(a) Resultados con datos de prueba 1.

Método de Selección	Características Seleccionadas
BE	1,3
BB	1,3
SFS	4,1
SBS	1,3
SFFS	1,3

(b) Resultados con datos de prueba 2.

Método de Selección	Características Seleccionadas
BE	1,2,4
BB	1,2,4
SFS	1,4,2
SBS	1,4,5
SFFS	1,4,2

(c) Resultados con datos de prueba 3.

Método de Selección	Características Seleccionadas
BE	1,6
BB	1,6
SFS	5,1
SBS	2,9
SFFS	1,6

(d) Resultados con datos de prueba 4.

Figura 5.1: Resultados con datos artificiales.

En los casos presentados anteriormente puede observarse que el método sub-óptimo de selección de características *SFFS* se comporta como un método óptimo, ya que devuelve en los cuatro casos los mismos subconjuntos de características que los métodos óptimos *BE* y *BB*. No sucede lo mismo con los métodos sub-óptimos *SFS* y *SBS* que en algunos casos devuelven subconjuntos sub-óptimos.

En general *SFFS* proporciona, la mayoría de las veces, el subconjunto óptimo de características. Por el contrario, *SFS* y *SBS* suelen proporcionar subconjuntos sub-óptimos de características.

5.1.2. Con datos de la BD *Iris Plants*

A continuación se presentan tres casos de prueba. Cada uno de ellos muestra las características que se obtienen con cada método de selección al procesar los datos de entrada y aplicar las funciones de la librería *libSelección*.

Los datos que se utilizaron fueron extraídos de la Base de Datos (BD) *Iris Plants* [35]. Ésta es una base de datos que contiene información sobre lirios, consta de 150 patrones de muestra pertenecientes a tres clases distintas: virgínica, setosa y versicolor (cuenta con 50 patrones para cada una de las clases). Cada patrón de muestra está formado por 4 características: (1) largo del sépalo, (2) ancho del sépalo, (3) largo del pétalo y (4) ancho del pétalo.

A partir de los patrones de muestra de la base de datos se calcularon los datos de entrada necesarios para aplicar los métodos de selección entre dos clases. En todos los casos los datos de entrada son: un vector de media para la clase 1, un vector de media para la clase 2, una matriz de covarianza asociada a las dos clases y la cantidad de características a seleccionar. Dichos datos corresponden a los scripts de carga de datos 5, 6 y 7 que posee la librería *libSelección*.

En la figura 5.2 se observan los valores resultantes respectivamente.

Método de Selección	Características Seleccionadas
BE	1,3
BB	1,3
SFS	3,1
SBS	1,3
SFFS	3,1

(a) Resultados con datos de prueba 5.

Método de Selección	Características Seleccionadas
BE	2,3
BB	2,3
SFS	3,2
SBS	2,3
SFFS	3,2

(b) Resultados con datos de prueba 6.

Método de Selección	Características Seleccionadas
BE	2,4
BB	2,4
SFS	4,2
SBS	2,4
SFFS	4,2

(c) Resultados con datos de prueba 7.

Figura 5.2: Resultados con datos de la BD *Iris Plants*.

5.2. Usando *libTexturas*

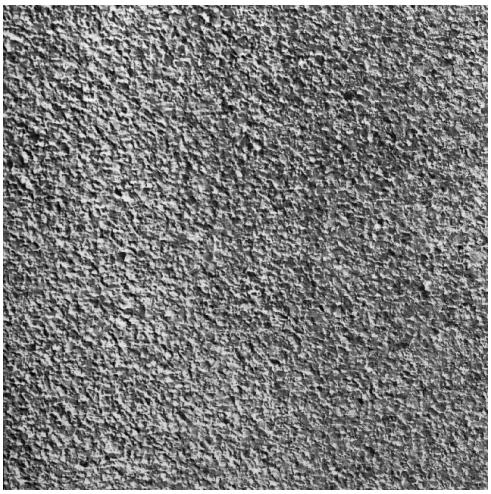
Los experimentos que se muestran a continuación fueron realizados utilizando la librería de funciones *libTexturas* implementada, la cual fue presentada en la sección 3.3.

5.2.1. Con imágenes de la BD *Brodatz*

A continuación se presentan cuatro casos de prueba. Cada uno de ellos muestra los valores de los descriptores estadísticos de primer y segundo orden que se obtienen al procesar una imagen y aplicar las funciones de la librería *libTexturas*. En la figura 5.3

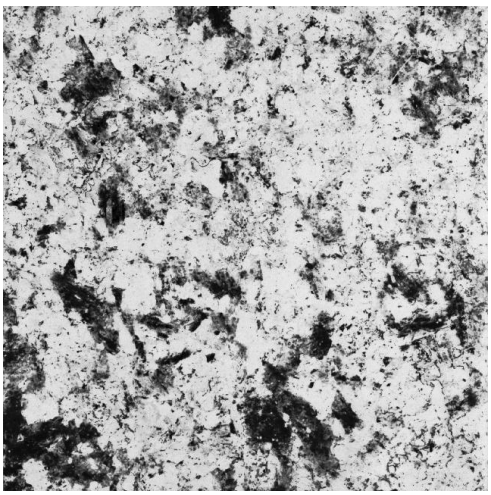
pueden verse las imágenes utilizadas. Y en las figuras 5.4, 5.5, 5.6 y 5.7 se observan los valores resultantes respectivamente.

Las imágenes que se utilizaron fueron extraídas de la Base de Datos (BD) de texturas *Brodatz* [48]. Son imágenes de 640 x 640 píxeles en escala de grises (256 niveles de gris). Corresponden a los scripts de carga de datos 1, 2, 3 y 4 que posee la librería *libTexturas*.

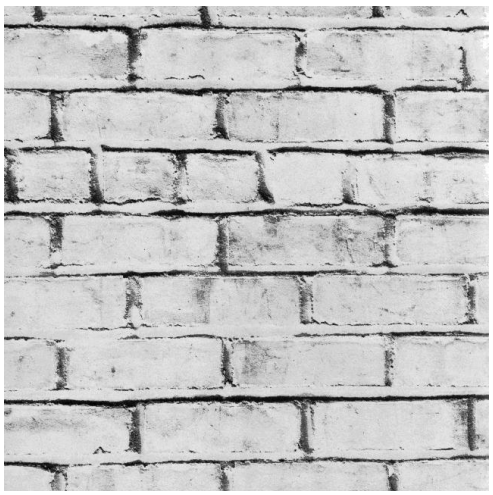


(a) Imagen de prueba 1: CORCHO.

(b) Imagen de prueba 2: MADERA.



(c) Imagen de prueba 3: MÁRMOL.



(d) Imagen de prueba 4: PARED.

Figura 5.3: Imágenes de prueba de la BD *Brodatz*.

Media	118.6511
Desviación Estándar	55.2396
Asimetría	0.2412
Kurtosis	1.9804
Energía	0.0149
Entropía	6.2898

(a) Primer orden

Energía	3.5728e-04
Contraste	2.0678e+03
Correlación	0.6612
Homogeneidad	0.0623
Entropía	12.0681

(b) Segundo orden

Figura 5.4: Descriptores estadísticos de la imagen de prueba 1: CORCHO.

Media	71.6536
Desviación Estándar	38.1767
Asimetría	1.3147
Kurtosis	5.1000
Energía	0.0226
Entropía	5.8710

(a) Primer orden

Energía	0.0010
Contraste	458.8415
Correlación	0.8423
Homogeneidad	0.1149
Entropía	10.8295

(b) Segundo orden

Figura 5.5: Descriptores estadísticos de la imagen de prueba 2: MADERA.

Media	167.6278
Desviación Estándar	65.9195
Asimetría	-0.9703
Kurtosis	2.5047
Energía	0.0416
Entropía	5.5119

(a) Primer orden

Energía	0.0065
Contraste	1.1684e+03
Correlación	0.8656
Homogeneidad	0.1992
Entropía	9.9227

(b) Segundo orden

Figura 5.6: Descriptores estadísticos de la imagen de prueba 3: MÁRMOL.

Media	193.7841
Desviación Estándar	47.8940
Asimetría	-2.3176
Kurtosis	7.4849
Energía	0.0910
Entropía	4.5076

(a) Primer orden

Energía	0.0175
Contraste	613.6756
Correlación	0.8664
Homogeneidad	0.2705
Entropía	8.1616

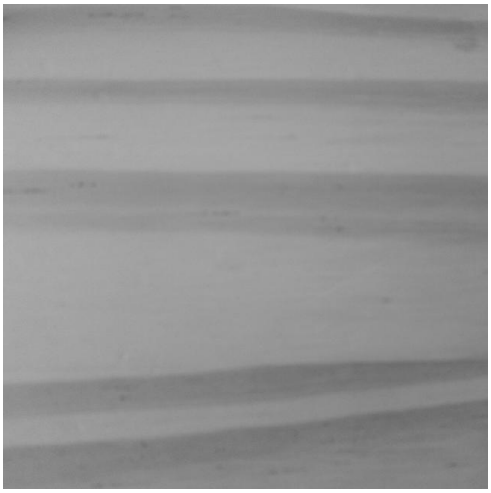
(b) Segundo orden

Figura 5.7: Descriptores estadísticos de la imagen de prueba 4: PARED.

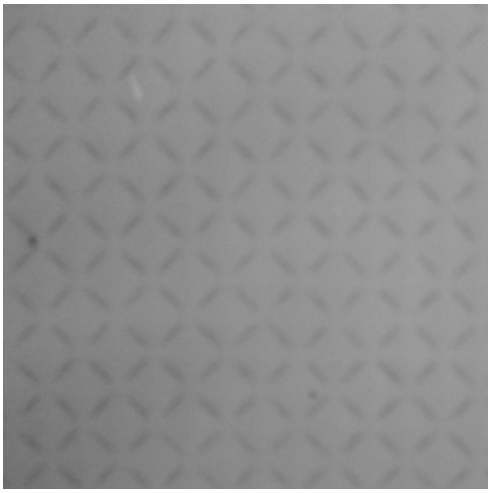
5.2.2. Con imágenes de una BD propia

A continuación se presentan cuatro casos de prueba. Cada uno de ellos muestra los valores de los descriptores estadísticos de primer y segundo orden que se obtienen al procesar una imagen y aplicar las funciones de la librería *libTexturas*. En la figura 5.8 pueden verse las imágenes utilizadas. Y en las figuras 5.9, 5.10, 5.11 y 5.12 se observan los valores resultantes respectivamente.

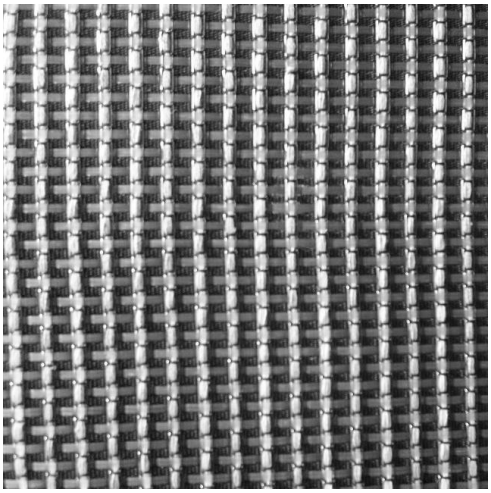
Las imágenes que se utilizaron fueron extraídas de una Base de Datos (BD) de texturas propia. Son imágenes de 640 x 640 píxeles que fueron tomadas en colores y pasadas a escala de grises (256 niveles de gris). Corresponden a los scripts de carga de datos 5, 6, 7 y 8 que posee la librería *libTexturas*.



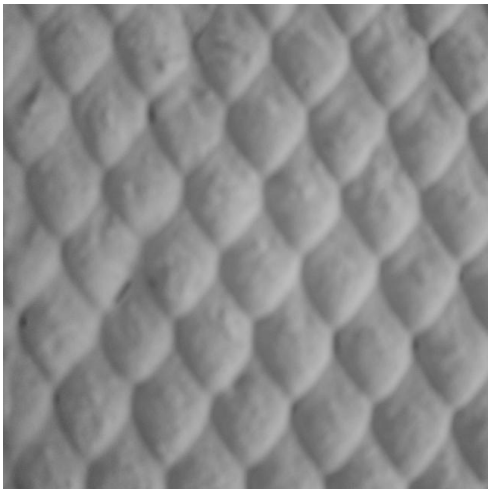
(a) Imagen de prueba 5: MADERA.



(b) Imagen de prueba 6: AZULEJO.



(c) Imagen de prueba 7: PARLANTE.



(d) Imagen de prueba 8: YESO.

Figura 5.8: Imágenes de prueba de una BD propia.

Media	146.3003
Desviación Estándar	13.7791
Asimetría	-0.3940
Kurtosis	2.3107
Energía	0.0207
Entropía	5.7109

(a) Primer orden

Energía	0.0080
Contraste	1.2315
Correlación	0.9968
Homogeneidad	0.7269
Entropía	7.6702

(b) Segundo orden

Figura 5.9: Descriptores estadísticos de la imagen de prueba 5: MADERA.

Media	141.0277
Desviación Estándar	9.6901
Asimetría	-0.5953
Kurtosis	2.9026
Energía	0.0302
Entropía	5.2106

(a) Primer orden

Energía	0.0108
Contraste	1.0857
Correlación	0.9942
Homogeneidad	0.7177
Entropía	7.1779

(b) Segundo orden

Figura 5.10: Descriptores estadísticos de la imagen de prueba 6: AZULEJO.

Media	117.1043
Desviación Estándar	65.2421
Asimetría	0.5558
Kurtosis	2.0896
Energía	0.0051
Entropía	7.7520

(a) Primer orden

Energía	1.1448e-04
Contraste	469.8973
Correlación	0.9448
Homogeneidad	0.1087
Entropía	13.7847

(b) Segundo orden

Figura 5.11: Descriptores estadísticos de la imagen de prueba 7: PARLANTE.

Media	137.8614
Desviación Estándar	19.6229
Asimetría	0.0274
Kurtosis	2.3184
Energía	0.0139
Entropía	6.3010

(a) Primer orden

Energía	0.0024
Contraste	4.1269
Correlación	0.9946
Homogeneidad	0.5078
Entropía	9.2313

(b) Segundo orden

Figura 5.12: Descriptores estadísticos de la imagen de prueba 8: YESO.

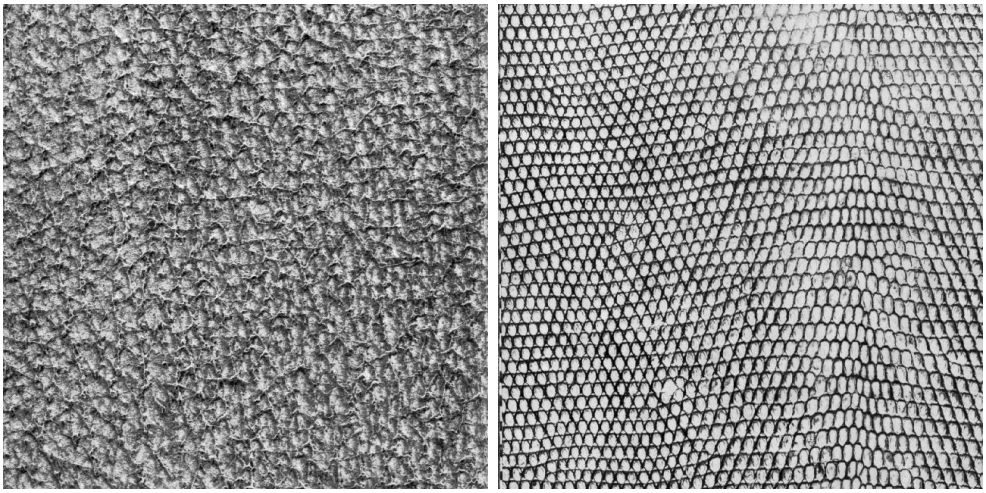
5.3. Usando *libSelTex*

Los experimentos que se muestran a continuación fueron realizados utilizando la librería de funciones *libSelTex* implementada, la cual fue presentada en la sección 4.3.

5.3.1. Con imágenes de la BD *Brodatz*

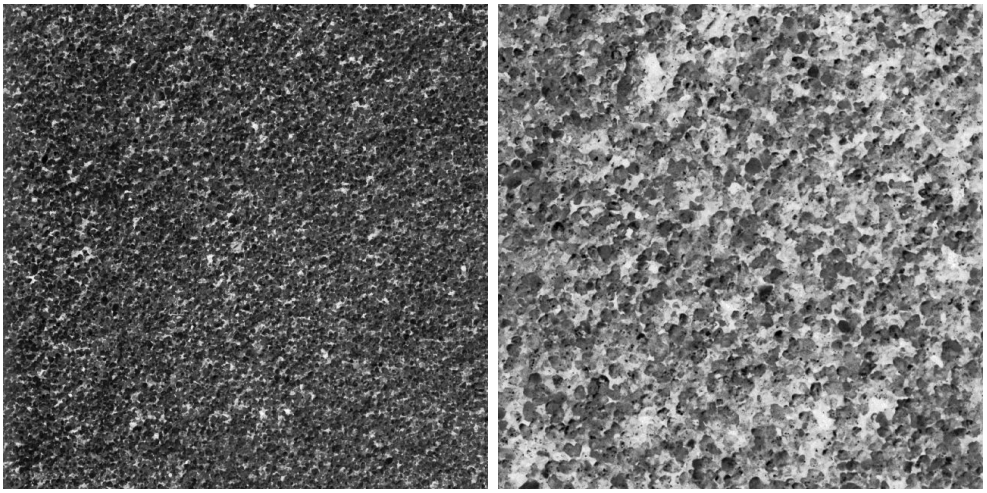
A continuación se presentan cuatro casos de prueba. Cada uno de ellos muestra los resultados que se obtienen al procesar dos imágenes y aplicar las funciones correspondientes a todas etapas del modelo de reconocimiento de patrones de *libSelTex*. En las figuras 5.13, 5.14, 5.15 y 5.16 pueden verse las imágenes utilizadas. Y en la figura 5.17 se observa el archivo de texto que contiene los valores resultantes respectivamente.

Las imágenes que se utilizaron fueron extraídas de la Base de Datos (BD) de texturas *Brodatz* [48]. Son imágenes de 640 x 640 píxeles en escala de grises (256 niveles de gris). Corresponden a los scripts de carga de datos 1, 2, 3 y 4 que posee la librería *libSelTex*.



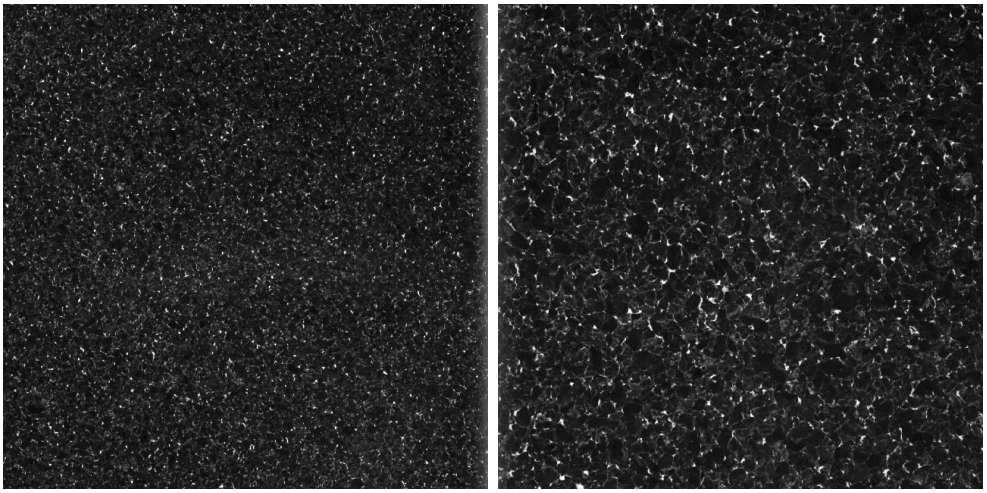
(a) Tipo 1. (b) Tipo 2.

Figura 5.13: Imágenes de prueba 1: PIEL.



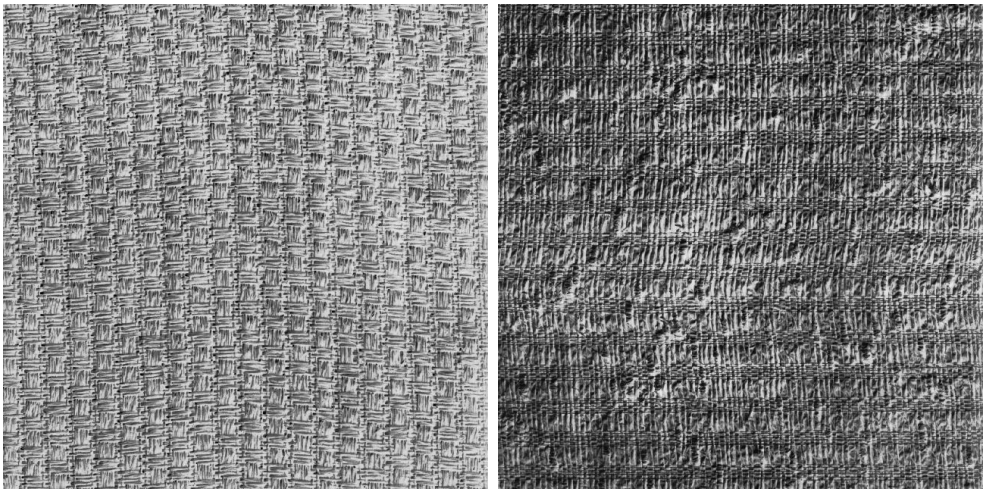
(a) Tipo 1. (b) Tipo 2.

Figura 5.14: Imágenes de prueba 2: ARENA.



(a) Tipo 1. (b) Tipo 2.

Figura 5.15: Imágenes de prueba 3: CORCHO.



(a) Tipo 1. (b) Tipo 2.

Figura 5.16: Imágenes de prueba 4: TELA.

resultados.txt

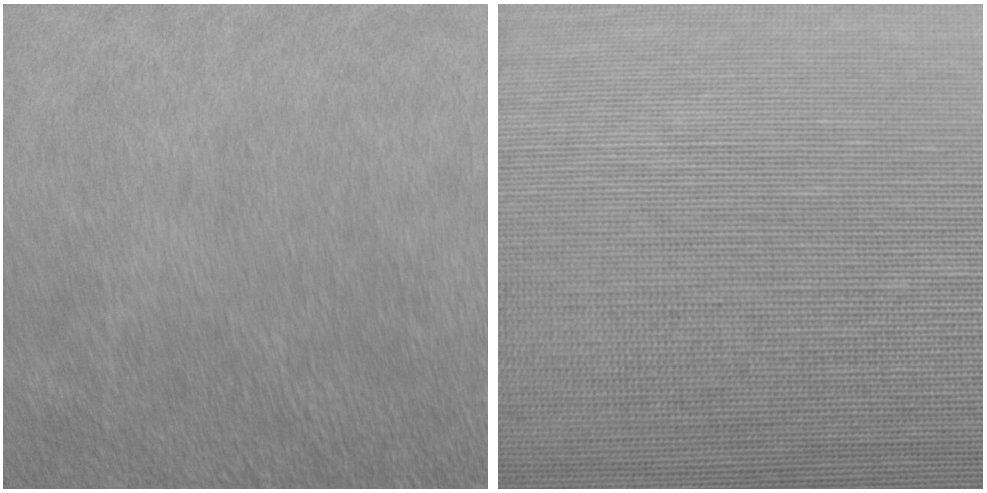
```
RESULTADOS DE LAS PRUEBAS REALIZADAS


1ra columna: IMÁGENES DE ENTRADA
2da columna: MÉTODO DE SELECCIÓN DE CARACTERÍSTICAS UTILIZADO
3ra columna: CARACTERÍSTICAS SELECCIONADAS
4ta columna: CLASIFICACIÓN OBTENIDA


1. piel_tipo1.gif   1. piel_tipo2.gif              be              2   9                   1   1   1   1   1   2   2   2   2   2
1. piel_tipo1.gif   1. piel_tipo2.gif              sfs             2   9                   1   1   1   1   1   2   2   2   2   2
1. piel_tipo1.gif   1. piel_tipo2.gif              sbs             6   11                  2   2   1   1   1   2   2   2   2   2
1. piel_tipo1.gif   1. piel_tipo2.gif              sffs                2   9                   1   1   1   1   1   2   2   2   2   2   2
2. arena_tipo1.gif  2. arena_tipo2.gif             be              3   8                   1   1   1   1   1   1   2   2   2   2
2. arena_tipo1.gif  2. arena_tipo2.gif             sfs             9   2                   1   1   1   1   1   2   2   1   2   1
2. arena_tipo1.gif  2. arena_tipo2.gif             sbs             5   8                   1   1   1   1   1   1   2   2   2   2
2. arena_tipo1.gif  2. arena_tipo2.gif             sffs                9   2                   1   1   1   1   1   2   2   1   2   1
3. corcho_tipo1.gif 3. corcho_tipo2.gif            be              5   9                   1   1   1   1   1   2   2   2   2   2
3. corcho_tipo1.gif 3. corcho_tipo2.gif            sfs             9   5                   1   1   1   1   1   2   2   2   2   2
3. corcho_tipo1.gif 3. corcho_tipo2.gif            sbs             10  11                  1   2   1   1   1   2   2   2   2   1
3. corcho_tipo1.gif 3. corcho_tipo2.gif            sffs                9   5                   1   1   1   1   1   2   2   2   2   2
4. tela_tipo1.gif   4. tela_tipo2.gif             be              4   11                  1   1   1   1   1   2   2   2   2   2
4. tela_tipo1.gif   4. tela_tipo2.gif             sfs             6   2                   1   1   1   2   1   1   2   2   2   2
4. tela_tipo1.gif   4. tela_tipo2.gif             sbs             2   6                   1   1   1   2   1   1   2   2   2   2
4. tela_tipo1.gif   4. tela_tipo2.gif             sffs                6   2                   1   1   1   2   1   1   2   2   2   2
```

Figura 5.17: Archivo de resultados obtenidos con imágenes de la BD *Brodatz*.

5.3.2. Con imágenes de una BD propia

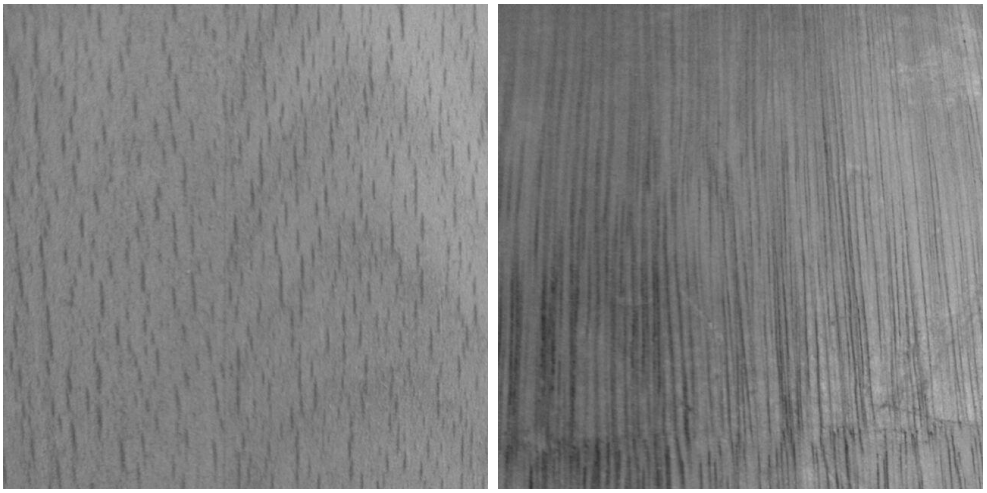
A continuación se presentan cuatro casos de prueba. Cada uno de ellos muestra los resultados que se obtienen al procesar dos imágenes y aplicar las funciones correspondientes a todas etapas del modelo de reconocimiento de patrones de *libSelTex*. En las figuras 5.18, 5.19, 5.20 y 5.21 pueden verse las imágenes utilizadas. Y en la figura 5.22 se observa el archivo de texto que contiene los valores resultantes respectivamente.

Las imágenes que se utilizaron fueron extraídas de una Base de Datos (BD) de texturas propia. Son imágenes de 640 x 640 píxeles que fueron tomadas en colores y pasadas a escala de grises (256 niveles de gris). Corresponden a los scripts de carga de datos 5, 6, 7 y 8 que posee la librería *libSelTex*.



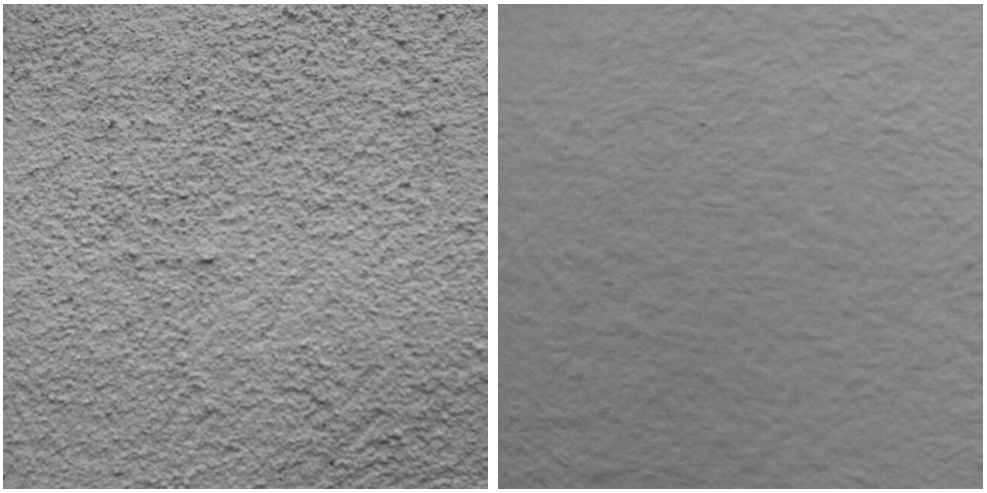
(a) Tipo 1. (b) Tipo 2.

Figura 5.18: Imágenes de prueba 5: TELA.



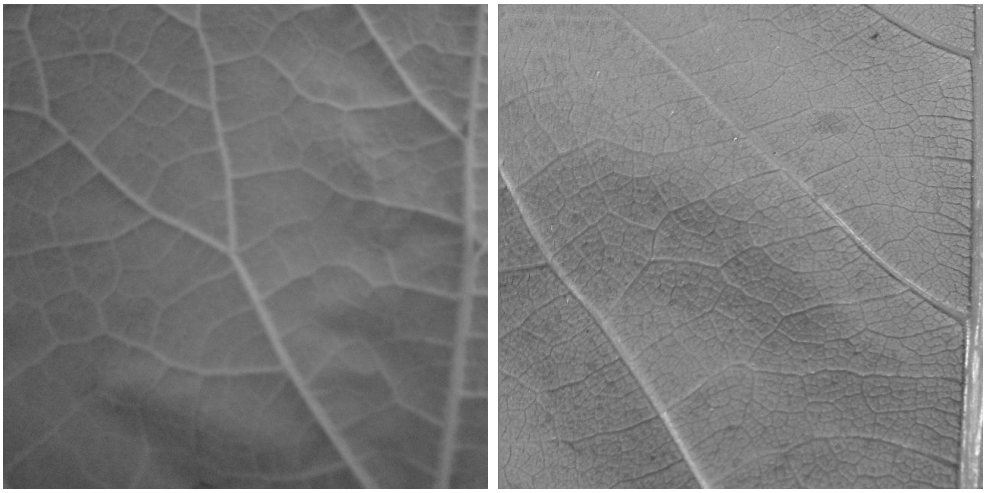
(a) Tipo 1. (b) Tipo 2.

Figura 5.19: Imágenes de prueba 6: MADERA.



(a) Tipo 1. (b) Tipo 2.

Figura 5.20: Imágenes de prueba 7: PARED.



(a) Tipo 1. (b) Tipo 2.

Figura 5.21: Imágenes de prueba 8: HOJA.

resultados.txt

```
RESULTADOS DE LAS PRUEBAS REALIZADAS


1ra columna: IMÁGENES DE ENTRADA
2da columna: MÉTODO DE SELECCIÓN DE CARACTERÍSTICAS UTILIZADO
3ra columna: CARACTERÍSTICAS SELECCIONADAS
4ta columna: CLASIFICACIÓN OBTENIDA


5. tela_tipo1.jpg   5. tela_tipo2.jpg   be    3  6          1  1  1  1  1  1  2  2  2  2
5. tela_tipo1.jpg   5. tela_tipo2.jpg   sfs   7  3          1  2  2  1  1  2  2  2  1  1
5. tela_tipo1.jpg   5. tela_tipo2.jpg   sbs   8  10         1  1  1  1  1  1  2  2  2  2
5. tela_tipo1.jpg   5. tela_tipo2.jpg   sffs     3  6          1  1  1  1  1  1  2  2  2  2
6. madera_tipo1.jpg 6. madera_tipo2.jpg be    1  10         1  1  1  1  1  2  2  1  2  2
6. madera_tipo1.jpg 6. madera_tipo2.jpg sfs   1  10         1  1  1  1  1  2  2  1  2  2
6. madera_tipo1.jpg 6. madera_tipo2.jpg sbs   1  7          1  1  1  1  1  2  2  1  2  2
6. madera_tipo1.jpg 6. madera_tipo2.jpg sffs     1  10          1  1  1  1  1  2  2  1  2  2
7. pared_tipo1.jpg  7. pared_tipo2.jpg  be    7  11         1  1  1  1  1  2  2  2  2  2
7. pared_tipo1.jpg  7. pared_tipo2.jpg  sfs   11 7          1  1  1  1  1  2  2  2  2  2
7. pared_tipo1.jpg  7. pared_tipo2.jpg  sbs   7  11         1  1  1  1  1  2  2  2  2  2
7. pared_tipo1.jpg  7. pared_tipo2.jpg  sffs     11 7          1  1  1  1  1  2  2  2  2  2
8. hoja_tipo1.jpg   8. hoja_tipo2.jpg   be    7  10         1  1  1  1  1  2  2  2  2  2
8. hoja_tipo1.jpg   8. hoja_tipo2.jpg   sfs   10 7          1  1  1  1  1  2  2  2  2  2
8. hoja_tipo1.jpg   8. hoja_tipo2.jpg   sbs   6  11         1  1  2  1  1  2  2  2  2  2
8. hoja_tipo1.jpg   8. hoja_tipo2.jpg   sffs     10 7          1  1  1  1  1  2  2  2  2  2
```

Figura 5.22: Archivo de resultados obtenidos con imágenes de una BD propia.

5.4. Comparaciones

5.4.1. Aciertos del clasificador

En base a los experimentos presentados anteriormente y los resultados expuestos en la sección 5.3, considerando un total de 40 pruebas realizadas en las que se seleccionaron 2 características, se confeccionó la tabla que se muestra en la figura 5.23. En la misma se puede ver el porcentaje de aciertos del clasificador de acuerdo al método de selección utilizado. Con los valores mostrados en la tabla puede observarse la influencia que tiene en el clasificador aplicar un método de selección que proporcione buenos resultados.

Cómo se observa el método de selección óptimo *BE* posee alto porcentaje de aciertos, esto se da debido a que dicho método siempre proporciona el subconjunto óptimo de características. También puede verse que el método de selección sub-óptimo secuencial flotante *SFFS* tiene un gran porcentaje de aciertos, cercano al del método óptimo. Este método, más allá de ser sub-óptimo, tiende a proporcionar el subconjunto óptimo de características. Por último se observa que los métodos de selección sub-óptimos secuenciales *SFS* y *SBS* poseen menores porcentajes de aciertos, ya que en algunos casos terminan proporcionando un subconjunto sub-óptimo de características.

Tipo de Método	Método de Selección	Porcentaje de aciertos del clasificador
Óptimo	BE	96.25%
Sub-óptimo	SFFS	92.50%
Sub-óptimo	SFS	88.75%
Sub-óptimo	SBS	87.50%

Figura 5.23: Porcentaje de aciertos del clasificador de acuerdo al método de selección utilizado.

5.4.2. Hallazgo del subconjunto óptimo

En la tabla de la figura 5.24 se muestra el porcentaje de veces que cada método de selección de características proporciona el subconjunto óptimo. Los valores correspondientes a los porcentajes de la tabla fueron calculados en base a los experimentos presentados en las secciones 5.1 y 5.3, con un total de 75 pruebas realizadas. Con los valores mostrados en la tabla se puede notar que los métodos de selección sub-óptimos secuenciales *SFS* y *SBS* tienen menores posibilidades de encontrar el subconjunto óptimo en comparación con el método de selección sub-óptimo flotante *SFFS*, el cual muestra un alto porcentaje de hallazgo del subconjunto óptimo.

Tipo de Método	Método de Selección	Porcentaje de veces que proporcionó el subconjunto óptimo
Óptimo	BE	100%
Sub-óptimo	SFFS	80%
Sub-óptimo	SFS	60%
Sub-óptimo	SBS	40%

Figura 5.24: Porcentaje de veces que el método de selección proporcionó el subconjunto óptimo.

5.4.3. Matrices de confusión

En las figuras 5.25 y 5.26 pueden verse las matrices de confusión correspondientes a las clasificaciones del caso de prueba 5 presentado en la sección 5.3, donde se usaron los métodos de selección *SFS* y *SFFS*. Cada matriz de confusión es una tabla de cruce que tiene dos entradas: el resultado de la clasificación y la clasificación real.

El desempeño de un clasificador está directamente relacionado con el error de clasificación del mismo. En las figuras 5.25 y 5.26 también pueden observarse los errores de comisión y omisión que se producen.

En base a los datos presentes en una matriz de confusión pueden calcularse los siguientes valores:

- *True Positive Fraction (TPF) o Sensibilidad*:

$$\frac{TP}{TP+FN}$$

- *Especificidad*:

$$\frac{TN}{TN+FP}$$

- *False Positive Fraction (FPF) o 1-Especificidad*:

$$\frac{FP}{FP+TN}$$

- *Positive Predictive Value (PPV) o Precisión*:

$$\frac{TP}{TP+FP}$$

- *Negative Predictive Value (NPV)*:

$$\frac{TN}{TN+FN}$$

- *Exactitud*:

$$\frac{TP+TN}{TP+TN+FP+FN}$$

Donde:

- *TP*: True Positive (verdadero positivo).
- *TN*: True Negative (verdadero negativo).
- *FP*: False Positive (falso positivo).
- *FN*: False Negative (falso negativo).

En las tablas de las figuras 5.27 y 5.28 se muestran los valores calculados para las matrices de confusión de las figuras 5.25 y 5.26 respectivamente. Dichos valores muestran que, en este caso, el clasificador tiene un mejor desempeño cuando utiliza las características provistas por el método de selección *SFFS* que cuando utiliza las características dadas por del método de selección *SFS*.

		Clasificación Obtenida TELA		
		Tipo 1	**Tipo 2**	**Error de Omisión**
Clasificación Real TELA	**Tipo 1**	3	2	2
	Tipo 2	2	3	2
	Error de Comisión	2	2	

Figura 5.25: Matriz de confusión usando *SFS* en la selección.

		Clasificación Obtenida TELA		
		Tipo 1	**Tipo 2**	**Error de Omisión**
Clasificación Real TELA	**Tipo 1**	5	0	0
	Tipo 2	1	4	1
	Error de Comisión	1	0	

Figura 5.26: Matriz de confusión usando *SFFS* en la selección.

TP	3
TN	3
FP	2
FN	2

TPF Sensibilidad	0.6
Especificidad	0.6
FPF 1-Especificidad	0.4
PPV Precisión	0.6
NPV	0.6
Exactitud	0.6

Figura 5.27: Valores calculados a partir de la matriz de confusión de la figura 5.25.

TP	5
TN	4
FP	1
FN	0

TPF Sensibilidad	1
Especificidad	0.8
FPF 1-Especificidad	0.2
PPV Precisión	0.83
NPV	1
Exactitud	0.9

Figura 5.28: Valores calculados a partir de la matriz de confusión de la figura 5.26.

Capítulo 6

Conclusiones y Trabajo Futuro

En esta tesina se planteó el problema de seleccionar características dentro de la etapa de reducción de dimensión de sistemas de reconocimiento de patrones. Por tal motivo, se estudió la función criterio como medida que permite comparar subconjuntos y se estudiaron métodos de selección de características óptimos y sub-óptimos.

En cuanto a los métodos de selección óptimos se analizaron los algoritmos *Búsqueda Exhaustiva (BE)* y *Branch and Bound (BB)*, este último fue presentado en su primera versión y también en la versión *Branch and Bound ordenado*. Se observó que estos algoritmos deben ser utilizados cuando se requiere la solución óptima, debido a que poseen gran costo computacional. El algoritmo *BB* puede resultar menos costoso que el algoritmo *BE* pero sólo es posible utilizarlo si la función criterio es monótona. Si se utiliza el algoritmo *BB*, entre las dos versiones mencionadas del mismo, la versión ordenada generalmente resulta ser más eficiente.

Con respecto a los métodos de selección sub-óptimos se tuvieron en cuenta para el análisis algoritmos de búsqueda secuencial y de búsqueda secuencial flotante. Para el primer caso se analizaron los algoritmos *Sequential Forward Selection (SFS)* y *Sequential Backward Selection (SBS)*, y para el segundo caso se analizó el algoritmo *Sequential Forward Floating Selection (SFFS)*. Se pudo notar que el algoritmo *SFFS* suele proporcionar buenos resultados, iguales o cercanos a los que brindan los métodos óptimos, y en cambio los algoritmos *SFS* y *SBS* terminan con mayor facilidad proporcionando soluciones

sub-óptimas.

Los métodos de selección sub-óptimos tienen menor costo computacional que los métodos óptimos, por lo tanto en muchos casos puede ser conveniente utilizarlos a expensas de no obtener siempre la mejor solución. Se establece una relación costo-beneficio que debe ser evaluada en cada caso particular al momento de decidir entre la utilización de un método óptimo o uno sub-óptimo, hay que determinar si resulta viable o no optar por un método que posea bajo costo pero que no garantice llegar a la solución óptima todas las veces.

Los métodos de selección estudiados fueron implementados en Matlab [34]. Se presentó la librería de funciones de selección *libSelección* que contiene los métodos implementados, además de varios scripts de carga de datos y un script de prueba.

En esta tesina también se abordó el estudio de texturas en imágenes digitales. Lo que llevó a estudiar diferentes tipos de descriptores de textura, focalizando en descriptores estadísticos de primer y segundo orden.

Se analizaron descriptores estadísticos de primer orden, basados en el histograma de la imagen, entre ellos: media, desviación estándar, asimetría, kurtosis, energía y entropía. También se hizo el análisis de descriptores estadísticos de segundo orden, basados en las matrices de co-ocurrencia de la imagen, entre ellos: energía, contraste, correlación, homogeneidad y entropía. Se observó que los descriptores estadísticos de primer orden no poseen información de la posición relativa que tienen los píxeles entre sí, y que esta limitación no la presentan los descriptores estadísticos de segundo orden.

Los descriptores estadísticos estudiados fueron implementados en Matlab [34]. Se presentó la librería de funciones de texturas *libTexturas* que contiene los descriptores implementados, además de varios scripts de carga de datos y un script de prueba.

Luego del estudio de métodos de selección de características y de descriptores estadísticos de textura, en esta tesina se planteó el problema de seleccionar características de texturas. Por lo tanto fue necesario armar un modelo de reconocimiento de patrones para clasificar texturas. Siguiendo el modelo propuesto se implementó la librería de funciones *libSelTex*, que luego fue utilizada para la realización de pruebas experimentales.

Por último, en esta tesina se mostraron algunos de los experimentos realizados y sus

respectivos resultados. Se presentaron casos de prueba para cada una de las librerías implementadas. Para la librería *libSelección* se utilizaron en las pruebas tanto datos artificiales como reales. Para las librerías *libTexturas* y *libSelTex* se utilizaron imágenes obtenidas de una base de datos de texturas conocida y también imágenes de una base de datos propia.

Luego de mostrar algunos casos de prueba se realizaron comparaciones de los resultados obtenidos. Los resultados de las pruebas realizadas con la librería *libSelTex* permiten ver cómo varía el rendimiento del clasificador de acuerdo al método de selección que se utilice. Se observa que el resultado de la clasificación mejora si se utilizan métodos de selección que proporcionen buenos subconjuntos de características. Los resultados de las pruebas también permiten decir que el algoritmo *SFFS* tiene buen desempeño en general, se comporta mejor que los demás métodos sub-óptimos y posee menor costo computacional que los métodos óptimos. Por dichas causas, es una opción viable cuando se decide qué método de selección de características se va a utilizar.

El trabajo realizado en esta tesina se vincula con el proyecto "*11-F017 - Cómputo Paralelo de Altas Prestaciones. Fundamentos y Evaluación de rendimiento en HPC. Aplicaciones a Sistemas Inteligentes, Simulación y Tratamiento de Imágenes*" acreditado por la *UNLP* dentro del *Programa de Incentivos* llevado a cabo por el *Instituto de Investigación en Informática - LIDI (III-LIDI).*

Como trabajo futuro se propone realizar el estudio de otros métodos de selección de características, lo cual incluye estudiar otros algoritmos de búsqueda (como podrían ser distintas versiones del algoritmo *Branch and Bound*) y otras funciones criterio. Además se propone como trabajo futuro analizar la posibilidad de hacer adaptaciones o modificaciones a los algoritmos de búsqueda estudiados, que permitan obtener mejoras en sus rendimientos.

Bibliografía

[1] F. Zhao, L. Jiao, H. Liu, X. Gao, and M. Gong. Spectral clustering with eigenvector selection based on entropy ranking. *Neurocomputing*, 2010.

[2] S. A. Toussi and H. S. Yazdi. Feature selection in spectral clustering. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2011.

[3] L. Lorenti, L. Violini, and J. Giacomantone. Selección sub-óptima del espectro asociado a la matriz de afinidad. *XIX Congreso Argentino de Ciencias de la Computación (CACIC)*, 2013.

[4] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Machine Vision*. McGraw-Hill, 1995.

[5] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[6] M. Ben Bassat. Irrelevant features in pattern recognition. *IEEE Transactions on Computers*, 1978.

[7] P. Devijer and J. Kittler. *Pattern Recognition, A Statistical Approach*. Prentice Hall, 1982.

[8] J. Kittler, A. Etemadi, and N. Choakjarernwanit. Feature selection and extraction in pattern recognition. *In Pattern Recognition and Image Processing in Physics, 37th Scottish Universities Summer School in Physics, R. A. Vaughan*, 1991.

[9] N. Choakjarernwanit. Feature selection in pattern recognition. *Technical Report VSSP-TR- 1 / 9 l, University of Surrey, UK*, 1991.

[10] J. Kittler. Feature set search algorithms. *Pattern Recognition and Signal Processing, C.H. Chen*, 1978.

[11] R. E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[12] G.V. Trunk. A problem of dimensionality: A simple example. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1979.

[13] A.K. Jain and B. Chandrasekaran. Dimensionality and sample size considerations. *Pattern Recognition in Practice, P.R. Krishnaiah and L.N. Kanal*, 1982.

[14] S.J. Raudys and A.K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *Trans. Pattern Analysis and Machine Intelligence*, 1991.

[15] A. Jain and D. Zongker. Feature selection: evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1997.

[16] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 2000.

[17] A. N. Mucciardi and E. E. Gose. A comparison of seven techniques for choosing subsets of pattern recognition properties. *IEEE Trans. Comput.*, 1971.

[18] A. K. Jain, R. P W Duin, and Jianchang Mao. Statistical pattern recognition: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2000.

[19] J. Kittler. Computational problems of feature selection pertaining to large data sets. *In E. S. Gelsema and L. N. Kanal, editors, Pattern Recognition in Practice*, 1980.

[20] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *Computers, IEEE Transactions on*, 1977.

[21] B. Yu and B. Yuan. A more efficient branch and bound algorithm for feature selection. *Pattern Recognition*, 1993.

[22] X. Chen. An improved branch and bound algorithm for feature selection. *Pattern Recognition Letters*, 2003.

[23] S. Nakariyakul and D. P. Casasent. Adaptive branch and bound algorithm for selecting optimal features. *Pattern Recognition Letters*, 2007.

[24] P. Somol, P. Pudil, F. J. Ferri, and J. Kittler. Fast branch and bound algorithm in feature selection. *In: Sanchez B., Pineda M. J., W. J., ed. Proceedings of 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2000), Orlando, Florida, USA: International Institute of Informatics and Systemics (IIIS)*, 2000.

[25] P. Somol, P. Pudil, and J. Grim. Branch and bound algorithm with partial prediction for use with recursive and non-recursive criterion forms. *In: Singh, S.; Murshed, N. A.; Kropatsch, W. G., eds. Proceedings of Second International Conference on Advances in Pattern Recognition (ICAPR 2001)*, 2001.

[26] P. Somol, P. Pudil, and J. Kittler. Fast branch and bound algorithms for optimal feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[27] E. L. Lawler and D. E. Wood. Branch and bound methods, a survey. *Oper. Res.*, 1966.

[28] P. Pudil, J. Novovicova, N. Choakjarernwanit, and J. Kittler. A comparative evaluation of floating search methods for feature selection. *Technical Report VSSP-TR-5/92, University of Surrey, UK.*, 1992.

[29] A. W. Whitney. A direct method of nonparametric measurement selection. *Computers, IEEE Transactions on*, 1971.

[30] T. Marill and D. M. Green. On the effectiveness of receptors in recognition systems. *Information Theory, IEEE Transactions on*, 1963.

[31] P. Pudil, J. Novovicova, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 1994.

[32] S. Stearns. On selecting features for pattern classifiers. *The Third International Conference of Pattern Recognition*, 1976.

[33] P. Pudil, F. J. Ferri, J. Novovicova, and J. Kittler. Floating search methods for feature selection with nonmonotonic criterion functions. In *Pattern Recognition, 1994. Vol. 2 - Conference B: Computer Vision amp; Image Processing. Proceedings of the 12th IAPR International. Conference on*, 1994.

[34] Matlab. `www.mathworks.com`, 2014.

[35] R.A. Fisher. Iris plants database, 1988.

[36] R. M. Pickett. Visual analysis of texture in the detection and recognition of objects. *Picture Processing and Psychopictorics, B. S. Lipkin and A. Rosenfeld*, 1970.

[37] H. C. Andrews, A. G. Tescher, and R. P. Kruger. Image processing by digital computer. *IEEE Spectrum*, 1972.

[38] M. Tuceryan and A. K. Jain. Texture analysis. *In: The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*, 1988.

[39] R. M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, 1973.

[40] R. C. Gonzalez and R. E. Woods. *Digital image processing.* Addison Wesley Publishing Company, 1992.

[41] R. M. Haralick, K. Shanmugam, and I. Dinstein. On some quickly computable features for texture. *In Proc. Symp. Computer Image Processing and Recognition, Univ. Missouri, Columbia*, 1972.

[42] D. C. He and L. Wang. Texture unit, texture spectrum, and texture analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 1990.

[43] E. B. Troy, E. S. Deutsch, and A. Rosenfeld. Gray-level manipulation experiments for texture analysis. *IEEE Trans. Syst., Man, Cybern*, 1973.

[44] R. M. Haralick and D. Anderson. Texture-tone study with applications to digitized imagery. *Univ. Kansas Center for Research, Inc., Lawrence, CRES Tech. Rep*, 1971.

[45] R. M. Haralick, I. Dinstein, K. Shanmugam, and D. Goel. Texture-tone study with applications to digitized imagery. *Univ. Kansas Center for Research, Inc., Lawrence, CRES Tech. Rep.*, 1972.

[46] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 1979.

[47] A. Baraldi and F. Parmiggiani. An investigation of the textural characteristics associated with gray level cooccurrence matrix statistical parameters. *Geoscience and Remote Sensing, IEEE Transactions on*, 1995.

[48] P. Brodatz. Textures: A photographic album for artists and designers, 1966.

[49] V. Batagelj, H. Block, and A. Ferligoj. *Data Science and Clasification.* Springer, 2006.

[50] Maria Petrou. *Image Processing: Dealing With Texture.* Wiley, 2006.