Algoritmos de búsqueda de vecinos más próximos en espacios métricos

Tesis doctoral de: María Luisa Micó Andrés

Dirigida por: Jose Oncina Carratalá

Enrique Vidal Ruiz

Departamento de Sistemas Informáticos y Computación.

Universidad Politécnica de Valencia, 1 de Marzo de 1996.

Índice general

Prólogo

La búsqueda del vecino más próximo es una técnica muy utilizada en reconocimiento de formas. Dado un conjunto de prototipos cuya clasificación se conoce, una muestra se clasificará en la clase donde se encuentre el prototipo cuya distancia a la muestra es mínima. En este trabajo se presenta una familia de algoritmos de búsqueda de los vecinos más próximos en espacios métricos. Estos algoritmos han sido diseñados para mejorar las prestaciones del algoritmo AESA propuesto por Vidal en 1986. En el capítulo 1 se repasa brevemente el estado de la cuestión en este tema, además de un breve recordatorio del algoritmo AESA. Este algoritmo se caracteriza porque presenta dos características que hasta ahora ningún método previo tenía:

- calcula un número medio de distancias *constante* respecto al tamaño del conjunto de prototipos para encontrar el vecino más próximo y,
- no necesita la representación de dichos prototipos en un espacio vectorial.

Este comportamiento es muy importante ya que cuando el algoritmo se aplique a un caso real, el coste asociado a la distancia utilizada es normalmente muy elevado. Sin embargo, para conseguir este resultado, el algoritmo necesita calcular y almacenar durante una fase de preproceso una matriz de distancias entre todos los prototipos del conjunto total. Este es un problema importante ya que dificulta trabajar con conjuntos grandes de prototipos. Además, el coste computacional del algoritmo, no asociado al coste de las distancias, es lineal respecto al tamaño del conjunto de prototipos.

Para resolver el primer problema, en este trabajo se plantea la utilización de un subconjunto de prototipos del conjunto total, de forma que la matriz de distancias almacena las distancias entre los prototipos de ambos conjuntos (capítulo 2). A los prototipos que pertenecen a dicho conjunto se les denominará *prototipos base*. Estos prototipos serán manipulados en el algoritmo utilizando diferentes estrategias. La utilización de un conjunto de *prototipos base* permite obtener el vecino más próximo calculando en promedio un número de distancias independiente del número de prototipos. En el capítulo 3 se propone una serie de métodos de selección del conjunto de *prototipos base*. La parte experimental que acompaña al mismo y la comparación de los costes computacionales de dichos métodos justifica la elección de unos u otros para los experimentos del resto de capítulos. En el capítulo 4 se proponen diferentes estrategias de aproximación y eliminación de los *prototipos base* en la fase de búsqueda. En el capítulo 5 se comparan dos criterios de selección de prototipos en la fase de aproximación del algoritmo. Estos criterios se

diferencian en el orden en el cual se seleccionan los prototipos (en cuanto a que si son base o no) para calcular su distancia a la muestra a clasificar.

El capítulo 6 de este trabajo está dedicado a la reducción de la sobrecarga computacional durante la fase de búsqueda. Para ello, se plantea la utilización de estructuras de datos que permitan manipular grupos de prototipos en lugar de prototipos individuales. De esta forma, se reduce el número de elementos que hay que manejar, ya que cuando se cumplen las condiciones de eliminación para un grupo, se están eliminando en un paso todos los prototipos asociados al mismo. Unidos a este objetivo están, por supuesto, el mantenimiento del comportamiento del número de distancias a calcular (independiente del número de prototipos) y de la complejidad espacial lineal (para los cual se utilizan los *prototipos base* de una forma similar a la de los algoritmos previos).

Para comparar en las mismas condiciones de trabajo el comportamiento de los métodos propuestos con otros aparecidos anteriormente, en el capítulo 7 se repiten algunos experimentos realizados para otras técnicas que sólamente usan las propiedades métricas del espacio. En este caso las técnicas se han elegido en función del tipo de almacenamiento utilizado para los prototipos: listas [Vid, 86] o árboles [FN, 75].

Por último, en el capítulo 8, los métodos propuestos son aplicados a un caso real; concretamente, al reconocimiento de caracteres manuscritos. Para la extracción de características de los caracteres manuscritos se utiliza un algoritmo muy simple que genera cadenas. Una vez obtenidas las cadenas, se necesita una distancia para aplicar los algoritmos a la tarea de reconocimiento. La distancia utilizada ha sido la *distancia de edición* o de Levenshtein [WF, 74]. El objetivo principal en este capítulo es comprobar que el comportamiento del algoritmo con datos reales es similar al observado con datos artificiales.

Parte I

Introducción

Capítulo 1

Introducción

1.1. Reconocimiento geométrico de formas

En los últimos años el reconocimiento de formas ha adquirido cierta popularidad gracias a la automatización de las soluciones a muchos problemas de la vida real. El hecho de que existan muchas disciplinas que utilizan el reconocimiento de formas, hace que no sea nada fácil encontrar un paradigma aplicable a todas ellas. El más utilizado, por su generalidad, es el paradigma de la clasificación. Según este, un sistema de reconocimiento de formas consta de tres etapas (ver figura 1.1) [DH, 73]: en la primera, se obtiene una representación del objeto como resultado de un conjunto de mediciones; en la segunda, denominada extracción de características, se realiza un proceso interpretativo cuyo resultado se considera como una nueva representación del objeto en la que se extrae información relevante sobre el mismo; la tercera etapa es la clasificación propiamente dicha o proceso de identificación [CV, 87].

En cualquier problema de reconocimiento es además importante una fase de adquisición de conocimiento, denominada aprendizaje o entrenamiento. En esta fase se parte de muestras controladas (de clasificación conocida) para establecer los modelos requeridos para el diseño del clasificador y/o los parámetros de estos modelos.

Cuando el espacio de representación en el que se manifiestan los objetos tiene una estructura esencialmente vectorial se obtiene la llamada *aproximación geométrica* al reconocimiento de formas. En esta aproximación, la clasificación se puede formular a partir de funciones discriminantes, también llamadas funciones de decisión. Estas funciones se definen de forma que permiten dividir en clases el espacio de representación al cual pertenecen las

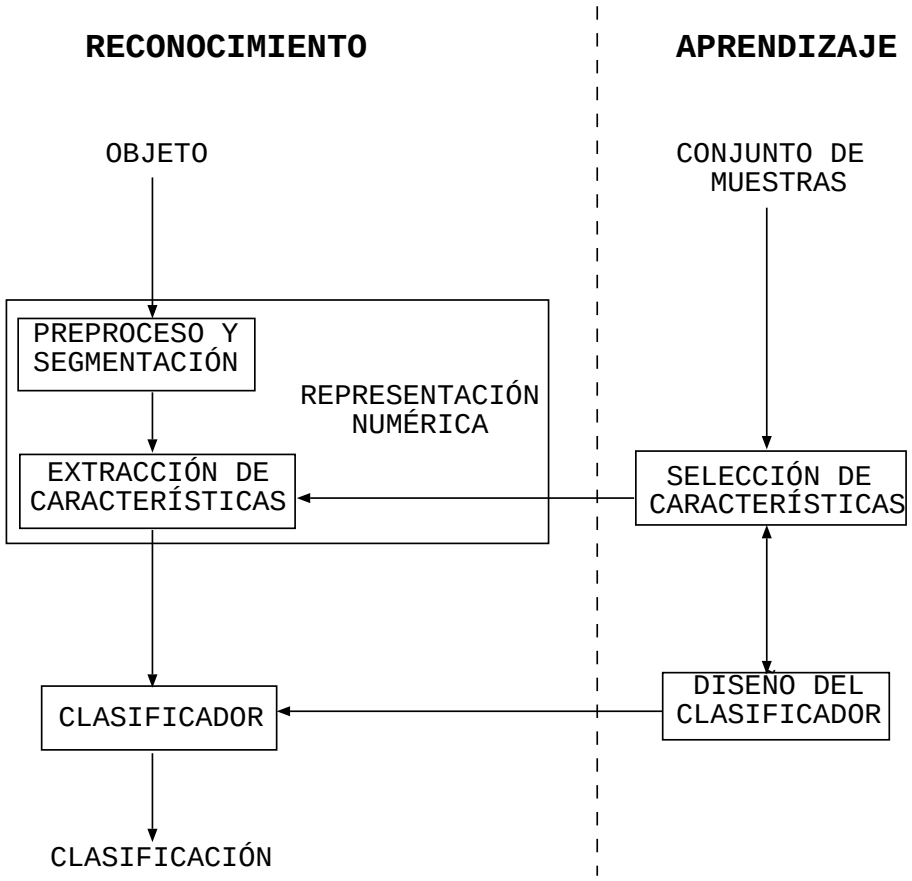

Figura 1.1: Elementos de un sistema de reconocimiento geométrico de formas.

diferentes formas [1]. Las fronteras de decisión (asociadas a cada clase) son los límites de separación entre las diferentes regiones obtenidas. A la hora de tomar una decisión para clasificar cada objeto en las C clases $(c_1, c_2 \dots c_C)$ en que se divide el espacio de representación, se intenta que el error cometido al realizar esta clasificación sea mínimo. Para ello será útil usar la máxima información posible asociada a los objetos, extraída a partir de observaciones de los mismos. Concretamente, si se conoce la probabilidad *a posteriori* de

[1] Sea E el espacio de representación y $c_1 \dots c_m$ el conjunto de clases correspondientes a regiones disjuntas de E, $\forall x \in E$ se definen las *funciones discriminantes* $D_i(x) : E \to \mathbb{R}$ asociadas a cada clase $c_i, i = 1 \dots C$, como aplicaciones del espacio E en los reales tal que, si x pertenece a la clase c_j el valor de $D_j(x)$ debe ser el mayor; esto es:

$$x \in c_j \Longleftrightarrow D_j(x) \geq D_i(x) \;\; i, j = 1 \dots C$$

elegiéndose arbitrariamente en caso de igualdad.

que un objeto x pertenece a una clase, se decide escoger aquella que presente el mayor valor:

$$p(c_i|x) > p(c_j|x) \;\; 1 \leq j \leq C, \; i \neq j \implies x \in c_i \tag{1.1}$$

Este criterio constituye la *regla de decisión de Bayes* de error mínimo, en el cual se basan la mayoría de los métodos de clasificación estadísticos. Sin embargo, la probabilidad *a posteriori* de que un objeto pertenezca a una clase determinada no es un dato del que se suela disponer en la mayoría de los casos. Sí que se pueden estimar a partir de conjuntos de entrenamiento las funciones de densidad de probabilidad de cada clase $p(x|c_i)$ en el espacio de representación (vectorial) de los objetos. También las probabilidades *a priori* pueden estimarse fácilmente según la frecuencia observada de las distintas clases. Por lo tanto, aplicando la fórmula de Bayes

$$p(c_j|x) = \frac{p(x|c_j)p(c_j)}{p(x)} \tag{1.2}$$

donde $p(x) = \sum_{j=1}^{C} p(x|c_j)p(c_j)$ y $p(c_j)$ es la probabilidad a priori de cada clase, la *regla de decisión de Bayes* queda como:

$$p(x|c_i)p(c_i) > p(x|c_j)p(c_j), \; i \neq j \implies x \in c_i. \tag{1.3}$$

Según el uso que se haga del conocimiento de la naturaleza de las funciones a estimar, se puede distinguir entre métodos de clasificación *paramétricos* y métodos *no paramétricos* [DH, 73], [Fuk, 90]. Los métodos paramétricos son aquellos en los que se supone conocida la forma funcional de la distribución estadística (densidad de probabilidad) de los objetos en cada clase. Los parámetros de esta función han de ser estimados. Por otro lado, en los métodos no paramétricos se obtienen las funciones discriminantes y las fronteras de decisión asociadas sin recurrir a asunciones previas sobre las distribuciones estadísticas subyacentes. Para este último grupo se ha propuesto una gran variedad de funciones discriminantes que dan lugar a diferentes tipos de clasificadores. Dentro de este grupo de clasificadores se encuentran aquellos que utilizan una medida de la *distancia* entre los objetos a clasificar y un conjunto de prototipos u objetos de referencia. Los clasificadores que se van a mencionar aquí requieren solamente que el espacio de representación tenga una estructura (pseudo)métrica.[2] Es razonable asumir en estos casos que

[2]Se llama distancia definida en un conjunto no vacío A a una aplicación $d : A \times A \to \mathbb{R}^+ \cup \{0\}$ tal que:

1. $d(p, q) = 0$ si, y solo si, $p = q$, donde p y q son elementos de A.
2. $d(p, q) = d(q, p)$, $\forall \; p, q \in A$.

aquellos objetos que se encuentran muy cercanos (en una métrica apropiada) tendrán la misma clasificación.

1.1.1. Clasificadores por distancia mínima

La regla de clasificación por distancia mínima es la más simple de las que se aplican en reconocimiento de formas. Sea E un conjunto dotado de una (pseudo)métrica $d : E \times E \to \mathbb{R}$. Supongamos que se dan m prototipos $p_1, p_2, \ldots p_m \in E$, tales que $p_i \in c_i$ representa a la clase c_i $(i = 1 \ldots m)$, es decir, un prototipo por clase. Una clasificación por distancia mínima respecto a $p_1, p_2, \ldots, p_m$ puede definirse para cualquier prototipo $x \in E$ como:

$$x \in c_i \iff d(x, p_i) \leq d(x, p_j);\ i, j = 1 \ldots m,\ i \neq j \tag{1.4}$$

donde d es la (pseudo)métrica definida en E.

En este tipo de clasificador, la fase de aprendizaje consiste únicamente en la elección de un buen representante de cada clase en el conjunto de prototipos que pertenecen a la misma. Normalmente el representante elegido suele ser aquel que se encuentra más centrado dentro de la distribución de los prototipos en la clase. En este clasificador la muestra x se clasifica en la clase cuyo representante se encuentra a menor distancia. En la figura 1.2 se puede ver gráficamente el funcionamiento de este clasificador para un ejemplo de las clases c_i, c_j y c_k, representadas por los prototipos p_i, p_j y p_k en espacios de representación vectorial con un producto escalar. En el ejemplo, se puede ver que x se clasificará en la clase c_k. Las fronteras de decisión en este tipo de espacios que separan las clases son los hiperplanos mediatrices de los segmentos formados por todos los pares de prototipos. Estas fronteras de decisión son un caso especial de clasificador lineal. El conjunto de regiones definidas por las fronteras de decisión asociadas a un clasificador de distancia

3. $d(p, q) + d(q, s) \geq d(p, s)$, $\forall p, q, s \in A$.

Un par (A, d) formado por un conjunto A y una distancia d definida en A se le llama espacio métrico. A la distancia se le llama también métrica. Si no se cumple la desigualdad triangular (3), a la distancia se le llama (pseudo)métrica.

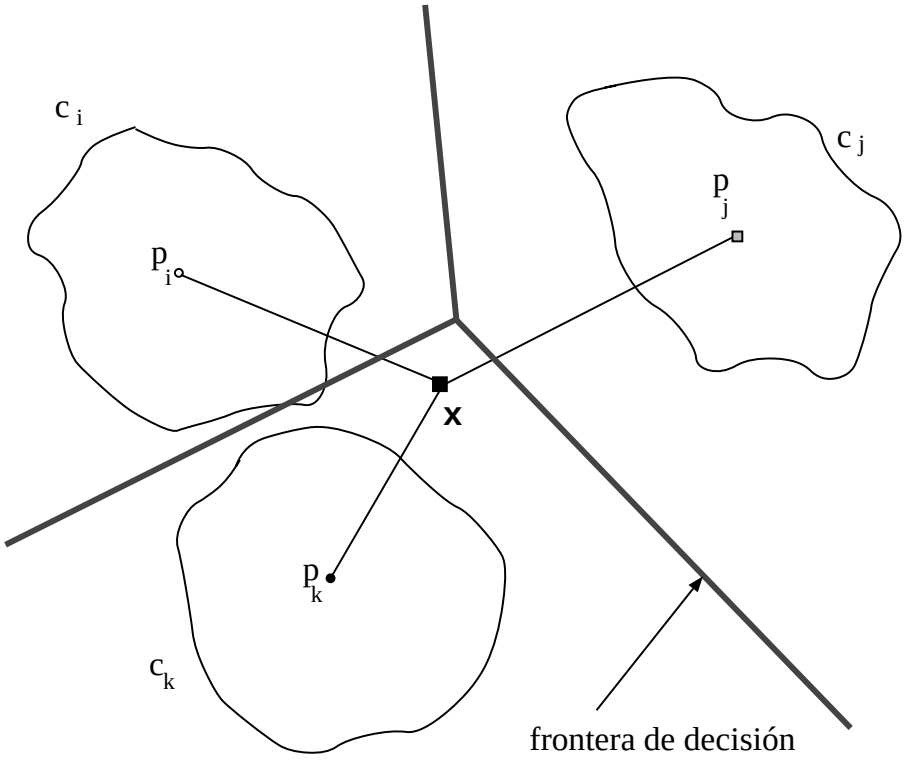

Figura 1.2: *Clasificador por distancia mínima.* La muestra se clasifica en la clase para la cual la distancia a su representante es mínima, en este caso, se clasifica en la clase c_k.

mínima recibe el nombre de *diagramas de Voronoi*[3]

1.1.2. Regla del vecino más próximo

Este clasificador es una generalización del clasificador por distancia mínima, ya que en este caso se permite que haya más de un prototipo por clase.

[3]Dado un conjunto de puntos $\{p_1 \dots p_N\} \in \mathbb{R}^k$, una celda de Voronoi c_i (asociada al punto p_i) se define como:

$$c_i = \{x \in \mathbb{R}^k : d(x, p_i) < d(x, p_j),\ p = 1 \dots N,\ j \neq i\}$$

Todas las regiones de Voronoi de los puntos p_i, $i = 1 \dots N$ forman una partición del espacio $\mathbb{R}^k$ de la forma $C = \{c_i, i = 1 \dots N\}$, y se cumple que

$$\cup_{i=1}^{N} c_i = \mathbb{R}^k, c_i \cap_{i \neq j} c_j = \emptyset$$

En este caso la distancia entre muestras y clases se define como sigue:

$$d_{\mathrm{VMP}}(x, c_i) = \min_{\forall p \in c_i} d(x, p);\ x \in E, c_i \in C \tag{1.5}$$

La muestra x se clasificará en la clase que contiene el prototipo más cercano.

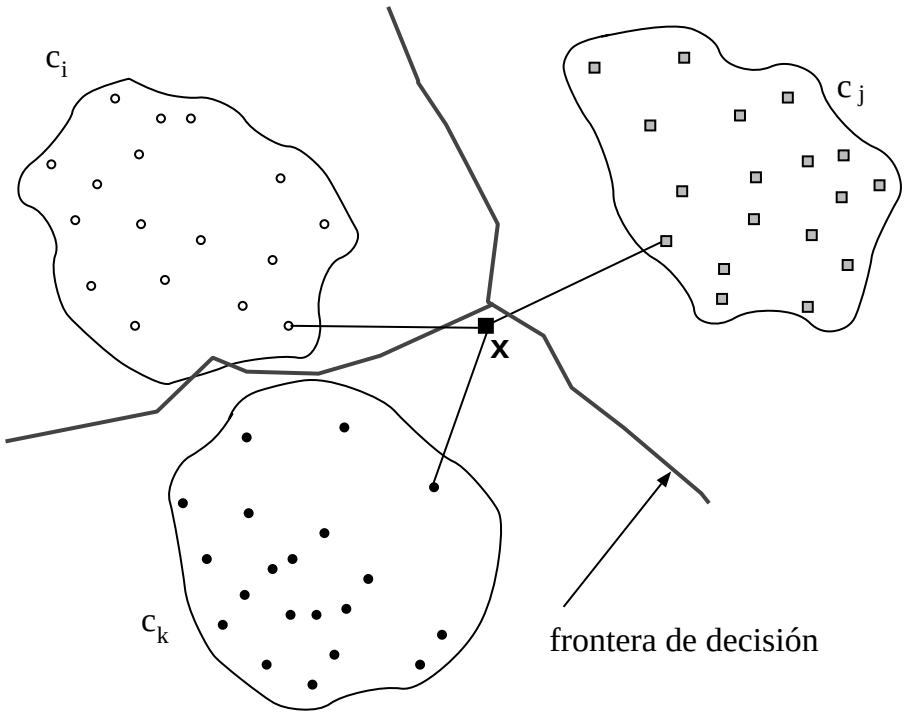

Figura 1.3: Clasificador por la *regla del vecino más próximo*. La muestra se clasifica en la clase para la que la distancia a uno de sus puntos es mínima. En este ejemplo el prototipo más cercano pertenece a la clase c_k.

En espacios de representación vectorial con producto escalar las funciones discriminantes asociadas a un clasificador basado en la regla del vecino más próximo son lineales a intervalos [TG, 74] (ver figura 1.3). Como consecuencia, se pueden obtener funciones discriminantes arbitrariamente complejas, lo que supone una ventaja clara sobre otros clasificadores (paramétricos o no paramétricos) que sólo producen ciertos tipos de fronteras (lineales, cuadráticas, ...). Cover y Hart demuestran en [CH, 67] que la probabilidad de error de este clasificador está acotado por el doble de la probabilidad de mínimo error de Bayes:

$$p_{\mathrm{B}} \leq p_{1-\mathrm{NN}} < p_{\mathrm{B}} \left(2 - \frac{C p_{\mathrm{B}}}{C-1} \right) < 2 p_{\mathrm{B}} \tag{1.6}$$

donde p_{B} es la probabilidad de mínimo error de Bayes y $p_{1-\mathrm{NN}}$ es la probabilidad de error del clasificador basado en la regla del vecino más próximo.

La fase de aprendizaje para un clasificador por el vecino más próximo consiste simplemente en la obtención del conjunto de objetos considerados como prototipos de cada clase, el cual constituye todo el conocimiento a priori del sistema. Este diccionario puede obtenerse de forma directa o se puede refinar usando técnicas de edición y condensado [DK, 82][Har, 68]. Con estas técnicas se intenta reducir el número de prototipos manteniendo o incluso mejorando las prestaciones del clasificador original.

1.1.3. Regla de los k vecinos más próximos

Es una generalización de la regla del vecino más próximo [FN, 75] [KPK, 85] [NG, 88] [AJV, 93]. En este caso la estructura del conjunto de prototipos viene dada de forma que cada clase c_i contiene al menos k prototipos ($|c_i| \geq k \geq 1$). Una muestra x se clasificará en aquella clase que contiene a la mayoría de los k prototipos más cercanos a la muestra (ver figura 1.4).

En este trabajo se presentará una serie de algoritmos de búsqueda del vecino más próximo. Debido a ello no se hará más hincapié en el resto de clasificadores por distancia comentados anteriormente.

1.2. Técnicas de búsqueda de vecinos

1.2.1. Introducción

La búsqueda del vecino más próximo es una técnica utilizada principalmente en reconocimiento de formas. Sin embargo, esto no quiere decir que no pueda ser de utilidad en otras disciplinas. Por ejemplo, se puede aplicar para realizar consultas a una base de datos. Estas consultas pueden ser de datos concretos, o incluso de datos aproximados (como por ejemplo la búsqueda de los apellidos parecidos a González (Gonsález, Gosálvez...)). Esta técnica también puede aplicarse a la detección y corrección de errores tipográficos en los procesadores de texto. El método consiste en la búsqueda en un diccionario de la palabra que más "se parece", o las k que más se parecen, a aquella que queremos corregir.

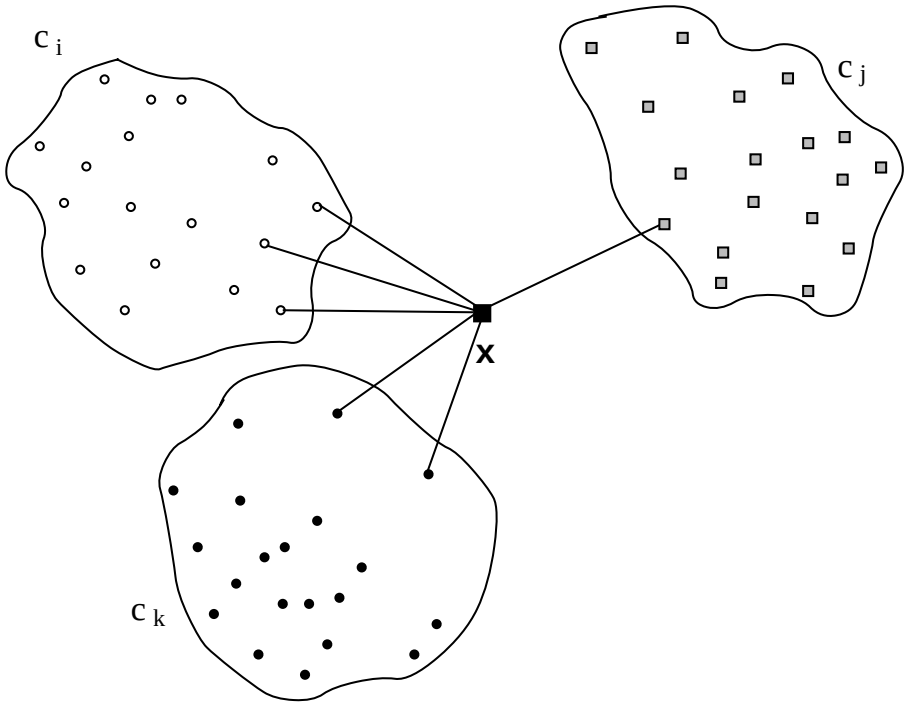

Figura 1.4: Clasificador basado en la *regla de los k vecinos más próximos*. La muestra se clasifica en la clase a la que pertenece la mayoría de los k vecinos más próximos (en el ejemplo k=6). En este ejemplo, la clase mayoritaria es c_i (3 vecinos).

1.2.2. Búsqueda exhaustiva

El método trivial para resolver el problema de la búsqueda del vecino más próximo consiste en recorrer la lista de prototipos. Para cada prototipo se calcula la distancia a la muestra y se guarda aquel que es más cercano hasta el momento (ver figura 1.5).

Sin embargo, el coste en número de distancias calculadas es proporcional al tamaño del conjunto de prototipos. En muchos problemas en los que se aplica la técnica de búsqueda de vecinos, este método es prohibitivo debido a que el coste inherente al cálculo de la distancia d puede ser elevado. Como consecuencia, han ido surgiendo en los últimos años una serie de técnicas más sofisticadas que realizan un *preproceso* para reducir el número de distancias a calcular en la fase de búsqueda a cotas sublineales [Das, 91]. Se trata, pues, de buscar un compromiso entre el coste (temporal y espacial) asociado al preproceso y el coste de la fase de búsqueda.

Algoritmo BÚSQUEDA EXHAUSTIVA
Entrada:
$P \subset E$ *// subconjunto finito de prototipos*
$x \in E$ *// muestra*
Salida:
$min \in P$ *// prototipo de P más cercano a la muestra*
$D_{\min} \in \mathbb{R}$ *// distancia del prototipo más cercano a la muestra*
Función: $d : E \times E \to \mathbb{R}$ *// función distancia*
Variables:
$p, s \in P$
$dxs \in \mathbb{R}$
Método:
$D_{\min} = \infty; min = \text{elemento\_aleatorio}(P);$
para todo $p \in P$ **hacer**
$dxs = d(x, s);$ *// cálculo de la distancia*
si $dxs < D_{\min}$ **entonces** *// actualiza mejor solución*
$D_{\min} = dxs;\ min = s;$
fin si
fin para todo
fin Método
fin Algoritmo

Figura 1.5: Algoritmo de búsqueda exhaustiva.

Entre todos los métodos propuestos existen algunos que necesitan una adecuada representación de los prototipos en un espacio vectorial, una medida de distancia entre los prototipos, o ambas [FBS, 75] [FBF, 77] [RP, 90] [Vid, 86] [FN, 75] [KPK, 85]. Sin embargo, existen otros que son especialmente interesantes porque sólo hacen uso de la segunda característica mencionada; esto hace que sean aplicables a una variedad mayor de problemas prácticos, como por ejemplo el reconocimiento de palabras aisladas, problema para el cual no se puede encontrar una adecuada representación de los elementos a reconocer en un espacio vectorial [VRCB, 88] [VL, 88] [AJV, 93].

1.2.3. Búsqueda de vecinos en espacios vectoriales

La mayoría de los métodos propuestos en la literatura pertenecen a este grupo. Esto se debe a que para muchos problemas de reconocimiento de formas puede obtenerse una adecuada representación vectorial de los datos u objetos a reconocer. Algunos ejemplos de ello son la cuantificación vectorial

aplicada a problemas como la codificación del habla [RP, 88][RP, 89].

Estos métodos basan su funcionamiento en estructuras de datos que utilizan una representación vectorial de los puntos. El árbol de búsqueda *d*- dimensional, más conocido como *k-dtree*, es la estructura de datos más usada en este tipo de algoritmos. Esta estructura, desarrollada por J.L. Bentley en 1975 fue originalmente diseñada para realizar búsquedas asociativas rápidas [Ben, 75].

Un k-dtree es un árbol binario, generalización del árbol unidimensional, también conocido como árbol binario de búsqueda. La idea que conlleva la búsqueda en una estructura de este tipo es dividir el dominio de búsqueda (dando un valor de partición) donde los puntos se encuentran ordenados en dos subdominios. La comparación con el valor de partición nos dirá en cuál de los dos subdominios se encuentra cada punto (los menores a la izquierda y los mayores a la derecha).

El k-dtree se basa en la misma idea original de dividir el espacio de representación $\mathbb{R}^k$ en dos espacios disjuntos a partir de un hiperplano perpendicular al vector correspondiente a una de las coordenadas k. Estos hiperplanos H, dados como $H = \{x \in \mathbb{R}^k : x_j = h\}$, define dos mitades, R_L y R_R como $R_L = \{x \in \mathbb{R}^k : x_j < h\}$ y $R_R = \{x \in \mathbb{R}^k : x_j > h\}$. Cada hiperplano viene representado por dos valores: 1) el índice del eje de coordenada ortogonal, j, también llamado discriminador y 2) la localización del hiperplano, h, en dicho eje. Cualquier prototipo x puede ser localizado con respecto al plano de partición realizando una comparación del tipo $x_j \leq h$ (ver si la coordenada j del punto x se encuentra a la derecha o la izquierda de la partición). Cada uno de los espacios obtenidos puede ser sucesivamente dividido por hiperplanos perpendiculares a los ejes de coordenadas. De esta forma, empezando en la raíz para el eje de coordenadas 1, los hiperplanos de partición para los siguientes niveles se obtienen cíclicamente desde este valor hasta k (ver figura 1.6).

Posteriormente, y gracias a las buenas prestaciones de esta estructura para realizar operaciones de búsqueda e inserción de puntos, Friedman et al. la utilizaron para realizar la búsqueda de los vecinos más próximos. Además, modificaron la estructura original para conseguir que la búsqueda fuese más rápida [FBF, 77]. Las modificaciones introducidas fueron desarrolladas bajo la restricción de no disponer de información a priori de la distribución del conjunto de datos. Estas modificaciones consistían en los siguientes puntos: 1) la coordenada discriminante (respecto a la cual se realiza la división) en cada nodo es aquella para la que la varianza del conjunto de datos es máxima y 2) el hiperplano separador en el eje correspondiente está situado en la mediana, de esta forma el número de puntos en ambas partes es el mismo, consiguiendo de esta manera un árbol equilibrado. El proceso de búsqueda

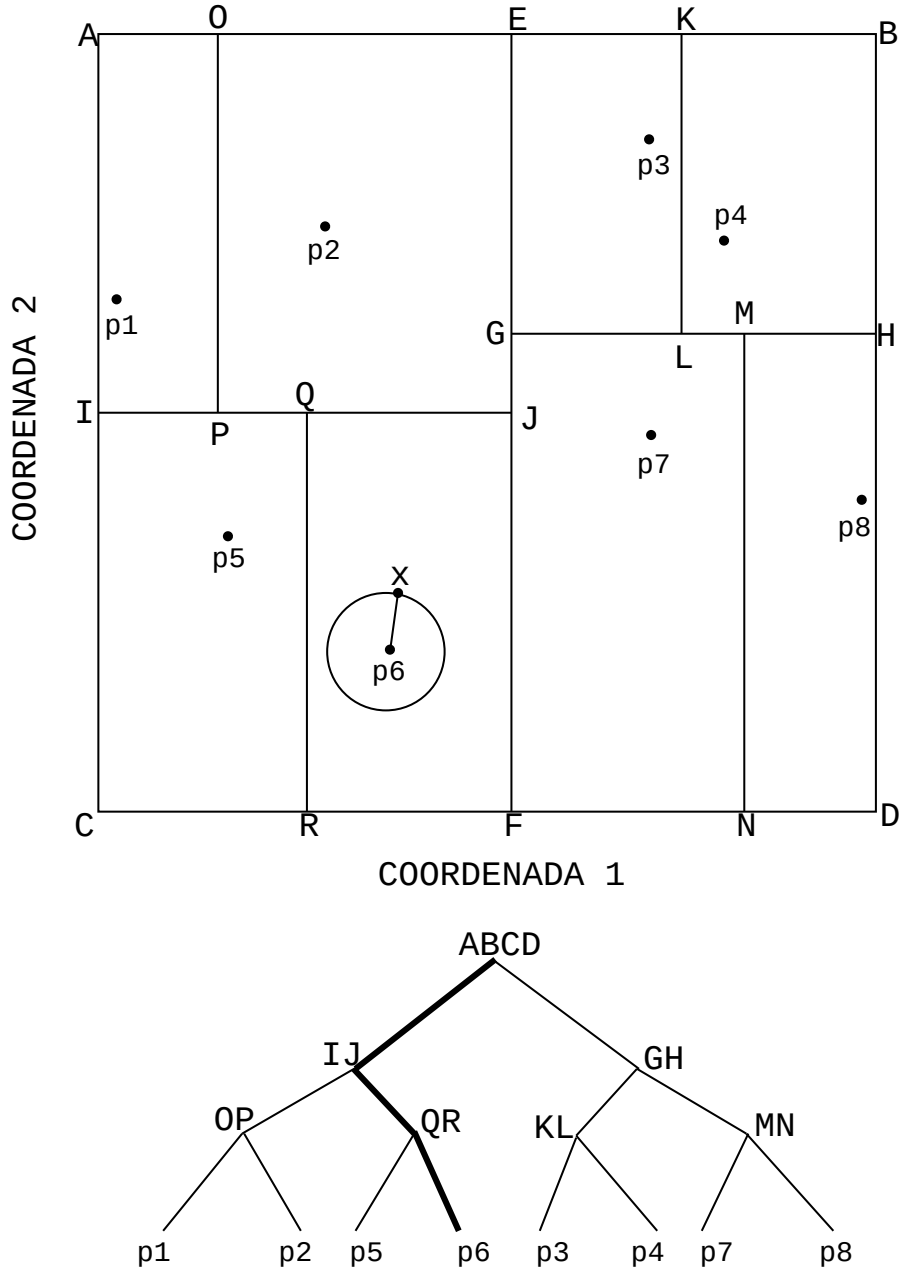

Figura 1.6: Espacios de búsqueda (arriba) y k-dtree (con k=2) y proceso de búsqueda del vecino más próximo a x (abajo).

sobre esta estructura se lleva a cabo descendiendo por las ramas del árbol, eligiendo en cada nivel aquella subregión que contiene el punto. Cuando se llega a una hoja, se examinan exhaustivamente todos los puntos que pertenecen a la misma y si la hiperesfera con centro en la muestra y radio dado por la distancia al vecino más cercano hasta el momento no corta ninguna otra región, el proceso de búsqueda ha terminado. De no ser así, el proceso debe repetirse para estas regiones. Los autores demuestran, para espacios vectoriales, que el coste temporal del proceso de búsqueda es en promedio logarítmico con el número de puntos y tiene una dependencia exponencial de la dimensión.

Debido a sus buenas prestaciones, el k-dtree ha sido utilizado en muchas aplicaciones de cuantificación vectorial como la codificación del habla [RP, 88], [RP, 89]. También ha sido estudiado para observar su comportamiento en espacios discretos [MS, 86].

Otro tipo de representación utilizada frecuentemente ha sido la proyección de los puntos en los ejes de coordenadas [FBS, 75] [Yun, 76] [CGRS, 84] [RP, 92]. Uno de los primeros métodos en el que se utilizaron las proyecciones fue el de Friedman et al. [FBS, 75]. El algoritmo que proponen utiliza la proyección de todos los prototipos en un eje para determinar de esta forma un orden relativo respecto a la muestra. Yunck propone un método que se basa en la utilización de un hipercubo (obtenido a partir de las proyecciones de los prototipos en cada uno de los ejes de coordenadas). El proceso consiste en la obtención de un hipercubo inicial que se va extendiendo o comprimiendo, siguiendo un proceso iterativo hasta que contiene exactamente k prototipos (los k vecinos más próximos a una muestra dada) [Yun, 76].

En [CGRS, 84] se propone un método de proyección similar al propuesto por Yunck (utilización de hipercubos alrededor de la muestra). También en [CGRS, 84] se estudian otros métodos que utilizan otro tipo de proyección, en este caso la proyección de las *celdas de Voronoi* sobre los ejes de coordenadas. Los límites de las celdas dan lugar a una serie de intervalos contiguos en cada uno de los ejes. La intersección de los subconjuntos candidatos a contener el vecino más próximo da lugar a una celda hiper-rectangular que contiene el subconjunto de prototipos sobre el que finalmente se realiza la búsqueda exhaustiva.

1.2.4. Búsqueda de vecinos en espacios métricos

Estos métodos son especialmente interesantes porque pueden ser aplicados a todos los problemas prácticos que puedan resolver los de la sección anterior, más algunos otros que esos métodos no pueden abordar debido a que no se puede obtener una *representación vectorial de los datos*. Para ello,

lo único que se requiere es la definición de una medida de disimilitud entre cada par de puntos en el espacio de representación, y que esa medida de disimilitud sea una métrica.

Haciendo una descripción cronológica de algunos de estos métodos, uno de los primeros fue el trabajo de Fischer y Patrick (1970). Estos autores presentan un preproceso en el que ordenan el conjunto de prototipos y los almacenan en una lista. La ordenación realizada es utilizada junto con la desigualdad triangular para evitar el cálculo de muchas distancias entre los prototipos y la muestra a clasificar [FP, 70].

Otro grupo de métodos propuestos utilizan estructuras arborescentes para almacenar los prototipos [FN, 75] [KM, 83] [KPK, 85]. En este caso el preproceso consiste básicamente en una descomposición jerarquizada del conjunto de prototipos en subconjuntos disjuntos. En estos métodos, la búsqueda se basa en técnicas de ramificiación y poda para reducir el número de prototipos examinados [HS, 78]. Un ejemplo, ya clásico dentro de este grupo, lo constituye el algoritmo propuesto por Fukunaga y Narendra [FN, 75]. Cada nodo p del árbol T, obtenido por descomposición del conjunto de prototipos P, representa un subconjunto $S_p \subset P$. La información real que se almacena en cada nodo es el número de prototipos pertenecientes al subconjunto, N_p, el centroide de dicho conjunto m_p y el radio del nodo definido como

$$r_p = \max_{x_i \in S_p} d(x_i, m_p). \tag{1.7}$$

En las hojas se almacena, además, la lista de prototipos pertenecientes a la misma, S_p. El método utiliza dos reglas de eliminación (una para los nodos internos y otra para las hojas) en el proceso de búsqueda. Estas reglas de eliminación están definidas en función de la información almacenada en los nodos. Un nodo interno p puede ser eliminado si se cumple:

$$D_{\min} + r_p < d(x, m_p) \tag{1.8}$$

donde $D_{\min}$ es la distancia del prototipo más cercano a la muestra hasta el momento y x es la muestra a clasificar. La regla de eliminación para los nodos hoja se aplica individualmente a cada uno de los prototipos. Cada uno de estos prototipos es eliminado si se cumple:

$$D_{\min} + d(x_i, m_p) < d(x, m_p) \tag{1.9}$$

donde $x_i \in S_p$. El proceso de búsqueda comienza expandiendo los hijos del nodo actual (empezando por la raíz) en la lista de *nodos vivos* en ese nivel (ramificación). Después de comprobar la regla de eliminación (1.8) (poda) para los nodos expandidos, se repite el proceso anterior para "el mejor" nodo

entre los supervivientes (el nodo para el que la distancia del centroide a la muestra es menor). Si no ha quedado ningún nodo se vuelve al nivel anterior del árbol. Cuando el nodo seleccionado es un nodo hoja, se aplica la segunda regla de eliminación (1.9) a cada uno de los prototipos pertenecientes al mismo. El proceso termina cuando, en la vuelta atrás, se llega al nodo raíz. Este método fue posteriormente mejorado por Kamgar-Parsi y Kanal (1985) con la incorporación de dos reglas más de eliminación [KPK, 85].

El coste *espacial* en casi todos los métodos mencionados es lineal con el número de prototipos. Sin embargo, para muchas aplicaciones es a veces preferible un coste espacial mayor a cambio de una reducción en el número de distancias a calcular [Vid, 86] [SW, 90].

Concretamente, el algoritmo conocido como AESA (Eliminating Approximating Search Algorithm) reduce significativamente el número de distancias calculadas respecto a otros métodos anteriormente propuestos. Los resultados empíricos obtenidos demuestran que el número de distancias calculadas para obtener el vecino más próximo permanece *constante* en promedio con el tamaño del conjunto de prototipos [Vid, 86] [Vid, 94]. Debido a la importancia que va a tener este algoritmo en el desarrollo de este trabajo, dedicaremos un apartado para la explicación detallada de su funcionamiento. Un estudio teórico en el que puede encuadrarse este algoritmo y los resultados empíricos mencionados ha sido presentado recientemente por Faragó, Linder y Lugosi en [FLL, 93].

Realizar la comparación de todos los métodos propuestos es muy difícil debido a que cada autor presenta los resultados asociados al comportamiento de cada algoritmo de diferente forma: tiempo consumido en la fase de búsqueda o número de distancias calculadas. A su vez, en cada caso, las condiciones de los problemas son diferentes: distribución de los prototipos, tamaño del conjunto de prototipos, etc. Además, en muy pocos casos se hace un estudio del coste asociado a la fase de búsqueda y la de preproceso. De todas formas, todos los métodos mencionados en esta sección presentan algún tipo de cuello de botella: en unos casos el coste espacial y/o temporal y en otros el número de distancias a calcular.

Algoritmo AESA

El AESA (Approximating Eliminating Search Algorithm) es un algoritmo propuesto por Vidal en 1985 y que, como su propio nombre indica, consta principalmente de dos fases: *aproximación y eliminación*. Asumiendo un orden arbitrario en el conjunto de prototipos P, este método utiliza una matriz triangular de $|P|(|P|-1)/2$ distancias entre prototipos, calculadas en la *fase de preproceso*. Esta matriz de distancias será utilizada en las dos fases

principales del algoritmo: 1) para guiar una búsqueda dinámica rápida por aproximación sucesiva hacia el prototipo más cercano, y 2) para dar soporte a las reglas de eliminación ([Vid, 85]).

Concretamente, la estrategia de búsqueda es la principal aportación del algoritmo. En esta fase se selecciona un prototipo *aproximadamente* cercano a la muestra, se calcula la distancia entre ambos y se actualiza el vecino más próximo si esta distancia es menor que la del más cercano hasta el momento. En la segunda fase (eliminación), usando la desigualdad triangular, se eliminan para posteriores búsquedas todos los prototipos que no pueden estar más cercanos a la muestra que el actualmente seleccionado como tal. El procedimiento termina cuando el conjunto de prototipos queda vacío.

A continuación se resumen las características principales de este algoritmo. Sea E un conjunto dotado de una métrica $d : E \times E \rightarrow \mathbb{R}$, y sea P un subconjunto finito de puntos ó elementos de E. Esta medida de disimilitud cumple las propiedades métricas que serán utilizadas para obtener reglas de aproximación y eliminación eficientes que nos conducirán a la obtención rápida del vecino más próximo. Para ver cómo se aplican las propiedades métricas de la medida de disimilitud utilizada, aparte del conjunto de prototipos P, se utiliza un conjunto U constituído por todos aquellos prototipos para los que ya se ha calculado (y almacenado) su distancia a la muestra x. Si $u \in U$, y n es el actual vecino más cercano a la muestra x ($d(x,n) \leq d(x,u)\ \forall u \in U$), la desigualdad triangular propia de una métrica se aplica para obtener las siguientes desigualdades [Vid, 94]:

$$|d(x,q) - d(q,p)| \leq d(x,p) \quad \forall p \in P,\ q \in U,\ x \in E \tag{1.10}$$

y como consecuencia:

$$\max_{\forall q \in U} |d(x,q) - d(q,p)| \leq d(x,p), \ \text{donde}\, U \subset E \tag{1.11}$$

Esta característica permite definir la función *cota inferior de la distancia* de la siguiente forma:

$$g_U(p) = \max_{q \in U} |d(x,q) - d(q,p)| \quad \forall p \in P \tag{1.12}$$

con $g_U(p) = 0$ si $U = \emptyset$.

Así pues, aplicando las propiedades métricas, se obtienen las condiciones suficientes para que un prototipo p no pueda estar más cercano a la muestra x que el más cercano, n, hasta el momento (ver figura 1.7):

"Todo prototipo p tal que $g_U(p) > d(x,n)$ es descartado para calcular su

distancia a la muestra x, por lo tanto, puede ser eliminado de P"

Una vez eliminados los prototipos que cumplen la condición, el procedimiento continúa seleccionando un nuevo prototipo no eliminado como candidato para calcular su distancia a la muestra. A continuación se aplica la regla de eliminación de la forma indicada anteriormente para los prototipos no eliminados. La condición de parada se cumple cuando todos los prototipos han sido eliminados, bien porque se les ha aplicado la regla de eliminación o porque han sido seleccionados para calcular su distancia a la muestra.

La eficiencia de este procedimiento aumenta (se eliminan más prototipos) con la cercanía del prototipo seleccionado para calcular su distancia a la muestra. Por ello, para minimizar el número de distancias a calcular, es conveniente seleccionar en las primeras etapas del procedimiento, prototipos candidatos que se encuentren lo más cerca posible a la muestra. Para conseguirlo, Vidal propone el siguiente criterio de aproximación:

"*Seleccionar el prototipo no eliminado* $s \in P$ *más cercano a la intersección de todas las hiperesferas con radio* $d(x,u)$ $\forall u \in U$*, centradas en cada prototipo* u *utilizado*"

Aunque esta intersección no puede obtenerse explícitamente, este criterio puede ser aproximado numéricamente de la siguiente forma [Vid, 94]:

$$s = \text{argmin}_{q \in P} g_U(q) \tag{1.13}$$

En la figura 1.8 se puede ver una ilustración gráfica de este criterio de aproximación.

El algoritmo (ver figura 1.9) encuentra el vecino más próximo calculando un número de distancias independiente del número de prototipos del conjunto P. Además, otra ventaja que presenta es que para obtener el vecino más próximo no necesita conocer las coordenadas de los datos sino que sólamente necesita las propiedades métricas del espacio. Este hecho es muy importante, ya que en muchos experimentos reales las coordenadas no se suelen conocer. Sin embargo, este algoritmo presenta dos inconvenientes que pueden hacer difícil su uso con conjuntos de prototipos relativamente grandes. El primer inconveniente, y más importante, es que el *AESA* necesita calcular y, lo peor, almacenar en preproceso las interdistancias de los prototipos del conjunto P. El otro inconveniente es el coste temporal del proceso de búsqueda, que es lineal con el tamaño del conjunto de prototipos. Esto limita su uso a conjuntos relativamente pequeños de prototipos. De todas formas, este segundo problema no es tan importante como el anterior en muchas aplicaciones de interés en las que lo verdaderamente caro es el cálculo de cada distancia.

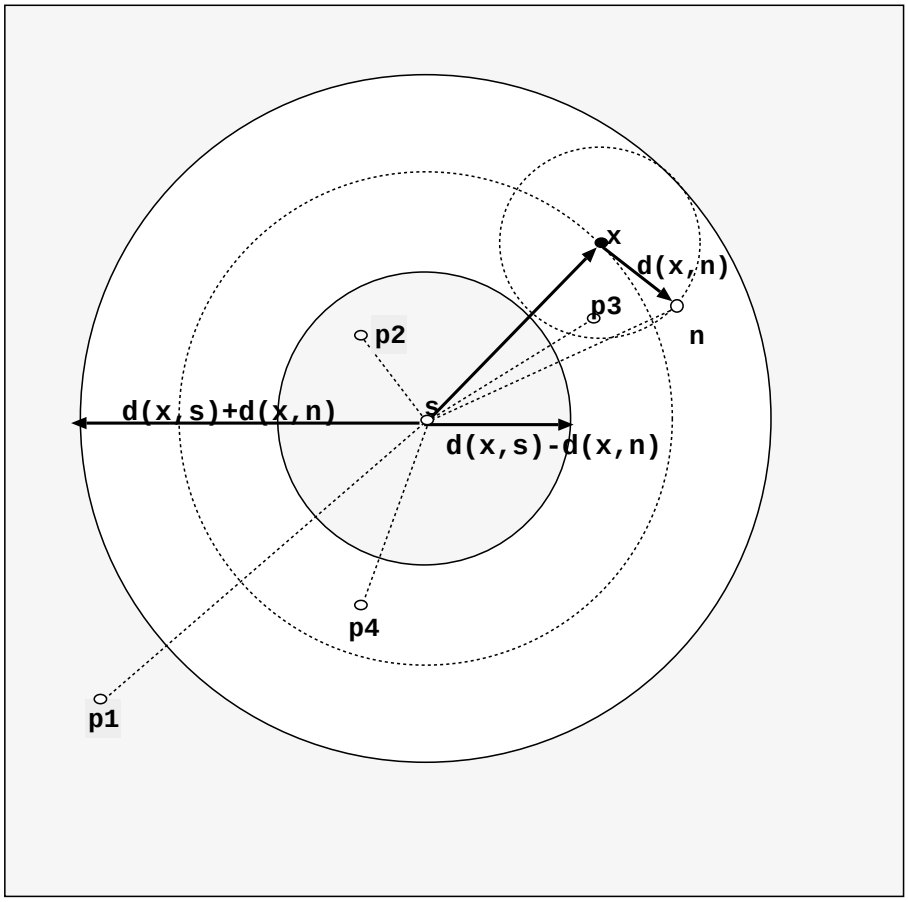

Figura 1.7: Criterio de eliminación en el plano euclídeo. Todos aquellos prototipos que se encuentran en la zona sombreada no pueden estar más cercanos a la muestra x que el actual vecino más próximo n, y por lo tanto pueden ser eliminados.

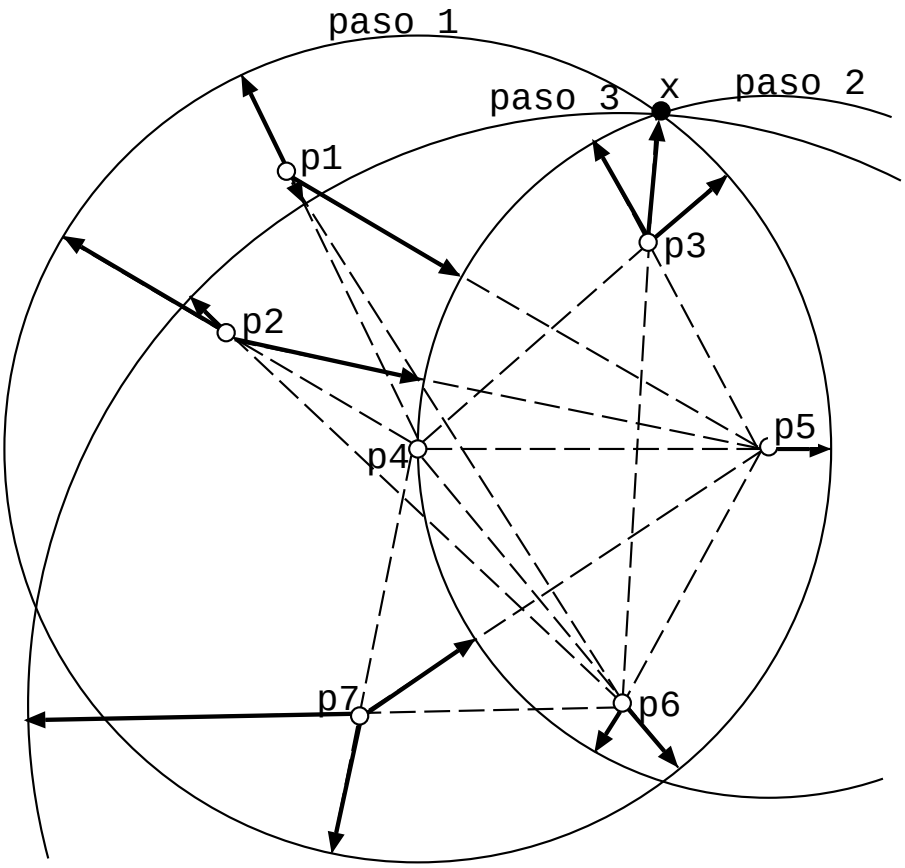

Figura 1.8: Criterio de aproximación en el plano euclídeo. En cada paso, se selecciona como prototipo candidato aquel que tiene un menor valor de la función cota inferior de la distancia (flecha más larga en cada uno de los prototipos). Tras la tercera iteración, el prototipo seleccionado es el más cercano a la muestra.

Algoritmo AESA
Entrada:
$P \subset E$ *// subconjunto finito de prototipos*
$M \in \mathbb{R}^{|P|\times|P|}$ *// matriz precalculada de $|P| \times |P|$ distancias entre prototipos*
$x \in E$ *// muestra*
Salida:
$min \in P$ *// prototipo de P más cercano a la muestra*
$D_{\min} \in \mathbb{R}$ *// distancia del prototipo más cercano a la muestra*
Función: $d : E \times E \to \mathbb{R}$ *// función distancia*
Variables:
$g \in \mathbb{R}^{|P|}$ *// vector de cotas inferiores*
$p, s, s_1 \in P$
$dxs \in \mathbb{R}$
Método:
$D_{\min} = \infty; min =$ elemento_aleatorio(P);
$g = \{0, \ldots, 0\}; s =$ elemento_aleatorio(P);
mientras $|P| > 0$ **hacer**
$dxs = d(x, s)$; $P = P - \{s\}$; *// cálculo de la distancia*
si $dxs < D_{\min}$ **entonces** *// actualiza mejor solución*
$D_{\min} = dxs$; $min = s$;
fin si
$s_1 = s; s =$ elemento_aleatorio(P);
para todo $p \in P$ **hacer** *// bucle de aprox. y eliminación*
$g[p] = \max(g[p], |dxs - M(s_1, p)|)$;
si $g[p] \geq D_{\min}$ **entonces** $P = P - \{p\}$; *// eliminación*
si no
si $g[p] < g[s]$ **entonces**
$s = p$; *// aproximación*
fin si
fin si
fin para todo
fin mientras
fin Método
fin Algoritmo

Figura 1.9: Algoritmo *AESA*

1.3. Justificación y desarrollo del proyecto

En este trabajo proponemos nuevos algoritmos de la familia AESA que no necesitan que los datos estén representados en espacios vectoriales y que, manteniendo el número de distancias calculadas *constante* en promedio, tienen un coste espacial *lineal* (en el AESA es cuadrática). Estos algoritmos recibirán el nombre LAESA (Linear AESA). Esta reducción en el coste se va a conseguir gracias a la utilización de un subconjunto del conjunto de prototipos P al que llamaremos conjunto de *prototipos base*. La matriz de distancias a calcular será por tanto la del conjunto de prototipos P a este subconjunto. Siguiendo una línea de trabajo como en [Sha, 77], se trata pues de elegir este subconjunto de forma que cumpla unas condiciones que haga posible el comportamiento deseado del algoritmo. Estos algoritmos constarán principalmente de dos fases en las que se distingue si los prototipos seleccionados o eliminables pertenecen o no al conjunto de *prototipos base*. Dependiendo de criterios que se estudiarán más detenidamente, se elige entre prototipos base y no-base en la selección. También, en la eliminación, se utilizan diferentes criterios para permitir, aun cumpliendo las condiciones exigidas, la eliminación o no dependiendo del tipo de prototipo con el que tratemos.

Por último, se propone una serie de algoritmos con los que se consigue reducir el coste temporal de los anteriores a cotas sublineales (en la familia LAESA el coste es lineal) manteniendo las otras características: número medio de distancias constante y coste computacional (overhead) lineal. Esto se consigue gracias a la utilización de estructuras de datos que permiten manipular grupos de prototipos en lugar de prototipos individuales. Comprobaremos experimentalmente que estos algoritmos son mejores que la familia LAESA (mejora del coste computacional) cuando trabajamos con conjuntos de prototipos muy grandes.

Parte II

Nuevos algoritmos de búsqueda de vecinos en espacios métricos

Capítulo 2

Búsqueda por aproximación y eliminación

En este trabajo se presentan varios métodos de búsqueda rápida de los vecinos más próximos. La propiedad común a todos ellos es que son aplicables a cualquier problema de reconocimiento de formas para el que se haya definido una distancia entre los objetos a reconocer. Estos métodos, por tanto, son más generales que aquellos que, además de la definición de la distancia, requieren una adecuada representación de los objetos en un espacio vectorial [Yun, 76] [FBF, 77] [RP, 90].

En el capítulo 1 se ha realizado una revisión de algunas técnicas de búsqueda de vecinos más próximos aparecidas hasta el momento y, en particular, de aquellas que pertenecen al grupo que nos interesa, es decir, aquellas que solo requieren el establecimiento de unas propiedades métricas en el espacio de búsqueda para su utilización. Dentro de este grupo se encuentra el algoritmo AESA (Approximating-Eliminating Search Algorithm) que es uno de los más interesantes de los aparecidos hasta el momento porque para obtener el vecino más próximo de un conjunto de prototipos P a una muestra x dada, calcula un número de distancias independiente, en promedio, del tamaño de este conjunto. Esta característica es muy importante, sobre todo, si las distancias con las que se trabaja son especialmente caras. Sin embargo, para obtener estos buenos resultados el algoritmo necesita calcular y almacenar en la fase de preproceso una matriz de distancias entre cada par de prototipos del conjunto P. Este almacenamiento cuadrático respecto a $|P|$ hace que sea problemática su utilización con conjuntos grandes de prototipos.

El objetivo de este trabajo es obtener una familia de algoritmos de búsqueda de vecinos más próximos que mantengan la característica principal del AESA (número medio de distancias independiente del tamaño del conjunto de prototipos), que reduzcan el coste cuadrático de almacenamiento a cotas

lineales. Para ello, se propone reducir el tamaño de la matriz de distancias que utiliza el AESA almacenando únicamente las distancias entre todos los prototipos del conjunto P y un subconjunto del mismo de tamaño independiente del tamaño de P. Lógicamente el algoritmo de búsqueda debe estar adaptado a la información de que va a disponer, la cual ha disminuido con la reducción del tamaño de la matriz. En este capítulo se va a estudiar una primera aproximación al objetivo planteado. En los siguientes, se realizará un detallado estudio experimental (con datos sintéticos) de todas las modificaciones que se han ido introduciendo en el algoritmo original para mejorar sus prestaciones. Por último se utilizan datos reales para comprobar la efectividad de los algoritmos propuestos. Concretamente, los algoritmos han sido aplicados al reconocimiento automático de caracteres manuscritos.

2.1. Preproceso

El algoritmo de búsqueda que se propone requiere una fase de preproceso en la que se calcula y almacena información que será utilizada posteriormente en las distintas fases del mismo. Esta fase es crucial para el objetivo planteado ya que en ella se obtendrá la reducción del coste espacial del algoritmo respecto al tamaño del conjunto de prototipos P. El procedimiento consiste en la selección de un subconjunto B del conjunto de prototipos P, de forma que $|B| \ll |P|$. La forma concreta de realizar esta selección será estudiada más detenidamente en el próximo capítulo. La elección del tamaño de B es un parámetro, en principio independiente del tamaño de P, que se debe ajustar experimentalmente. Los prototipos seleccionados, a los que se llamará *prototipos base*, (PB), serán utilizados para calcular las distancias al resto de prototipos de P, las cuales serán almacenadas en una matriz. Esta matriz de distancias de tamaño $|B| \times |P|$, crece, por tanto, *linealmente* con el tamaño de P, con un factor constante que depende de $|B|$.

En resumen, sea E el espacio de representación de los objetos, sea $P = \{p_1 \dots p_{|P|}\} \subset E$ el conjunto (finito) de prototipos, y sea $B = \{b_1, \dots b_{|B|}\}$ el conjunto de PB, donde $|B| \ll |P|$. Cada una de las componentes de la matriz de distancias $M \in \mathbb{R}^{|B| \times |P|}$ viene dada por

$$m_{ij} = d(b_i, p_j), \forall\, b_i \in B,\ \forall\, p_j \in P. \tag{2.1}$$

El siguiente paso es conseguir mantener constante el número medio de distancias respecto al tamaño del conjunto de prototipos aprovechando la información que da la matriz reducida de distancias M. Tanto la elección de los elementos del subconjunto B, como el valor de $|B|$ necesarios para maximizar la eficiencia del algoritmo serán discutidos en el próximo capítulo.

2.2. El algoritmo LAESA

El algoritmo que se propone utiliza la información calculada y almacenada en la fase de preproceso durante la fase de búsqueda. Concretamente, esta información será utilizada en las dos fases principales del algoritmo: *aproximación* (elección de un prototipo *probablemente* muy cercano a la muestra) y *eliminación* (de todos aquellos prototipos que no puedan estar más cercanos a la muestra que el más cercano hasta el momento). Además, utilizando la desigualdad triangular es posible calcular de forma sencilla una cota inferior de la distancia de cada prototipo a la muestra. Esta cota será utilizada en las fases de aproximación y eliminación del algoritmo. A continuación se describe cómo se calcula esta cota.

2.2.1. Función cota inferior de la distancia

Sea $d : E \times E \rightarrow \mathbb{R}$ una distancia definida en el espacio de representación E. Sea P un conjunto de prototipos y $B \subseteq P$ el conjunto de PB. Sea x una muestra y $U \subseteq P$ el conjunto de prototipos $u \in B$ para los que la distancia $d(x, u)$ ya ha sido calculada y almacenada en los pasos previos del proceso de búsqueda. Entonces, para todo $p \in P$, una cota inferior de la distancia de x a p, $g_U(p)$, puede derivarse de forma directa a partir de la desigualdad triangular de la medida de disimilitud dada [Vid, 94]:

$$d(x,p) \geq g_U(p) = \begin{cases} 0 & \text{si } U \cap B = \emptyset \\ \text{máx}_{\forall u \in U} |d(p,u) - d(x,u)| & \text{en cualquier otro caso} \end{cases} \tag{2.2}$$

Esta función es poco costosa de calcular en el proceso de búsqueda ya que tanto $d(p, u)$ como $d(x, u)$ han sido calculadas y almacenadas previamente. Esta función tiene el mismo perfil que la función distancia: $g_U : E \times E \rightarrow \mathbb{R}$, con la diferencia de que uno de los dos objetos pertenecientes al espacio de representación E es siempre la muestra a reconocer, por ello la expresamos como si fuese una función de una sola variable: $g_U(x, p) \equiv g_U(p)$.

2.2.2. Proceso de eliminación

A continuación se estudia el proceso de eliminación del algoritmo, que se basa en la función cota inferior.

Sea b un prototipo base y n el prototipo más cercano hasta el momento a la muestra x. Según la ecuación 2.2 un prototipo $p \in P - B$ puede ser eliminado del proceso de búsqueda si su cota inferior de distancia es mayor

que la distancia de x a n; es decir, una condición suficiente para la eliminación es la que aparece en la figura 2.1

$$g_B(p) = |d(p,b) - d(x,b)| \geq d(x,n) \tag{2.3}$$

o, lo que es lo mismo:

$$d(p,b) \geq d(x,b) + d(x,n)$$
$$d(p,b) \leq d(x,b) - d(x,n)$$

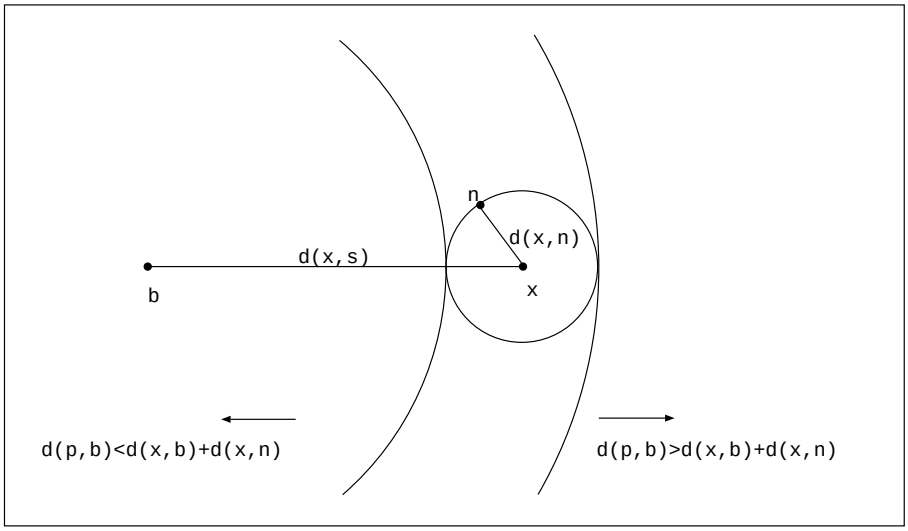

Figura 2.1: Criterio de eliminación.

2.2.3. Proceso de aproximación

En la fase de aproximación se trata de encontrar un prototipo s candidato a ser el vecino más próximo. La elección de este candidato conlleva posteriormente el cálculo de la distancia de ese prototipo a la muestra y, como consecuencia, la (posible) actualización de la función cota inferior de la distancia del resto de prototipos no eliminados. Por ello, mientras queden PB, éstos son elegidos siempre como candidatos. De esta forma se puede realizar la actualización mencionada anteriormente en las primeras etapas del proceso de búsqueda. Una vez que han sido utilizados todos los PB, se procede a la selección del resto de prototipos siguiendo el mismo criterio de aproximación utilizado por Vidal [Vid, 94], es decir, el siguiente prototipo candidato a ser

el vecino más próximo es aquel que tiene un valor menor de la función cota inferior:

$$s = \text{argmin}_{\forall q \in P-F} g_U(q) \tag{2.4}$$

y donde:

P es es conjunto total de prototipos
F es el conjunto de prototipos eliminados y
U es el conjunto de prototipos base para los que ya se ha calculado su distancia a la muestra

En la figura 2.2 se puede ver un ejemplo gráfico del funcionamiento de este criterio de aproximación en un espacio vectorial euclídeo de dimensión 2.

2.2.4. Algoritmo LAESA

Antes de desarrollar con detalle el algoritmo propuesto se pasa a expresar brevemente su funcionamiento a partir de los pasos principales del mismo:

1. **Iniciación.** Consiste en la selección aleatoria de un primer prototipo base para calcular su distancia a la muestra x. Iniciación a un valor infinito de la distancia $D_{\min}$ del prototipo más cercano a la muestra hasta el momento, y a 0 el valor de la cota inferior de la distancia para cada prototipo, $g_U(p)$.

2. **Cálculo de la distancia** del prototipo seleccionado s a la muestra x

3. **Actualización** del prototipo más cercano y de la función cota inferior de la distancia de cada prototipo.

4. **Eliminación.** Todos los prototipos que no son base y que cumplen la regla de eliminación se descartan del conjunto P.

5. **Aproximación.** Obtención del prototipo seleccionado s para calcular su distancia a la muestra en la siguiente iteración.

Siguiendo estos pasos el algoritmo se divide en dos partes bien diferenciadas una vez efectuada la iniciación (paso 1). En la primera existe un bucle principal en el que se efectúan todos los pasos del algoritmo con alguna pequeña modificación. El número de veces que se repite este bucle está acotado por el número de PB, pero puede finalizar antes si el conjunto P se queda vacío por la aplicación del paso 4. En esta parte la fase de aproximación (paso 5) se reduce simplemente a elegir aleatoriamente un PB. En el paso 2 se

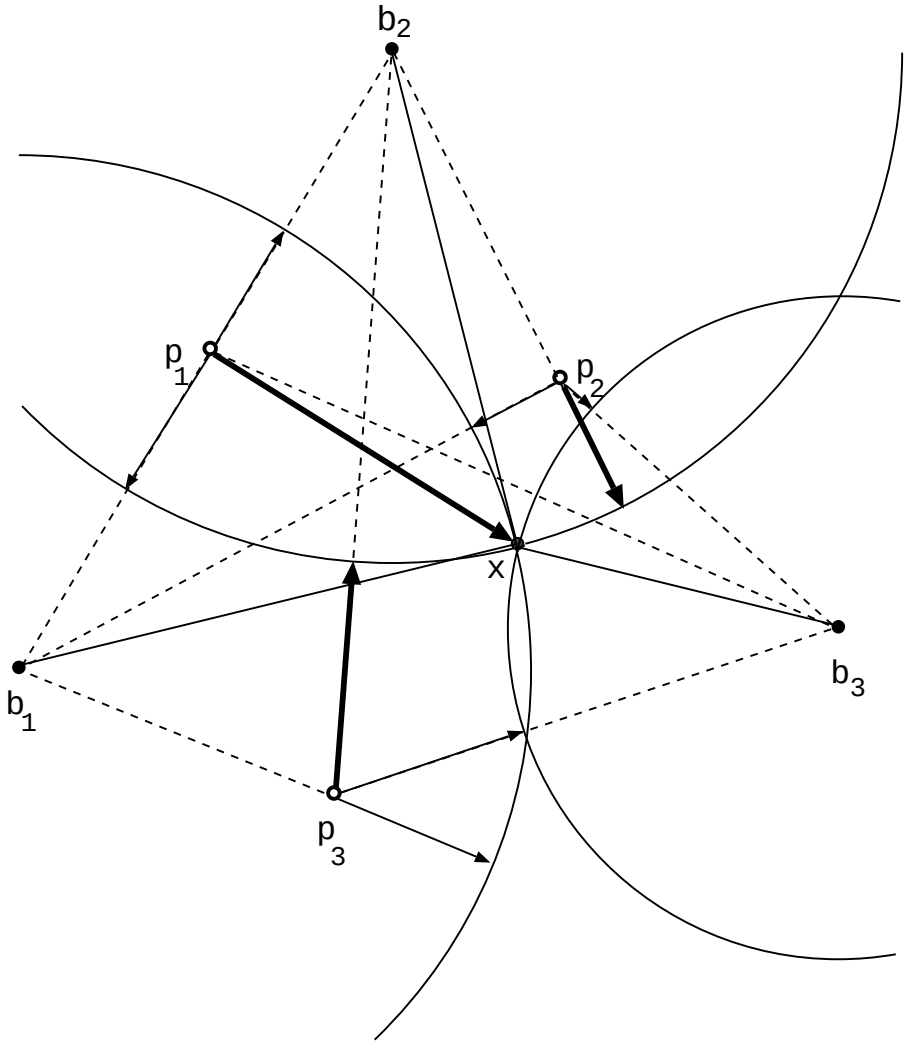

Figura 2.2: Criterio de aproximación. En el ejemplo, $\mathtt{b}_1$, $\mathtt{b}_2$ y $\mathtt{b}_3$ son PB y $\mathtt{p}_1$, $\mathtt{p}_2$ y $\mathtt{p}_3$ son prototipos de P. El prototipo seleccionado como nuevo candidato para calcular su distancia a x será p_2 por ser aquel para el cual el valor de la función cota inferior de la distancia es menor.

calcula la distancia del prototipo base aleatoriamente seleccionado a la muestra. A continuación con otro bucle interno se recorren todos los prototipos $p \in P - B$ sobre los que se realizan los pasos 3 y 4, en este orden.

Si después de esta primera parte, aún quedan prototipos en $P - B$, se repite el proceso con ciertas modificaciones. En el bucle principal se calcula la distancia del prototipo seleccionado a la muestra (paso 2) y en el bucle interno que contiene se recorren todos los prototipos que todavía no han sido eliminados realizándose los pasos 4 y 5. En este caso no se realiza el paso 3 pues no tendríamos información suficiente para actualizar el valor de g (no tenemos almacenadas las distancias entre prototipos que pertenecen a $P-B$). En la fase de aproximación, sin embargo, se utiliza el criterio visto en la sección anterior aplicada al conjunto de prototipos de $P-B$. El algoritmo por tanto queda de la forma que podemos observar en la figura 2.3.

Puede obtenerse una versión más compacta del algoritmo unificando los dos bucles principales en uno solo. De esta forma el esqueleto básico del algoritmo es el mismo que el del AESA. Esta segunda versión la podemos ver en la figura 2.4 [MOV, 91].

Algoritmo
Entrada:
$P \subset E$ *// subconjunto finito de prototipos*
$B \subset P$ *//conjunto de prototipos base*
$M \in \mathbb{R}^{|P|\times|B|}$ *// matriz precalculada de $|P| \times |B|$ distancias entre prototipos*
$x \in E$ *// muestra*
Salida:
$min \in P$ *// prototipo de P más cercano a la muestra*
$D_{\min} \in \mathbb{R}$ *//distancia del prototipo más cercano a la muestra*
Función: $d : E \times E \to \mathbb{R}$ *// función distancia*
Variables:
$g \in \mathbb{R}^{|P|}$ *// vector de cotas inferiores*
$p, s \in P$, $dxs \in \mathbb{R}$
Método:
$D_{\min} = \infty; min = \text{elemento\_aleatorio}(P);$
$g = \{0, \ldots, 0\};$
mientras $|B| > 0\, y\, |P - B| > 0$ **hacer**
$s = \text{elemento\_aleatorio}(B);\ B = B - \{s\};$
$dxs = d(x, s);$ *// cálculo de la distancia*
si $dxs < D_{\min}$ **entonces** *// actualiza mejor solución*
$D_{\min} = dxs;\ min = s;$
fin si
para todo $p \in P - B$ **hacer** *// bucle de aprox. y eliminación*
$g[p] = \max(g[p], |dxs - M(s, p)|);$
si $g[p] \geq D_{\min}$ **entonces** $P = P - \{p\}$; **fin si** *// eliminación*
fin para todo
fin mientras
si $|P - B| > 0$ **entonces** $s = \text{argmin}_{p\in P} g[p]$; **fin si**
mientras $|P| > 0$ **hacer**
$dxs = d(x, s); P = P - \{s\};$ *// cálculo de la distancia*
si $dxs < D_{\min}$ **entonces** *// actualiza mejor solución*
$D_{\min} = dxs;\ min = s;$
fin si
$s = \text{elemento\_aleatorio}(P);$
para todo $p \in P$ **hacer** *// bucle de aprox. y eliminación*
si $g[p] \geq D_{\min}$ **entonces** $P = P - \{p\};$ *// eliminación*
si no
si $g[p] < g[s]$ **entonces**
$s = p;$ *// aproximación cuando ya no quedan PB*
fin si
fin si
fin para todo
fin mientras
fin Método
fin Algoritmo

Figura 2.3: Algoritmo *LAESA*, versión en dos partes

Algoritmo
Entrada:
$P \subset E$ *// subconjunto finito de prototipos*
$B \subset P$ *//conjunto de prototipos base*
$M \in \mathbb{R}^{|P|\times|B|}$ *// matriz precalculada de $|P| \times |B|$ distancias entre prototipos*
$x \in E$ *// muestra*
Salida:
$min \in P$ *// prototipo de P más cercano a la muestra*
$D_{\min} \in \mathbb{R}$ *// distancia del prototipo más cercano a la muestra*
Función: $d : E \times E \to \mathbb{R}$ *// función distancia*
Variables:
$g \in \mathbb{R}^{|P|}$ *// vector de cotas inferiores*
$p, s, s_1 \in P$
$dxs \in \mathbb{R}$
$sdB \in Booleano$ *// booleana que nos indica si $s \in B$*
Método:
$D_{\min} = \infty; min =$ elemento_aleatorio(P);
$g = \{0, \ldots, 0\}$;
mientras $|P| > 0$ **hacer**
si $|B| > 0$ **entonces**
$s =$elemento_aleatorio(B); $B = B - \{s\}$;
$sdB = verdadero$
si no
$sdB = falso$
fin si
$dxs = d(x, s)$; *// cómputo de la distancia*
si $dxs < D_{\min}$ **entonces** *// actualiza mejor solución*
$D_{\min} = dxs$; $min = s$;
fin si
$P = P - \{s\}; s_1 = s; s =$elemento_aleatorio$(P)$;
para todo $p \in P - B$ **hacer** *// bucle de aprox. y eliminación*
si sdB **entonces**
$g[p] = \max(g[p], |dxs - M(s_1, p)|)$;
fin si
si $g[p] \geq D_{\min}$ **entonces** $P = P - \{p\}$; *// eliminación*
si no
si $|B| = 0$ y $g[p] < g[s]$ **entonces**
$s = p$; *// aproximación cuando ya no quedan PB*
fin si
fin si
fin para todo
fin mientras
fin Método
fin Algoritmo

Figura 2.4: Algoritmo *LAESA*, versión en una parte.

2.3. Experimentos

En los experimentos que se presentan a continuación no se ha utilizado ninguna técnica especial de selección de PB; concretamente han sido escogidos de forma aleatoria dentro del conjunto de prototipos P. Tampoco ha sido un objetivo en este apartado estudiar cuál sería el tamaño del conjunto B, ya que estos son aspectos que se estudiarán en el siguiente capítulo. Los resultados empíricos que se presentan en esta sección se han obtenido utilizando la métrica euclídea para distintos tamaños del conjunto de prototipos P. Estos prototipos han sido extraídos a partir de distribuciones uniformes de diferentes dimensiones en el hipercubo unidad.

Para cada uno de los experimentos presentados en este capítulo y en los siguientes de esta misma parte, se extrajeron diez conjuntos aleatorios de prototipos a partir de la misma distribución. Para cada uno de estos conjuntos se extrajeron mil muestras de la misma distribución, aplicándose el algoritmo a cada una de ellas. El método de estimación de las prestaciones medias consistió en un *muestreo en dos etapas* [Coc, 77]: para cada uno de los conjuntos se calcula la media del número de cómputos de distancia, y estas medias resultantes se promedian de nuevo sobre los conjuntos extraídos. Este procedimiento permite obtener estimaciones estadísticamente más consistentes que un muestreo convencional. Todos los resultados presentados en este trabajo tienen asociado un intervalo de confianza del 95 %.

En el primer experimento se eligieron aleatoriamente varios conjuntos de PB para una distribución uniforme sobre un espacio vectorial euclídeo de dimensión 6. En la figura 2.5 se puede observar que el comportamiento del algoritmo varía según la elección del tamaño de estos conjuntos. Concretamente, a medida que aumenta el tamaño del conjunto de PB, llega un momento en que el número de distancias calculadas coincide con el mismo. Esto puede hacer pensar que no será necesario aumentar innecesariamente el tamaño de este conjunto. En la figura también se puede observar que a partir de un conjunto de 10 prototipos base, el número medio de distancias calculadas es prácticamente independiente del número de prototipos.

Repitiendo el experimento anterior con un conjunto de 20 prototipos base para diferentes dimensiones, se observó que el número de distancias permanece constante en promedio conforme aumenta el número de prototipos y por otra parte el número de distancias aumenta con la dimensión. Concretamente, en la figura 2.6 se puede ver este comportamiento, donde además parece intuirse ya para la curva de dimensión 8 que si se sigue aumentando la dimensión va a ser necesario aumentar el número de PB para que siga manteniéndose constante el número medio de distancias. En el próximo capítulo se verá que, efectivamente será necesario aumentar el número de PB con la

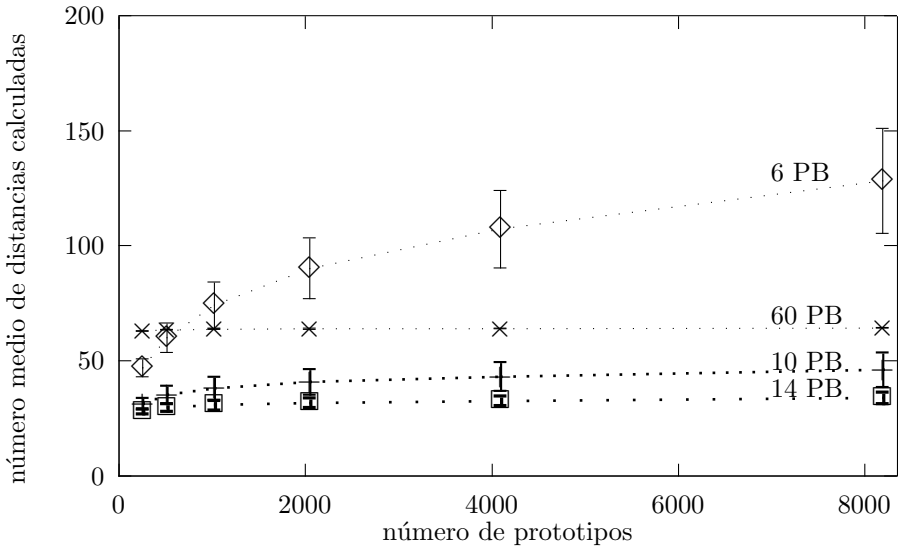

Figura 2.5: Número medio de distancias calculadas por el algoritmo, al variar el tamaño del conjunto de prototipos en un espacio vectorial euclídeo de dimensión 6 sobre una distribución uniforme. El experimento ha sido repetido utilizando como tamaños del conjunto de PB los valores 6, 10, 14 y 60.

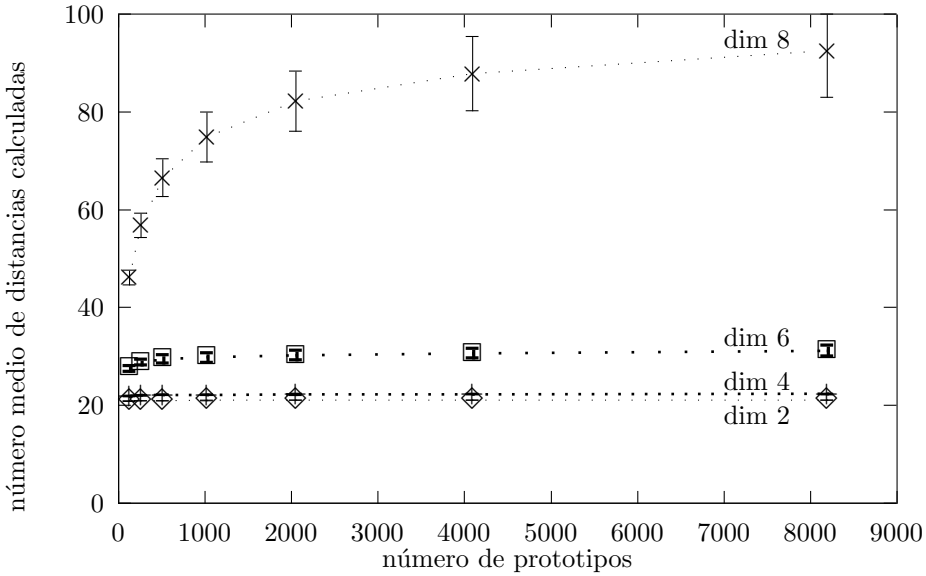

Figura 2.6: Número medio de distancias calculadas por el algoritmo, al variar el tamaño del conjunto de prototipos en un espacio vectorial euclídeo y la dimensión, usando una distribución uniforme. Para el experimento se han utilizado 20 prototipos base.

dimensión de los prototipos, aunque eso no ha impedido demostrar experimentalmente que para un tamaño determinado del conjunto de PB el número medio de distancias permanece constante.

En la figura 2.7 se puede observar el comportamiento del algoritmo en cuanto al número de distancias calculadas al variar el tamaño del subconjunto de PB y estos han sido escogidos aleatoriamente dentro del conjunto de prototipos P. Estos resultados se han obtenido utilizando un diccionario de 1024 prototipos en un espacio vectorial euclídeo de dimensión 6. Como se puede observar en la gráfica, existe un valor de $n = |B| \approx 16$, pequeño comparado con el tamaño del conjunto (1024 prototipos), para el que el número de distancias calculadas es mínimo.

2.4. Conclusiones

La principal característica del algoritmo propuesto es la reducción en el coste tanto espacial como temporal del preproceso requerido con respecto al algoritmo AESA. Concretamente, ambos han sido reducidos a cotas lineales respecto al tamaño del conjunto de prototipos. Por otro lado, el algoritmo

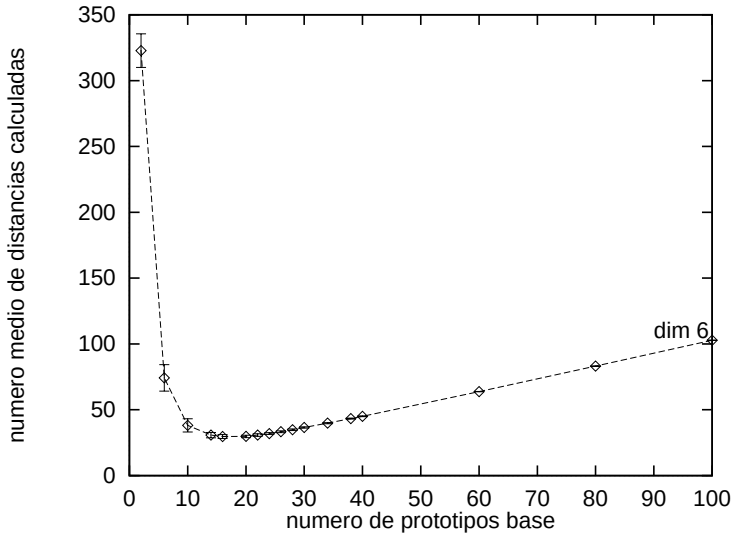

Figura 2.7: Número medio de distancias calculadas por el algoritmo, al variar el tamaño del conjunto de prototipos base en un espacio vectorial de dimensión 6 usando una distribución uniforme para un conjunto de 1024 prototipos.

consigue encontrar el vecino más próximo calculando un número de distancias constante en promedio respecto al número de prototipos del conjunto de trabajo. A partir de los resultados obtenidos se puede constatar que existe un tamaño del conjunto de PB a partir del cual el número de distancias permanece *constante* y, además, este número de distancias tiende a coincidir con el tamaño del conjunto de PB a medida que este aumenta. Se trata, pues, de hacer un estudio más detallado de esta circunstancia para independizar los efectos de estas dos variables.

Capítulo 3

Selección de prototipos base

En la estrategia de búsqueda del algoritmo propuesto en el capítulo 2 se utiliza la distancia de la muestra x a todos los PB para obtener la cota inferior de la distancia de cada prototipo a la muestra. Es de esperar que el valor de la cota dependerá de los prototipos base seleccionados, por lo tanto realizar una buena elección de este conjunto es uno de los objetivos que se plantea en este capítulo. Obviamente, la eficiencia de la búsqueda dependerá no solamente del número de PB seleccionados, sino también de su localización respecto al resto de prototipos. Esta última cuestión ya fue estudiada hace algún tiempo por Shapiro [Sha, 77] con referencia a un algoritmo de búsqueda previo de Burkhard y Keller [BK, 73]. Los resultados de este trabajo sugieren que se pueden obtener algunas mejoras si se usan *puntos de referencia* que están situados lo más alejados posible del conjunto de prototipos. Esta forma de seleccionar puntos de referencia también es utilizada por Ramasubramanian y Paliwal en [RP, 90]; sin embargo, en este caso los puntos de referencia no pertenecen al conjunto de prototipos. En [Set, 81] se propone la utilización de puntos de referencia pertenecientes al conjunto de prototipos, pero fijando su número a tres, independientemente de la dimensión, y sin hacer mención de la localización de los mismos.

El concepto de prototipo base aquí utilizado guarda cierta relación con los *puntos de referencia* que se usan en los trabajos mencionados. La diferencia fundamental respecto a algunos de los métodos mencionados viene dada porque en los que aquí se presentan, los PB seleccionados pertenecen al conjunto prototipos. Esta diferencia no se produce respecto al método propuesto en [Set, 81], sin embargo este último método no profundiza en la selección ni el número de puntos de referencia para mejorar el comportamiento del algoritmo de búsqueda.

En este capítulo se estudian diferentes técnicas para la selección de los PB. Una de las técnicas aplicables es la más conocida dentro de las técni-

cas de agrupamiento (clustering): el algoritmo tipo c-medias [DH, 73]. Se utilizarán en este caso los centroides de las clases de prototipos como PB. También se estudiarán dos técnicas que se aportan en este trabajo, y que seleccionan como PB aquellos prototipos que se encuentran muy alejados respecto al resto de prototipos. Ambos métodos utilizan una técnica voraz en la que la selección de un nuevo PB viene determinada por los que han sido seleccionados previamente. Por último se utilizará una técnica propuesta por Ramasubramanian, que selecciona como PB aquellos que minimizan el error

$$e = \sum_{p,q \in P} (d(p,q) - g(p,q)) \tag{3.1}$$

donde $g(p,q)$ es la función cota inferior de la distancia d [Ram, 92].

A continuación, después de presentar detalladamente cada una de las técnicas, se compararán aplicando a los PB seleccionados el algoritmo de búsqueda LAESA, y se justificará la elección de la técnica a utilizar en función de los resultados obtenidos y del coste espacial y temporal.

3.1. Técnicas de agrupamiento (clustering)

Dentro de las técnicas de agrupamiento, los algoritmos tipo c-medias [DH, 73] [JV, 94] son interesantes por su simplicidad y por su buen comportamiento. Estos algoritmos minimizan la suma de distancias entre todos los prototipos y los centroides de los grupos obtenidos.

Existe otro grupo de técnicas que obtienen los prototipos más representativos de un conjunto dado y que pueden ser aplicados en técnicas de agrupamiento, como es el caso. Chaudhuri et al. presentan un algoritmo que obtiene los prototipos más representativos de una distribución dada basándose en las densidades[1] de cada prototipo [CMC, 94]. De todos estos algoritmos se usará el de las c-medias por su simplicidad. A continuación, pasamos a resumir en 4 pasos este algoritmo.

Sea (E,d) un espacio métrico, y $P = \{p_1, p_2, ..., p_m\} \subseteq E$ un conjunto de prototipos a dividir en ($2 \leq K \leq n$, con $n \leq m$) agrupaciones.

- *Paso 0. Iniciación*
  Se establece una partición inicial (aleatoria) de los prototipos en K agrupaciones, $W_1 \ldots W_K$.

[1] La *densidad* de un prototipo p es el número de prototipos presentes en un disco abierto de radio θ alrededor de p. El valor de θ se obtiene a partir del árbol de expansión mínimo que se obtiene a partir del conjunto de prototipos.

- *Paso 1. Obtención de los centroides*
  Para cada agrupación W_i, se calcula para cada prototipo $p \in W_i$ la suma de distancias entre p y el resto de prototipos en W_i (distorsión de p), y se selecciona el prototipo con la distorsión mínima, J_i, como centroide de W_i.

- *Paso 2. Redistribución*
  Se cambia de clase un prototipo p aleatoriamente seleccionado. Si al efectuar el cambio el valor de la suma de las distorsiones mínimas de cada clase es mayor que la actual, deshacer el cambio; si no es así (el prototipo cambia de clase) actualizar el nuevo valor a minimizar.

- *Paso 3. Bucle*
  Si en la asignación del paso anterior el prototipo ha cambiado de clase se vuelve al paso 1, en otro caso se para.

El coste temporal de este algoritmo es, en el peor caso, $|B| \times |P|^2$, donde $|B|$ es el tamaño del conjunto de PB (número de W_i).

3.2. Selección por separación máxima

En este punto se proponen procedimientos de selección de PB de forma que estos se encuentren muy separados entre ellos y también respecto al resto de prototipos. Una forma similar de selección ha sido propuesta en otros trabajos también para la búsqueda de los vecinos más próximos [Sha, 77] [RP, 90]. A diferencia de estos, los que se proponen aquí se caracterizan porque los prototipos seleccionados pertenecen al conjunto de prototipos P sobre los que se va a realizar posteriormente la búsqueda. Para la selección de los PB solo es necesario utilizar las propiedades métricas del espacio y por lo tanto esta técnica es aplicable a una mayor cantidad de problemas prácticos.

Empezando con un prototipo base seleccionado arbitrariamente, en el primer método que se propone, denominado SMD (suma máxima de distancias), se calcula la distancia del resto de prototipos al elegido, y se seleccioná como nuevo prototipo base aquel que se encuentre a mayor distancia. Las distancias calculadas son almacenadas en una matriz para su posterior utilización en el proceso de búsqueda. Además, las distancias son sumadas en un vector acumulador que al principio tiene el valor 0. El procedimiento continúa calculando la distancia de los sucesivos prototipos base seleccionados al resto de prototipos, almacenando las distancias calculadas, sumando su valor en el vector acumulador y seleccionando el siguiente prototipo base como aquel para el que la suma de distancias acumuladas al resto de PB seleccionados es

máxima. La condición de finalización se da cuando se han obtenido el número preestablecido de PB. A la vez se ha calculado y almacenado en una matriz la distancia entre los prototipos de P y los PB [MOV, 92] [MOV, 94]:

$$B = \{b_1, \ldots, b_k\}$$

donde b_1 es un prototipo base aleatoriamente seleccionado

$$b_i = \text{argmax}_{p \in P - B_i} \sum_{k=1}^{i-1} d(p, b_k), \;\; i = 2, 3 \ldots n \tag{3.2}$$

donde $B_i = \{b_1, \ldots b_{i-1}\}$

Se ha estudiado además una variante basada en la misma idea pero en la cual, en lugar de seleccionarse en cada iteración aquel prototipo cuya suma de distancias a los previamente calculados es máxima, se selecciona aquel que se encuentra a mayor distancia de entre las distancias mínimas al resto de prototipos base previamente calculados. A este método se le denominará MDM (máximo de distancias mínimas):

$$B = \{b_1, \ldots, b_k\}$$

donde b_1 es un prototipo base aleatoriamente seleccionado

$$b_i = \text{argmax}_{p \in P - B_i} (\min_{k=1}^{i-1} d(p, b_k)) \;\; i = 2, 3 \ldots n \tag{3.3}$$

Los métodos SDM y MDM utilizan, como se puede observar, una estrategia voraz para la selección de PB y por tanto, subóptima (no se garantiza la máxima separación). Sin embargo, ambas técnicas requieren un procesamiento temporal lineal respecto al tamaño del conjunto de prototipos que es menor que en la técnica de las c-medias.

3.3. Selección por error mínimo de la distancia

Este método ha sido propuesto por Ramasubramanian para intentar encontrar el *mejor* conjunto de PB [Ram, 92]. El problema se puede enunciar de la siguiente forma: dado un conjunto $P = \{p_1, ..., p_m\}$ de prototipos, se trata de obtener el conjunto de prototipos base $B^* = \{b_1, ..., b_n\} \subseteq P$, que minimice la diferencia entre el valor verdadero de la distancia entre todos los prototipos del conjunto P y el valor de la función cota inferior de la distancia

para un conjunto de PB obtenido incrementalmente. De forma resumida, el conjunto que se quiere obtener es aquel que cumple:

$$B^* = \text{argmin}_{B \subseteq P} \left(\sum_{p,q \in P} |d(p,q) - max_{b \in B}(|d(p,b) - d(q,b)|)| \right) \tag{3.4}$$

El procedimiento tiene un bucle principal en el que en cada iteración selecciona un nuevo prototipo base a añadir al conjunto B. El nuevo prototipo base seleccionado es tal que hace que el error cometido entre la distancia real entre cada dos prototipos del conjunto P y la dada por la función cota inferior de esa distancia es mínimo. Una vez realizada esta selección, se actualiza la cota inferior de la distancia de todos los prototipos usando este nuevo prototipo base (ver figura 3.1). El coste de este algoritmo es $|B| \times |P|^3$.

3.4. Resumen comparativo de costes

En la tabla 3.1 se hace un resumen de los costes temporales y espaciales de los métodos de selección de PB mencionados. Estos costes vienen expresados en función del número de prototipos, $|P|$, del conjunto P y del número de PB, $|B|$. En estas tablas no se considera el coste espacial de la matriz de distancias que se calcula en cada uno de los métodos por ser equivalente en todos ellos. Esta matriz de distancias tiene un coste $O(|P| \times |B|)$, valor que debe sumarse al coste espacial aparecido en la tabla.

	COMPLEJIDAD	
MÉTODO	TEMPORAL	ESPACIAL
aleatorio	O($\|B\|$)	O(1)
c-medias	O($\|B\| \times \|P\|^2$)	O($\|P\|$)
suma máxima de distancias (SMD)	O($\|B\| \times \|P\|$)	O($\|P\|$)
máximo de distancias mínimas (MDM)	O($\|B\| \times \|P\|$)	O($\|P\|$)
error mínimo de la distancia (EMD)	O($\|B\| \times \|P\|^3$)	O($\|P\|^2$)

Cuadro 3.1: Costes de los diferentes métodos de selección de PB estudiados.

Como se puede observar en la tabla 3.1, existen diferencias sustanciales en el coste de los métodos. Desde el método aleatorio que tiene un coste constante respecto a $|P|$ al más caro de todos, el método EMD con un coste cúbico respecto a $|P|$. Sería muy interesante que las técnicas con coste

Algoritmo Error mínimo de la distancia (EMD)
Entrada:
$P \subset E$ *// conjunto finito de prototipos*
$n \in \mathbb{N}$; *// número de prototipos base*
Salida:
$B \subseteq \mathrm{P}$; *// conjunto de prototipos base*
Función: $d : E \times E \to \mathbb{R}$ *// función distancia*
Variables:
$g \in \mathbb{R}^{|P|^2}$; *// matriz de cotas inferiores*
$e, e^*, \triangle \in \mathbb{R}$;
$s, p, p*, q \in P$;
Método
para todo $p, q \in P$ **hacer**
$g[p, q] = 0$;
fin para todo
repetir
$e^* = \infty$;
para todo $s \in P - B$ **hacer**
$\mathrm{e} = 0$;
para todo $p, q \in P$ **hacer**
$\triangle = d(p, q) - \max(g(p, q), |d(p, s) - d(q, s)|)$;
$e = e + \triangle$;
fin para todo
si $e < e^*$ **entonces**
$p^* = s; e^* = e$;
fin si
fin para todo
$B = B \cup \{p^*\}$;
para todo $p, q \in P$ **hacer**
$g[p, q] = \max(g[p, q], |d(p, p^*) - d(q, p^*)|)$;
fin para todo
hasta $|B| \leq n$
fin Método
fin Algoritmo

Figura 3.1: Algoritmo *error mínimo de la distancia*

temporal constante o lineal condujeran a un buen rendimiento del algoritmo de búsqueda pues, en otro caso, el coste de preproceso podría resultar prohibitivo.

3.5. Experimentos

Los resultados que se presentan en este apartado se han realizado utilizando varias métricas para distintos tamaños del conjunto de prototipos uniformemente distribuidos en el hipercubo unidad.

Las métricas utilizadas pertenecen a la familia de métricas L_p, más conocidas como métricas de Minkowski:

$$L_p(x,y) = (\sum_{k=1}^{K} \mid x_k - y_k \mid^p)^{1/p} \tag{3.5}$$

La métrica L_∞ o *métrica máxima* se define como

$$\lim_{p\to\infty} L_p(x,y)$$

pudiendo calcularse como

$$L_\infty(x,y) = \max_{k=1..K} \mid x_k - y_k \mid \tag{3.6}$$

De esta familia se ha utilizado L_1 (*métrica mínima*), L_2 (*métrica euclídea*) y L_∞ (*métrica máxima*).

3.5.1. Experimentos sobre la selección del conjunto de prototipos base

En la figura 2.7 del capítulo anterior, se ha podido observar que existe un número de PB, relativamente pequeño comparado con el tamaño del conjunto de prototipos P, para el cual el número de distancias calculadas es mínimo. Si aquellos resultados han sido obtenidos eligiendo de forma aleatoria los PB, cabe pensar que una elección mejor de los elementos de este conjunto llevará a resultados mejores. Por ello, se han estudiado diversas formas de selección de los mismos. En las secciones anteriores se han presentado algunos métodos que van a servir para la selección de PB, unos ya conocidos, otros propuestos en este trabajo. Estos métodos serán estudiados experimentalmente utilizando las mismas condiciones de trabajo (tamaño de los conjuntos de prototipos, mismos conjuntos de prototipos y muestras, tipo de distancia aplicada, etc), y así poder comparar posteriormente los resultados.

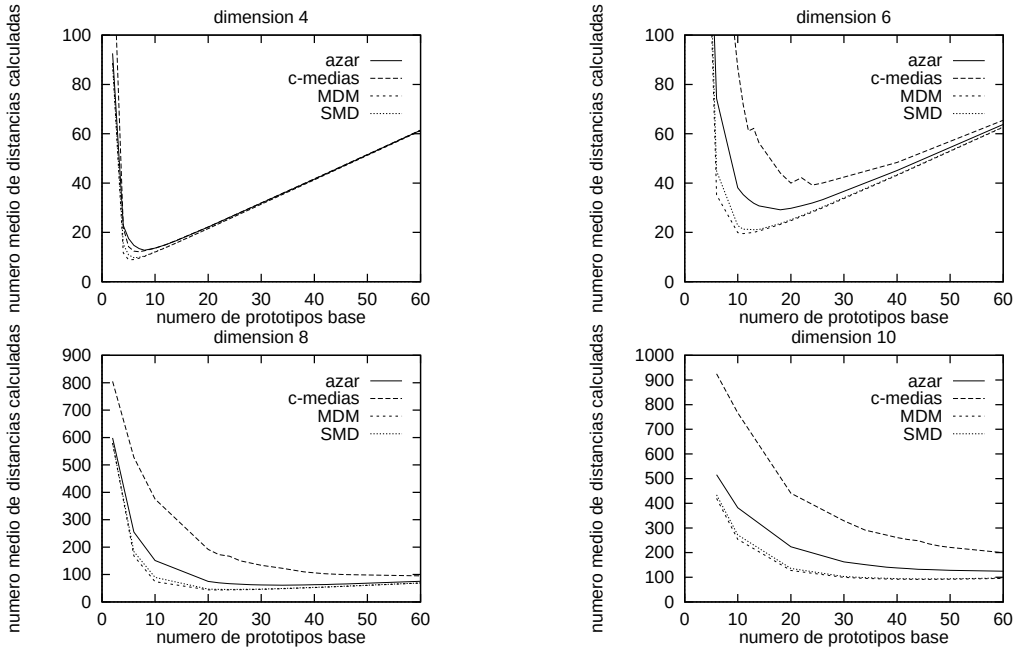

Figura 3.2: Número medio de distancias calculadas por el algoritmo, para los diferentes métodos de selección de PB para la métrica euclídea, al variar el tamaño del conjunto de PB. El conjunto de prototipos utilizado, de tamaño 1024, presenta una distribución uniforme. La desviación típica de todas las estimaciones es del orden del 3 %.

El primer experimento realizado es la repetición del de la figura 2.7 utilizando los métodos de selección de PB mencionados en este capítulo. Los resultados obtenidos para varias dimensiones utilizando la distancia euclídea se pueden ver en la figura 3.2.

Todos los métodos se comportan de forma similar al de la selección aleatoria, en el sentido de que existe un valor para el tamaño del conjunto de PB para el cual el número de distancias calculadas es mínimo. Esto es de esperar en parte debido a que el algoritmo calcula todas las distancias de la muestra a los PB, y a medida que aumenta el número de PB aumenta el número de distancias calculadas. Sin embargo, se puede ver en la figura que las diferencias mayores entre los distintos métodos se encuentran precisamente en la zona para la que el número de distancias es mínimo. Curiosamente, además, a medida que aumenta la dimensión, la selección aleatoria es mejor que aquella que utiliza los centroides obtenidos por el algoritmo de las c-medias como prototipos base. En esta figura no se han presentado resultados obtenidos con el criterio denominado *error mínimo de la distancia* (EMD) porque su elevado coste computacional hacía inviable realizar todos los expe-

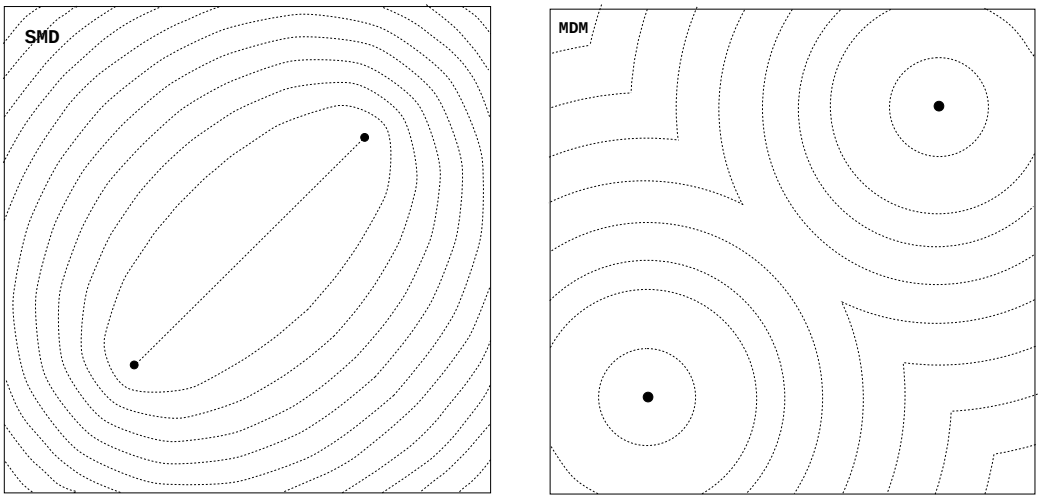

Figura 3.3: Selección de un tercer prototipo base según los métodos SMD y MDM para la métrica euclídea.

rimentos, aunque sí se ha realizado un pequeño experimento cuyos resultados se muestran en la tabla 3.2 que será comentada un poco más adelante.

En la figura 3.3 se ha representado gráficamente el comportamiento de algunos de estos métodos para la métrica euclídea. Concretamente, los métodos de selección de PB que se han representado son los denominados *suma máxima de distancias*, SMD, y *máximo de las distancias mínimas*, MDM. Las líneas que aparecen en cada uno de los dibujos representan los prototipos para los que la suma máxima de distancias (SMD) ó el máximo de la distancias mínimas (MDM) a dos prototipos previamente seleccionados es la misma.

Para comprobar el comportamiento del criterio de selección de PB propuesto por Ramasubramanian se ha realizado un pequeño experimento utilizando la distancia euclídea cuyos resultados se pueden ver en la tabla 3.2. En esta tabla, se ha añadido además los resultados de los otros métodos para los mismos parámetros.

Los mejores resultados se obtienen utilizando los criterios EMD, MDM y SMD (son similares teniendo en cuenta el error de la media). Pero ¿cómo se eligen verdaderamente los PB con el método EMD? Para comprobarlo, se ha representado un ejemplo pequeño con 128 prototipos pertenecientes a una distribución uniforme utilizando la métrica euclídea, donde se puede ver el orden en el que se seleccionan los PB con los métodos EMD, MDM y SMD (ver figura 3.4). Los prototipos seleccionados en cada caso son prácticamente los mismos, aunque cambia ligeramente el orden en que se van eligiendo. La

número de PB	EMD	MDM	SMD
5	8.56 ± 0.15	8.88 ± 0.51	10.89 ± 0.98
6	9.07 ± 0.13	8.97 ± 0.11	9.73 ± 0.28
7	9.68 ± 0.10	9.54 ± 0.07	9.91 ± 0.24
8	10.40 ± 0.09	10.26 ± 0.07	10.44 ± 0.12
10	12.04 ± 0.04	11.99 ± 0.03	12.10 ± 0.08

número de PB	c-medias	azar
5	14.22 ± 1.58	17.54 ± 2.21
6	12.40 ± 0.90	14.77 ± 1.41
7	12.14 ± 0.22	13.42 ± 1.41
8	12.56 ± 0.35	12.77 ± 0.64
10	13.56 ± 0.22	13.65 ± 0.37

Cuadro 3.2: Número medio de distancias calculadas por los diferentes métodos de selección de PB para un conjunto de 1024 prototipos extraídos de un espacio vectorial euclídeo de dimensión 4.

característica común en todos ellos es que los prototipos seleccionados se encuentran muy cercanos a la frontera del hipercubo donde estan representados y, a la vez, muy separados unos de otros. El prototipo `PB1`, que no aparece en la frontera para los métodos SMD y MDM, se encuentra en esta posición porque es el único que ha sido elegido aleatoriamente en estos métodos.

Comparando las tres selecciones se puede ver que los tres métodos prácticamente tienen un comportamiento similar. Por lo tanto, se puede asegurar que los métodos SMD y MDM seleccionan los PB de forma que el error cometido al utilizar la función cota inferior de la distancia en lugar de la propia distancia, si no es mínimo se acerca bastante al mismo. Si para estos mismos ejemplos se aplica la regla de eliminación para una muestra x y un vecino más próximo a la misma, n, después de calcular la distancia a x de los tres primeros PB seleccionados, se puede ver en la figura 3.5 que los mejores resultados (mayor cantidad de prototipos eliminados) se presentan cuando los PB se encuentran muy separados unos de otros, debido a que el volumen determinado por la intersección de los hiperanillos que producen la regla de eliminación es menor.

Observando la tabla 3.1, donde aparecen los costes temporales de los algoritmos, se puede ver que los métodos SMD y MDM son lineales con el tamaño del conjunto de prototipos, mientras que el método MED es cúbico. Debido a esto, y al coste espacial que conlleva se descartará este método, y de ahora en adelante se seleccionarán los PB utilizando únicamente los métodos

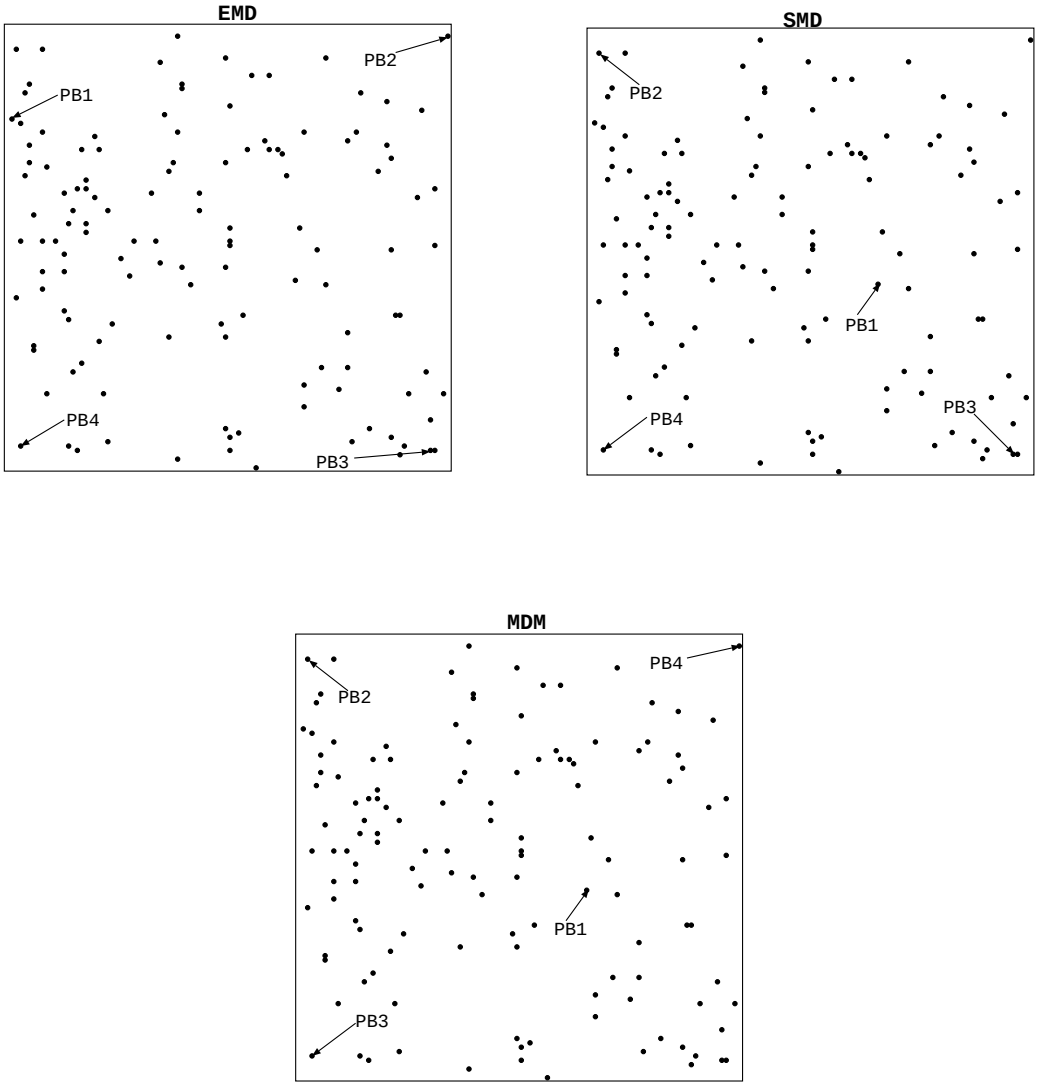

Figura 3.4: Orden en el cual los métodos EMD, SMD y MDM han seleccionado los cuatro primeros PB.

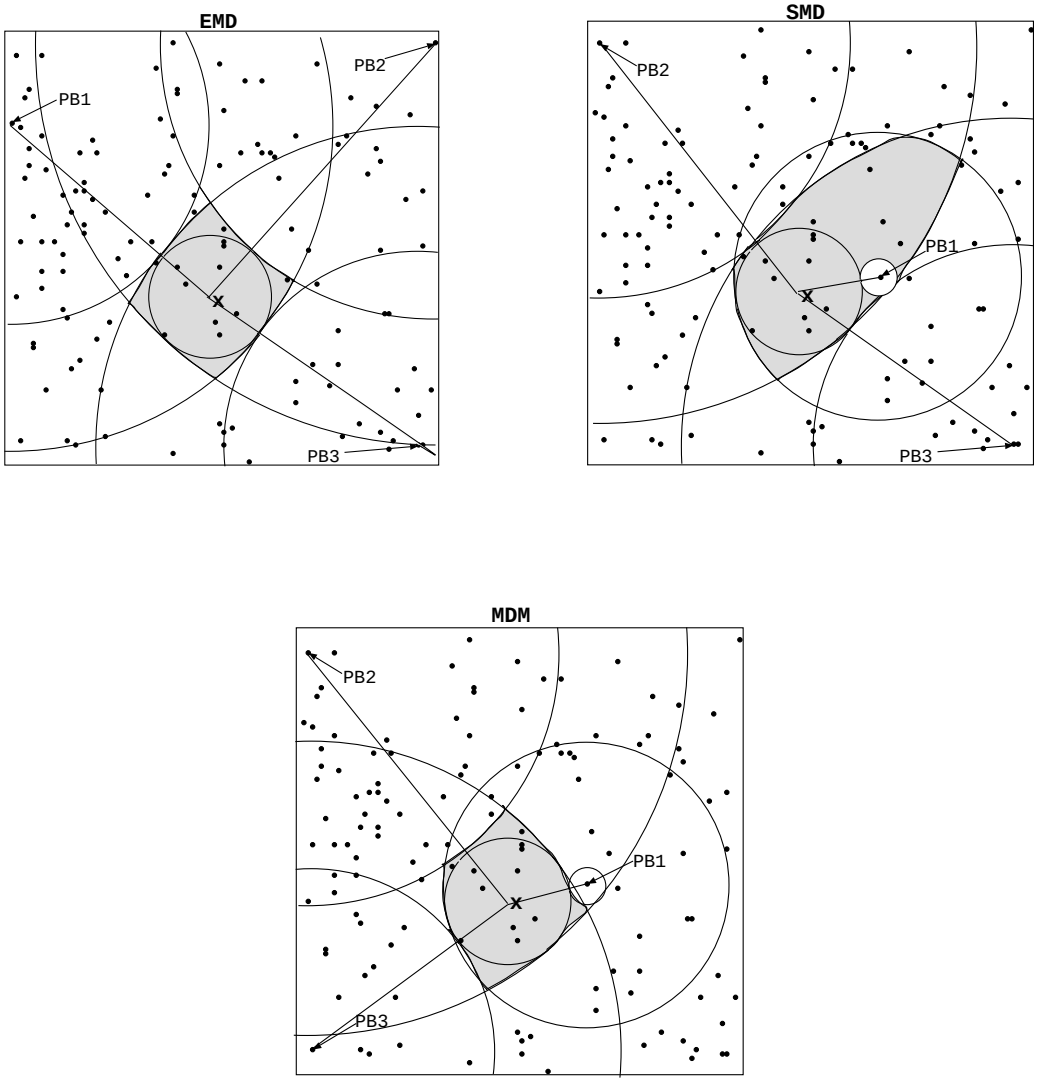

Figura 3.5: Orden en el cual los métodos EMD, SMD y MDM han seleccionado los tres primeros PB usando la métrica euclídea. La zona sombreada indica el espacio de búsqueda que queda después de haber aplicado la regla de eliminación tres veces (para los tres PB seleccionados).

SMD y MDM.

3.5.2. Experimentos para obtener el tamaño del conjunto de prototipos base

En la figura 3.6 se estudia el número de distancias calculadas para diferentes dimensiones (4, 6, 8, 10) y un tamaño del conjunto de prototipos $|P| = 1024$, variando el número de PB.

Para cada valor de la dimensión del espacio se obtiene un valor mínimo del número medio de distancias calculadas asociado a un tamaño $|B|$ determinado del conjunto de PB.

Para comprobar que la obtención de los mínimos óptimos en cada dimensión no dependen del tamaño del conjunto de prototipos, se repitió el experimento anterior variando este tamaño. En la figura 3.7 se han representado los valores de $|B|$ óptimos, $|B|^*$, obtenidos al variar el tamaño del conjunto de prototipos para los métodos SMD y MDM. Como se puede observar, la obtención de los $|B|^*$ para cada dimensión es prácticamente independiente del tamaño del conjunto de prototipos.

Este valor de $|B|$ óptimo aumenta con la dimensión del espacio de forma aparentemente exponencial según se pone de manifiesto explícitamente en la figura 3.8. Las diferencias entre uno y otro método son mínimas, como puede observarse en la tabla 3.3.

Este comportamiento se produce porque se trabaja con distribuciones uniformes de prototipos y a medida que aumenta la dimensión del espacio, aumenta exponencialmente la distancia media entre las muestras y sus prototipos más cercanos. Cabe esperar que esto no ocurra cuando estas distancias estén acotadas de alguna forma; por ejemplo, cuando se trabaja con muestras que presenten cierto grado de agrupamiento alrededor de los prototipos (esto se discutirá en el capítulo 5).

En la figura 3.9 se ha representado, para los valores óptimos de $|B|$ obtenidos para 8192 prototipos, el número medio de distancias calculadas en función del tamaño del conjunto de prototipos. Los resultados han mejorado respecto a los obtenidos en el capítulo anterior y se observa, además, que el método MDM produce, para los valores $|B|^*$ anteriores, resultados mejores que el método SMD en dimensiones altas.

3.5.3. Experimentos con distintas métricas

Los experimentos discutidos en la sección anterior han sido repetidos con otros tipos de métrica, concretamente se ha utilizado, dentro de la familia de métricas L_p, la mínima (L_1) y la máxima (L_∞).

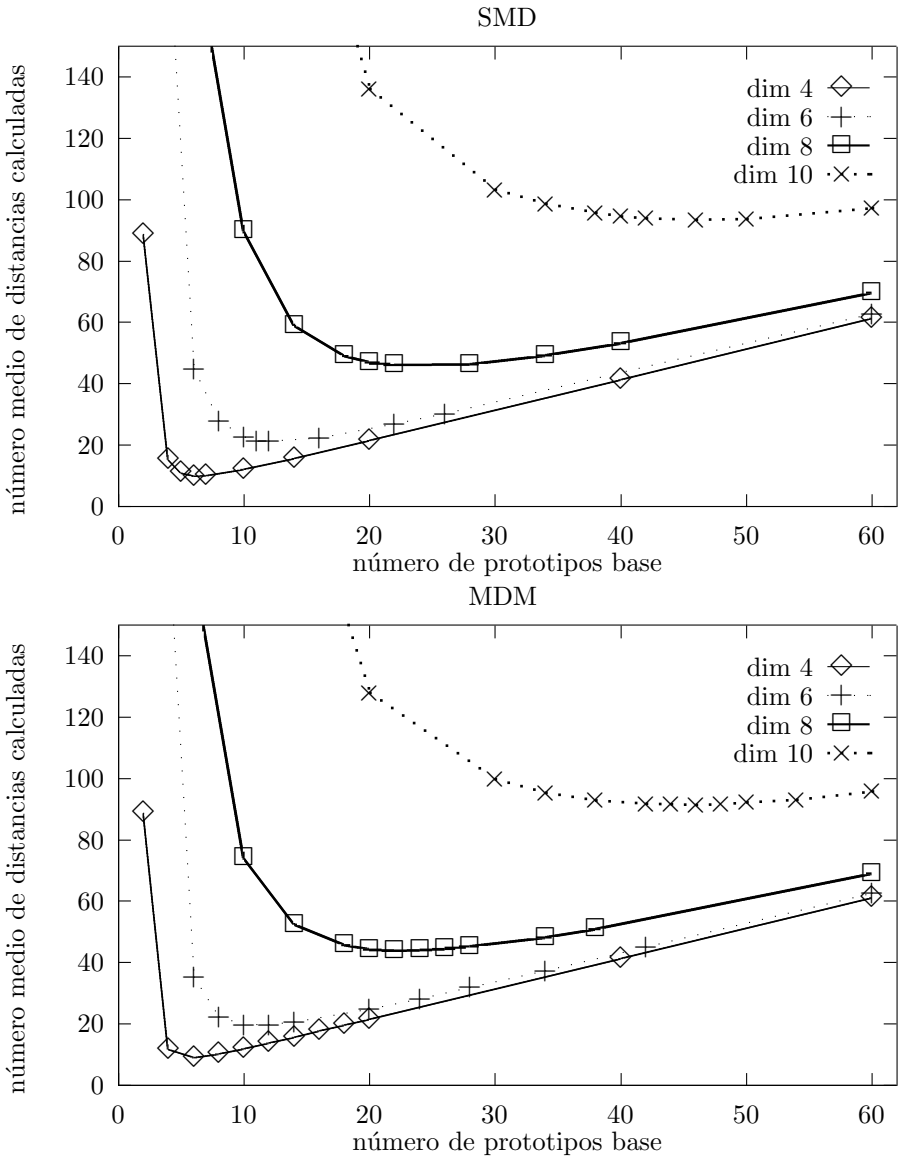

Figura 3.6: Variación del número medio de distancias calculadas por el algoritmo en un espacio vectorial euclídeo de n dimensiones en función de $|B|$, usando una distribución uniforme para un conjunto de 1024 prototipos con los criterios de selección de prototipos base SMD y MDM. La desviación típica de todas las estimaciones es del orden del 3 % en los valores para el tamaño del conjunto de prototipos en los que el número medio de distancias es mínimo. Este valor de la desviación disminuye a medida que aumenta el número de prototipos base utilizado.

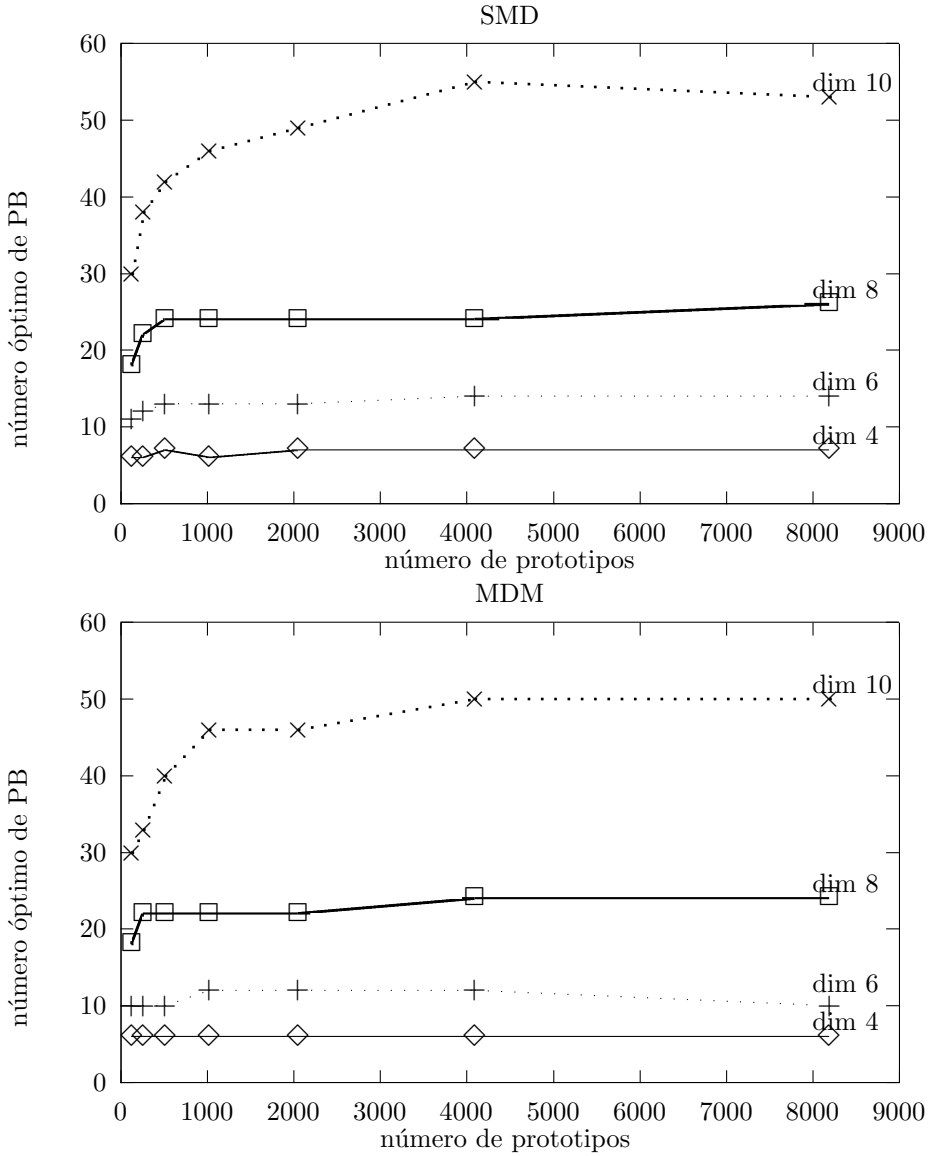

Figura 3.7: Número óptimo de PB ($|B|^*$) para el algoritmo, en función del tamaño del conjunto de prototipos $|P|$ y diferentes dimensiones usando la métrica euclídea para los métodos de selección de prototipos base SMD y MDM.

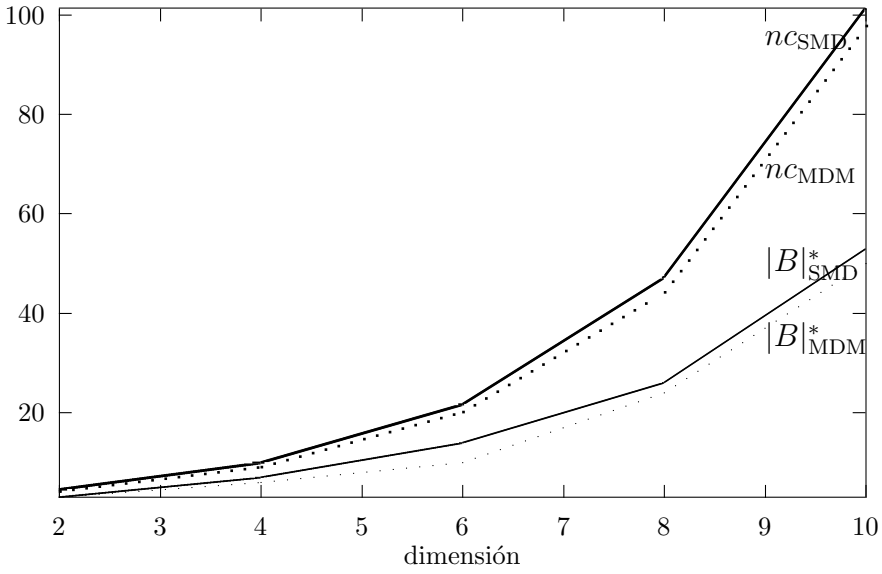

Figura 3.8: Número óptimo de PB ($|B|^*$) en función de la dimensión y número medio de distancias calculadas (nc) para ese valor, usando la métrica euclídea para una distribución uniforme de un conjunto de 8192 prototipos.

	$\|B\|^*$			
	dim 4	dim 6	dim 8	dim 10
SMD	7	14	26	53
MDM	6	10	24	50

Cuadro 3.3: Tamaños óptimos del conjunto de PB para los métodos de selección de prototipos base SMD y MDM.

Métrica L_1

En la figura 3.10 se ha repetido el experimento de la figura 3.2, es decir, se ha realizado el estudio de la efectividad de los diferentes métodos de selección de PB para la métrica L_1. Comparando los resultados de ambas métricas, se ve que en general las conclusiones a las que se llega para la métrica L_1 son las mismas que para la métrica L_2. En casi todas las dimensiones los mejores resultados se obtienen utilizando los métodos de selección SMD y MDM, aunque son un poco mejores en el caso del MDM. Además, el número medio de distancias calculadas en este caso es menor que el obtenido con la métrica euclídea.

Al igual que con la distancia euclídea, el tamaño óptimo ($|B|^*$) de PB

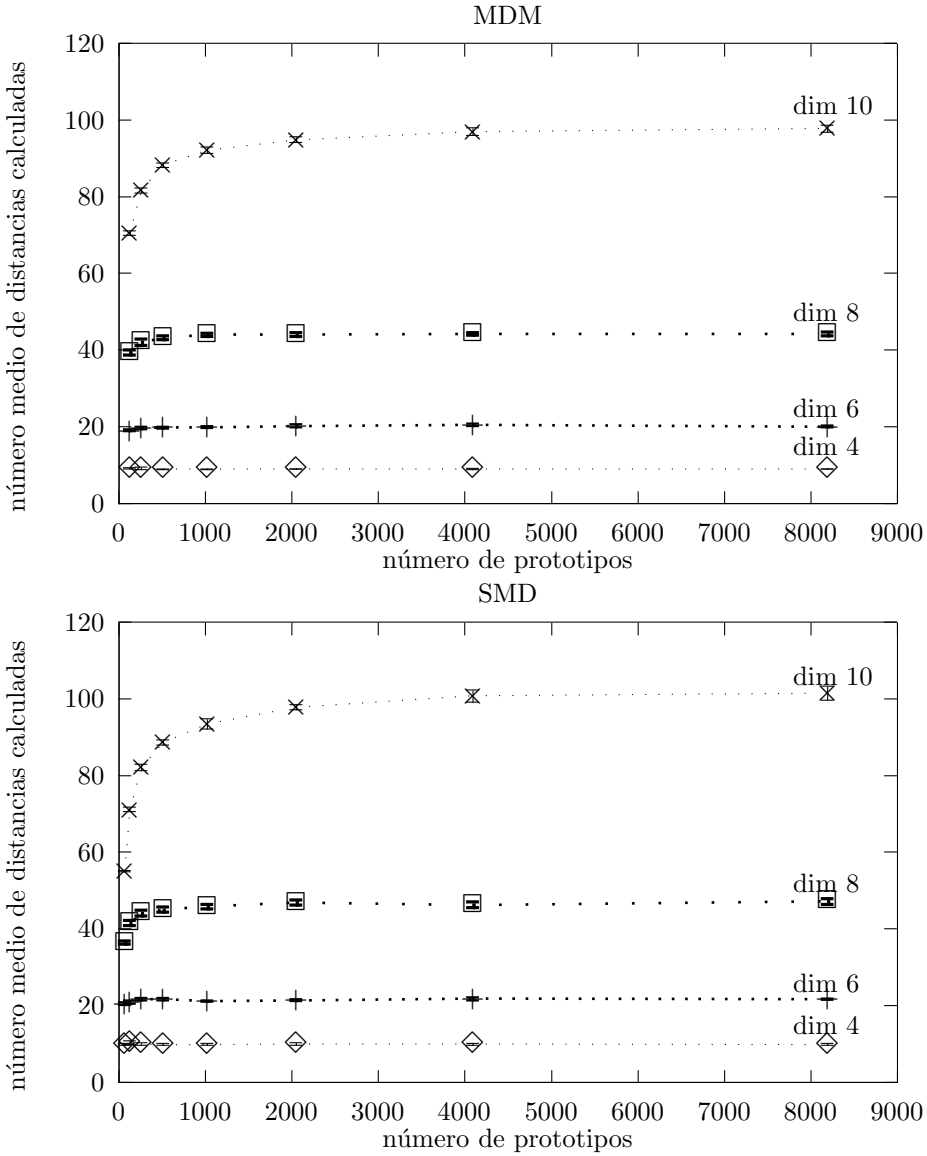

Figura 3.9: Número medio de distancias calculadas en función del número de prototipos $|P|$ y diferentes dimensiones, usando la métrica euclídea para los métodos de selección de prototipos base SMD y MDM. El número de PB utilizados en cada método es el que aparece en la tabla 3.3.

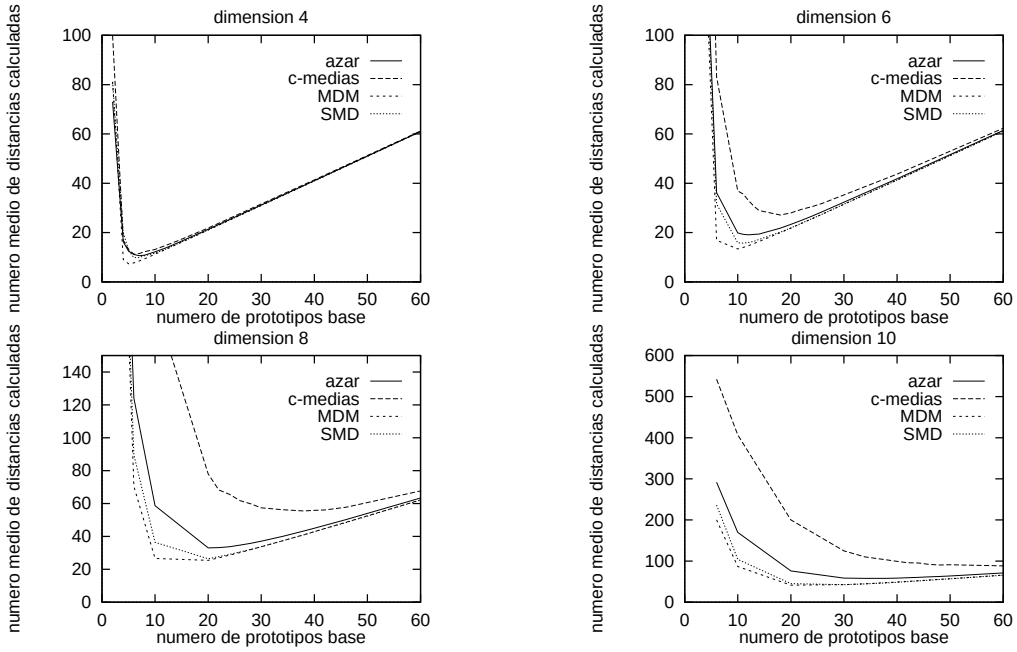

Figura 3.10: Número medio de distancias calculadas por el algoritmo, para los diferentes métodos de selección de PB para la métrica L_1 al variar el tamaño del conjunto de PB. El conjunto de prototipos utilizado, de tamaño 1024, presenta una distribución uniforme. La desviación típica de todas las estimaciones es del orden del 3 %.

es independiente del tamaño del conjunto de prototipos sobre el que ha sido seleccionado, con la única salvedad de que ese valor óptimo difiere para los dos métodos utilizados para cada dimensión. Los resultados para 8192 prototipos se pueden ver en la tabla 3.4.

	$\|B\|^*$			
	dim 4	dim 6	dim 8	dim 10
SMD	10	12	18	30
MDM	6	8	14	20

Cuadro 3.4: Número óptimo de PB para los métodos de selección de prototipos base SMD y MDM obtenidos a partir de un conjunto de 8192 prototipos, usando la métrica L_1.

En la figura 3.11 se representa el número medio de distancias calculadas para los valores óptimos del número de PB obtenidos en cada dimensión, para los métodos de selección de prototipos base SMD y MDM. Como en el caso de la distancia euclídea, la selección realizada con MDM da mejores resultados en cuanto al número de distancias calculadas.

Métrica L_∞

Los resultados de la comparación de los diferentes métodos de selección de PB para la métrica L_∞ se pueden ver en la figura 3.12. Se observa el mismo comportamiento que con las otras métricas.

	$\|B\|^*$			
	dim 4	dim 6	dim 8	dim 10
SMD	8	18	34	78
MDM	8	18	36	82

Cuadro 3.5: Número óptimo $|B|^*$ para los métodos de selección de prototipos base SMD y MDM para un conjunto de 8192 prototipos usando la métrica L_∞.

En la figura 3.13 se puede ver el número medio de distancias calculadas para los valores óptimos del número de PB obtenidos en cada dimensión, en los diferentes métodos de selección.

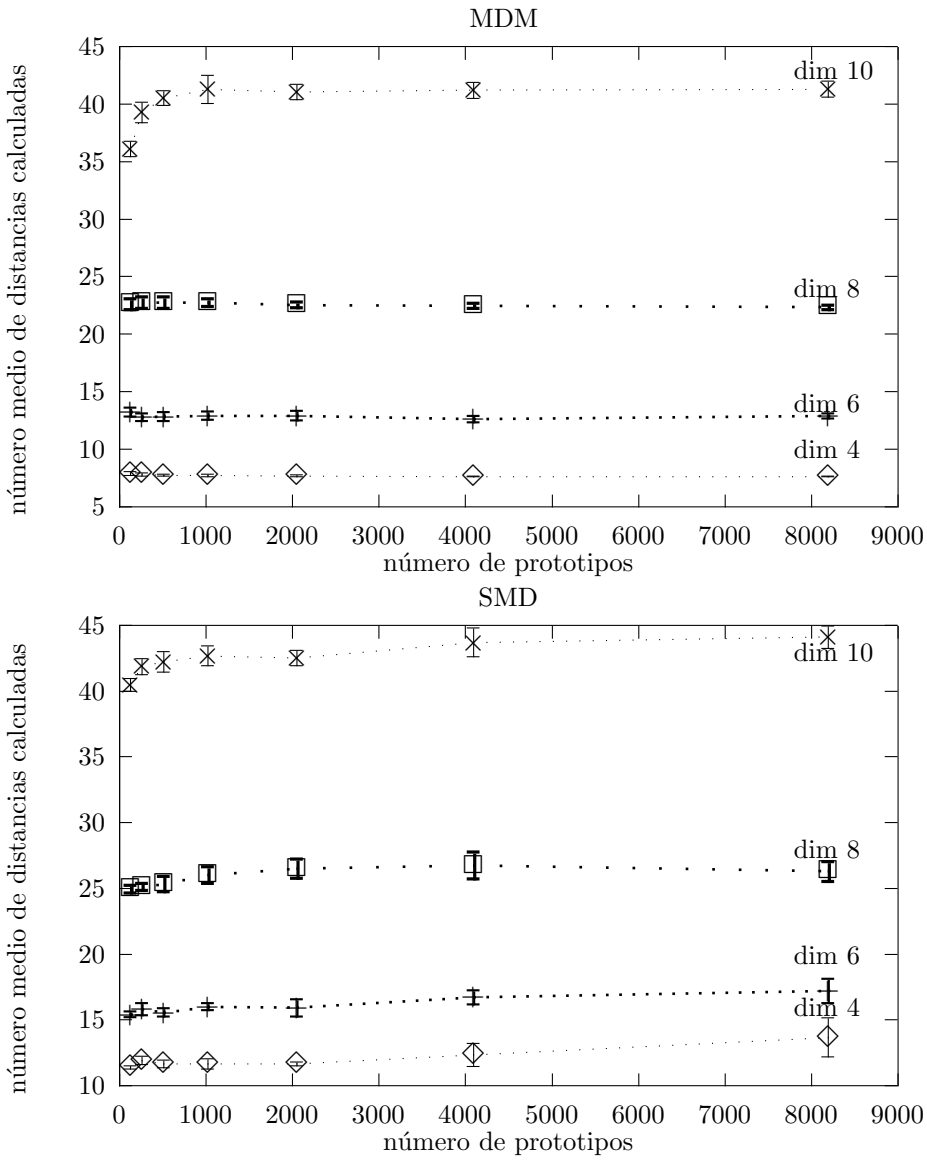

Figura 3.11: Número medio de distancias calculadas (nc) en función del número de prototipos ($|P|$) y diferentes dimensiones, usando la métrica L_1 para los métodos de selección de prototipos base SMD y MDM.

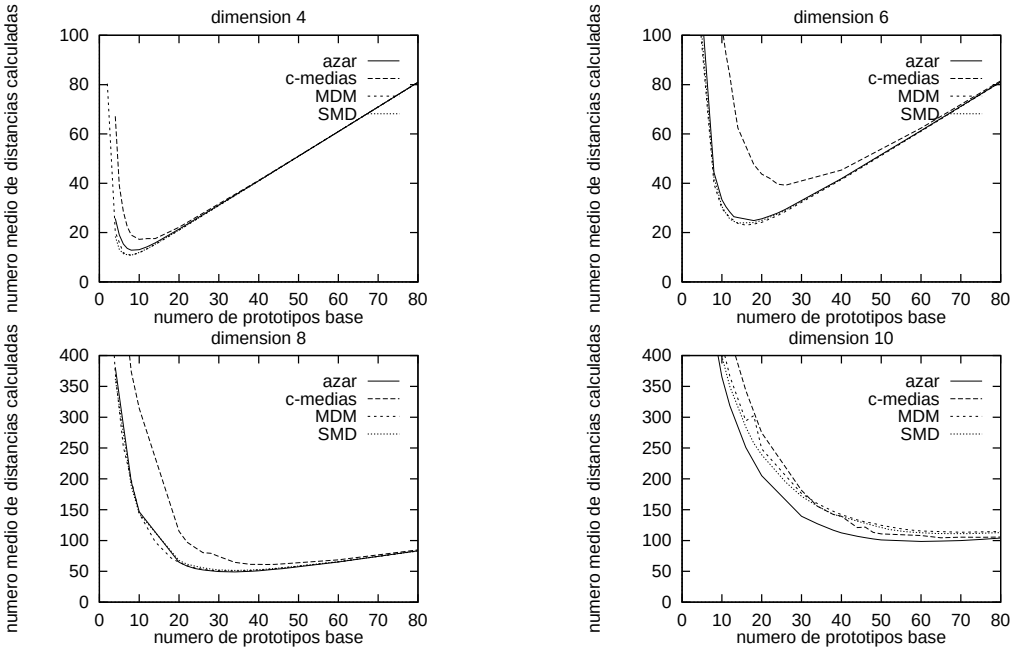

Figura 3.12: Número medio de distancias calculadas para la métrica L_∞ por el algoritmo, para los diferentes métodos de selección de PB al variar el tamaño del conjunto de PB. El conjunto de prototipos utilizado, de tamaño 1024, presenta una distribución uniforme. La desviación típica de todas las estimaciones es del orden del 3 %.

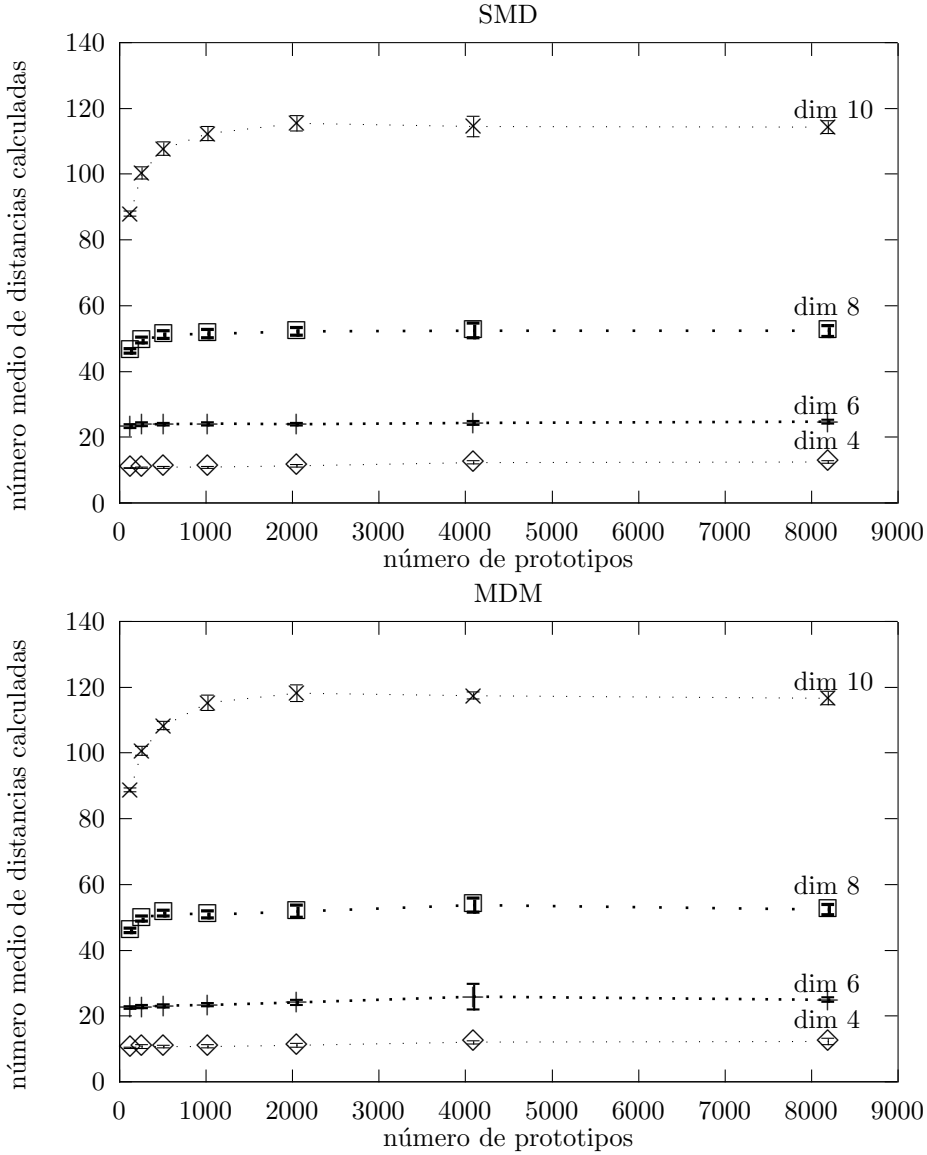

Figura 3.13: Número medio de distancias calculadas para la métrica L_∞, usando como tamaño del conjunto de PB el valor óptimo, $|B|^*$, que aparece en la tabla 3.5.

3.6. Conclusiones

En este capítulo se ha realizado un estudio sobre la obtención de los PB. Se ha podido ver que tan importante es la selección de los mismos como el tamaño de este conjunto. De los diversos métodos de selección de PB, se concluye que los métodos de selección de PB más rápidos y más eficientes son SMD y MDM, aunque MDM ha sido sistemáticamente un poco mejor que SMD. Ambos métodos seleccionan los PB de forma que éstos se encuentran muy alejados unos de otros y por tanto muy cerca de las fronteras del hipercubo que representa el conjunto de datos. De hecho, se ha comprobado gráficamente que el volumen de la intersección de los hiperanillos que se obtiene por aplicación de la fase de eliminación es menor cuando los PB se encuentran muy alejados unos de otros. En cuanto a la elección del tamaño del conjunto de PB, se ha visto que existe un valor $|B|^*$ óptimo para cada dimensión del espacio para el cual el número de distancias calculadas es mínimo en promedio. A partir de este tamaño óptimo el número de distancias calculadas aumenta con $|B|$, de ahí el interés que va a tener una buena determinación del mismo. Estas conclusiones se mantienen al cambiar la métrica con la que se trabaja. La única diferencia que se observa es que para L_1 el número de distancias calculadas siempre era menor que con la distancia euclídea y mayor en el caso de L_∞. Estos resultados coinciden con los obtenidos por Ramasubramanian en su tesis doctoral [Ram, 91].

Capítulo 4

Gestión de los prototipos base

Tomando como base el algoritmo LAESA propuesto en el capítulo 2, se presenta aquí una serie de modificaciones al mismo que mejoran los resultados presentados hasta el momento. Estas modificaciones están relacionadas básicamente con la manipulación de los PB en las distintas fases del algoritmo. En la figura 2.7 del capítulo anterior se puede ver que la elección del tamaño del conjunto de PB es muy crítica, ya que a partir del valor óptimo, para el cual el número de distancias calculadas es menor, este número de distancias crece linealmente con el tamaño del conjunto de PB.

La primera modificación hace referencia a la posibilidad de evitar el cálculo de distancias de los PB a la muestra cuando se cumplen ciertas condiciones. En principio, podría pensarse que realizar eliminaciones de los PB en las primeras etapas son desfavorables, pues no se aprovecharía suficientemente la información contenida en los mismos en las etapas posteriores. No obstante, para cada configuración del conjunto de prototipos P y para cada muestra x, la utilidad de los diferentes PB puede ser diversa y se podrían admitir eliminaciones "tempranas"de PB si, de hecho, los prototipos eliminados no aportan nada a la actualización de la cota inferior de la distancia del resto de prototipos.

La segunda modificación se refiere a la selección de PB en la *fase de aproximación*, que hasta ahora era aleatoria. Concretamente, se puede mejorar esta fase, sin aumentar por ello el coste, realizando la *aproximación-selección* de PB de la misma forma que se efectúa para los prototipos que no son base, es decir, seleccionar en cada iteración el PB candidato a ser el más cercano a la muestra hasta el momento.

En el capítulo anterior se ha realizado un estudio del número óptimo $|B|^*$ de PB y cómo estos son seleccionados en el conjunto de prototipos. Esta selección se lleva a cabo en el preproceso, luego es independiente del conjunto de muestras sobre los que se van a aplicar posteriormente. En este capítulo se

estudiará cómo gestionar los PB en las fases de aproximación y eliminación.

4.1. Criterios de eliminación de prototipos base

En la fase de eliminación del algoritmo propuesto en el capítulo 2 no se permitía eliminar nunca PB (solo se eliminaban prototipos que no eran base). Debido a esto, era muy importante la selección de un valor óptimo ($|B|^*$) para el tamaño de este conjunto ya que el número de distancias calculadas aumentaba linealmente con dicho tamaño. La desventaja que tiene este criterio es que la eficiencia del algoritmo (medida en función del número de distancias calculadas) está directamente relacionada con la elección del valor de $|B|$ (ver figura 2.5), además de cual sea el método concreto de elección de los PB.

Se va a añadir la fase de eliminación de PB en los algoritmos presentados en la sección 2.2.4, de forma que cualquier PB podrá ser eliminado cuando se cumplan dos condiciones: 1) la condición de eliminación presentada para los prototipos que no son base ($g[b] \geq D_{\min}$, siendo b un PB), y 2) sea cierta una función booleana denominada CONDICIÓN (en la sección 5.4 se presenta el algoritmo LAESA con la incorporación de esta función). Se pretende que con la introducción de esta modificación la eficiencia del algoritmo no dependa tanto del valor elegido para $|B|$. Concretamente se presenta una familia de funciones que se apoyan, en general, en la idea de permitir descartar PB en la fase de eliminación [MOV, 94]. Las funciones booleanas son muy sencillas y se basan en la siguiente idea: sea n_b el número de PB seleccionados hasta la iteración actual del algoritmo; la eliminación de PB es permitida cuando este valor es mayor que $|B|/k$, siendo $|B|$ el número total de PB y k una constante. Esto es,

$$\text{CONDICIÓN} = (n_b > |B|/k). \tag{4.1}$$

Para valores de $k = 1, 2, \ldots \infty$ esta función da lugar a una serie de criterios de gestión de los PB referenciados como E_1, E_2, ... E_∞.

A medida que aumenta el valor de k, se permite adelantar el proceso de eliminación de PB en el algoritmo. Los dos casos extremos, con comportamiento por tanto bien diferente, son:

- $k = 1$ (E_1)
  Es el criterio que se ha utilizado hasta ahora, es decir, nunca se permite

eliminar PB. Esta condición se podría haber expresado también como CONDICION = falso

- $k = \infty$ (E_∞)
  En este caso desde el comienzo del proceso de búsqueda se permite eliminar PB si cumplen la regla de eliminación ya formulada. Esta condición se podría haber expresado también como CONDICION = verdadero

Esta familia de funciones booleanas solamente tiene en cuenta el tamaño del conjunto de PB.

Adicionalmente, se ha estudiado otro criterio independiente de la familia $E_1 \ldots E_\infty$ que utiliza información sobre la influencia que puedan tener los PB en el proceso de búsqueda. Concretamente, se va a permitir la eliminación de PB cuando estos son de poca utilidad en la fase de eliminación ("no sirven"para eliminar otros prototipos) [MOV, 92]. Este criterio no tendría mucho sentido utilizando la aproximación de PB que se efectúa en cada iteración del algoritmo propuesto en el capítulo 2 ya que en ésta los PB eran elegidos arbitrariamente. Esto lleva a que la aplicación de este criterio de eliminación concreto sobre PB arbitrariamente seleccionados no tenga ningún sentido.

Este nuevo criterio, al que se ha llamado E_{elim}, se define de la siguiente forma:

$$\text{CONDICIÓN} = \text{ningún prototipo de } P \text{ ha sido eliminado en el paso anterior} \tag{4.2}$$

La introducción de cualquiera de estas nuevas condiciones de eliminación de los PB produce ligeras modificaciones en la estructura del algoritmo original. Concretamente, ahora un prototipo base podrá ser eliminado si se cumple

$$\text{CONDICION y } g[b] \geq D_{\min} \tag{4.3}$$

mientras que la condición para eliminar prototipos que no es base continúa siendo solamente

$$g[b] \geq D_{\min} \tag{4.4}$$

4.1.1. Experimentos

El primer experimento fue realizado para comparar el comportamiento del algoritmo con las diferentes estrategias de eliminación de PB basadas en la

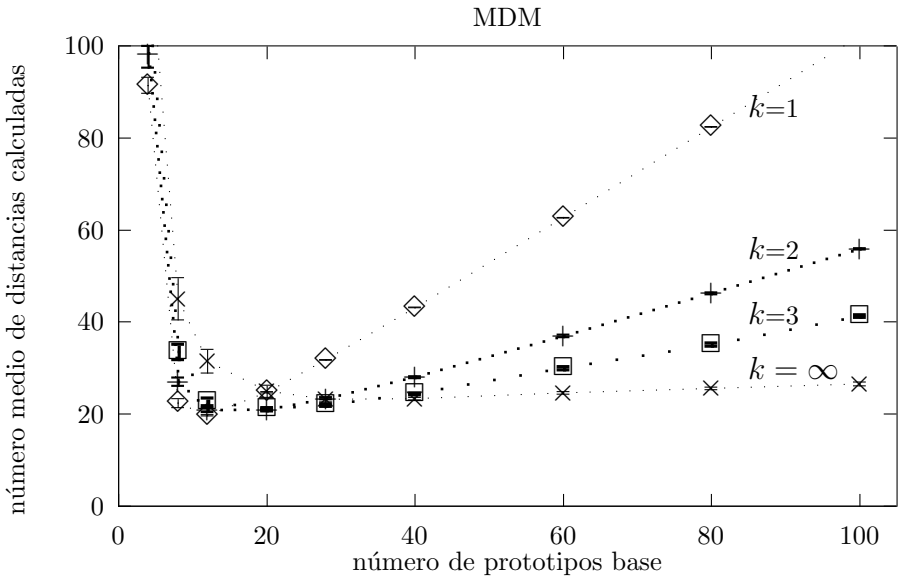

Figura 4.1: Número medio de distancias calculadas por el algoritmo, al variar el tamaño del conjunto de PB, en un espacio de dimensión 6, usando las estrategias de eliminación E_k y el método de selección de prototipos base MDM.

familia denominada E_k. El experimento se llevó a cabo utilizando un conjunto de 1024 prototipos y la métrica euclídea para dimensión 6. El número de PB, $|B|$, variaba en un rango de 2 a 100. Los resultados para el criterio de selección de prototipos base MDM (los de SMD son muy parecidos a los presentados para MDM) se pueden ver en la figura 4.1.

Se puede observar que, excluyendo la estrategia E_∞ ($k = \infty$), para el resto de estrategias sigue existiendo un rango pequeño en el que elegir de forma óptima el tamaño del conjunto de PB. Este rango es mayor a medida que se da una mayor libertad al algoritmo para eliminar PB, correspondiendo el mayor rango al criterio E_∞. En este último criterio la fase de eliminación no distingue entre prototipos base o prototipos que no son base. Aunque el hecho de tener un amplio rango para elegir el tamaño del conjunto de prototipos base es interesante, el funcionamiento de E_∞ es sensiblemente peor que el resto de estrategias para tamaños del conjunto de PB relativamente pequeños. Por lo tanto, las estrategias E_k, con $k < \infty$, se pueden elegir cuando interese más el número de distancias calculadas que la libertad en la elección del tamaño del conjunto de prototipos base.

En el siguiente experimento, se compara la estrategia E_∞ con el criterio de eliminación basado en el propio proceso de búsqueda (E_{elim}). Como se

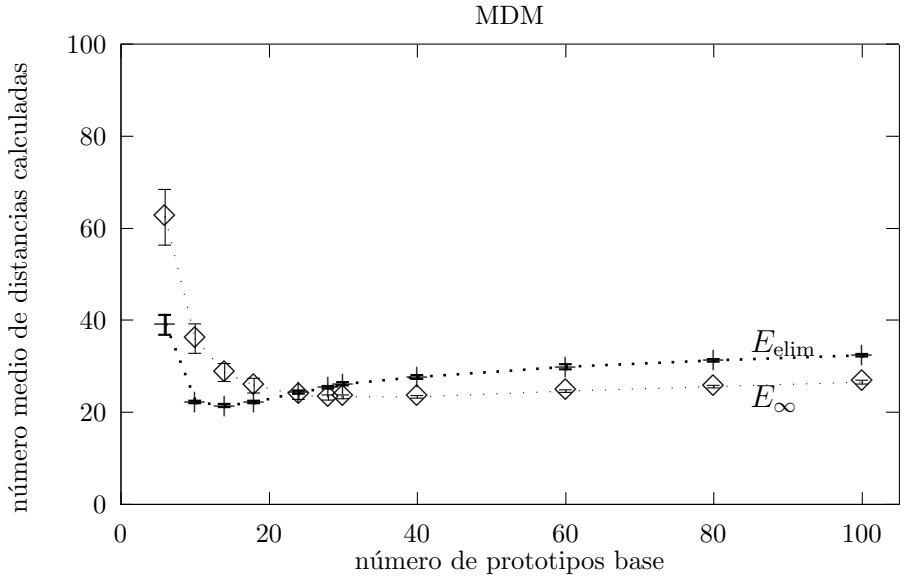

Figura 4.2: Número medio de distancias calculadas por el algoritmo en un espacio de dimensión 6, al variar el tamaño del conjunto de PB usando los criterios E_∞ y E_{elim} para un conjunto de 1024 prototipos. El criterio de selección de PB utilizado es el MDM.

puede ver en la figura 4.2, con este último criterio el valor óptimo para el tamaño del conjunto de PB tiende a ser menor que con E_∞, aunque el número de distancias sea un poco superior a partir de este tamaño. Este comportamiento podría hacer pensar en descartar este último criterio, pero se verá en la próxima sección que cambios en la fase de aproximación de los PB van a hacer que este comportamiento varíe.

4.2. Aproximación con prototipos base

Hasta ahora en la fase de aproximación del algoritmo se seleccionaban los PB de forma arbitraria. Esto no tenía mayor importancia, pues se calculaba la distancia a la muestra de todos ellos y, por tanto, el orden en que se realizaban estos cálculos no afectaba en ninguna medida a la fase de búsqueda. Sin embargo, si se permite eliminar PB es lógico que este criterio tenga que ser cambiado ya que sería posible eliminar aquellos que de alguna manera pudiesen aportar mayor información en la actualización de la cota inferior de la distancia al resto de prototipos. El criterio de aproximación que se propone para los PB es el mismo que el que se aplica al resto de prototipos, es decir, en cada iteración del algoritmo se seleccionará aquel prototipo base para el que el valor de la cota inferior de su distancia a la muestra es menor entre todos los PB que todavía no han sido eliminados. Es decir, se sustituye

$$b = \text{elemento\_aleatorio(B)} \tag{4.5}$$

por

$$b = \text{argmin}_{\forall p \in B} g[p] \tag{4.6}$$

4.2.1. Experimentos

Se ha repetido el experimento de la figura 4.2 utilizando las estrategias de eliminación E_∞ y E_{elim}, para comparar los resultados al modificar la fase de aproximación de los PB en el algoritmo.

Como se puede ver en la figura 4.3, a partir de un tamaño relativamente pequeño del conjunto de PB, $|B|$, el número medio de distancias calculadas disminuye monótonamente, de forma que los dos criterios se comportan de forma similar. Además, para tamaños pequeños del conjunto de PB, el criterio E_{elim} es mejor que el E_∞. Es más significativo todavía que el número de distancias a calcular permanezca constante (incluso decrezca ligeramente) a medida que aumenta el tamaño del conjunto de PB, mientras que con el criterio anterior de aproximación este número iba aumentando lentamente. Se puede decir que, en el límite, para $|P| = |B|$, el algoritmo tiende al AESA cuando se usa cualquiera de las dos estrategias de eliminación mencionadas. Hay que recordar que para el algoritmo AESA todos los prototipos del conjunto de trabajo son, a su vez, PB.

El experimento de la figura 4.3 fue repetido para el criterio E_{elim} variando el tamaño del conjunto de PB y diferentes dimensiones. Como se puede observar en la figura 4.4, a partir de un tamaño relativamente pequeño de $|B|$, un aumento en este valor es prácticamente irrelevante, ya que el número de distancias a calcular es prácticamente el mismo.

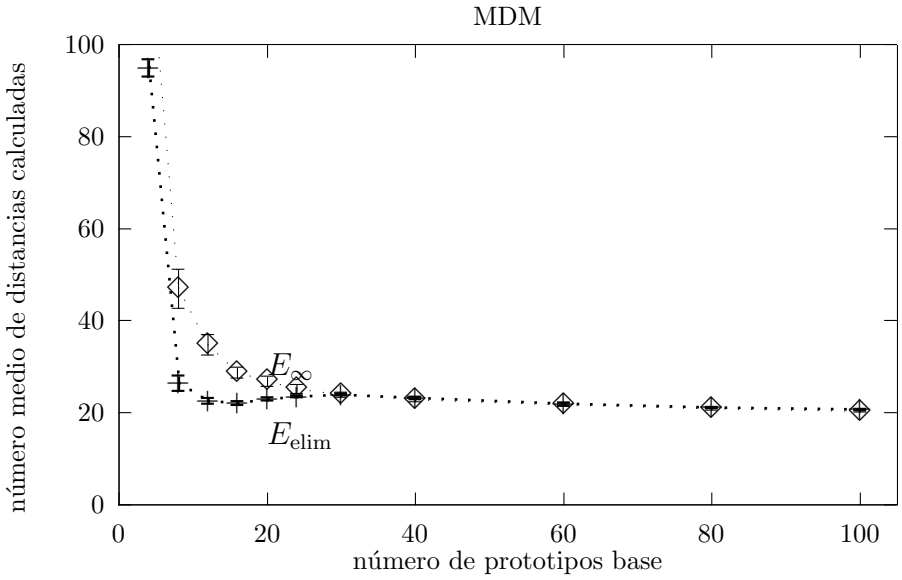

Figura 4.3: Número medio de distancias calculadas en un espacio vectorial euclídeo de dimensión 6 para el algoritmo con una nueva fase de aproximación de PB, al variar el tamaño del conjunto de PB con un conjunto de 1024 prototipos usando los criterios E_∞ y E_{elim}. El criterio de selección de PB utilizado es el MDM.

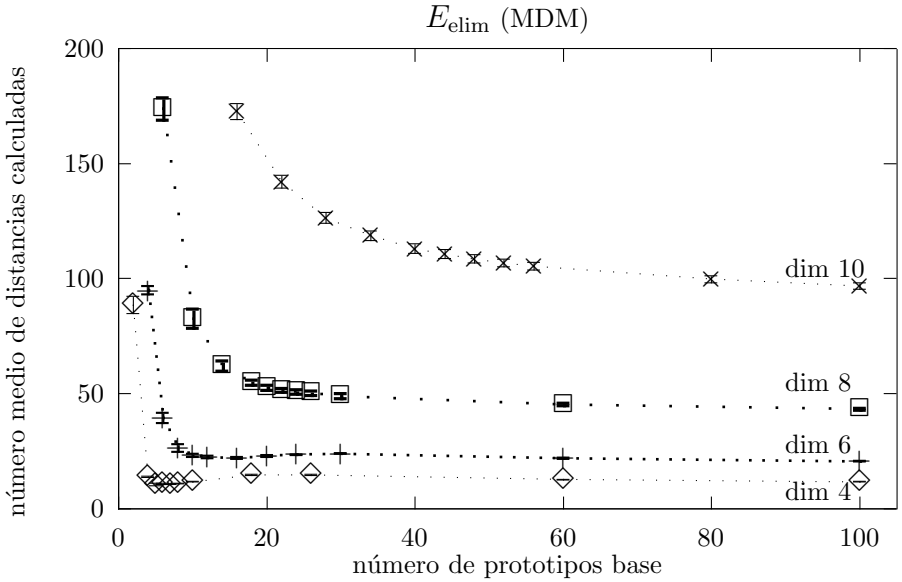

Figura 4.4: Número medio de distancias calculadas en un espacio vectorial euclídeo de N dimensiones por el algoritmo, al variar el tamaño del conjunto de PB usando el criterio E_{elim} para la nueva fase de aproximación de los PB sobre un conjunto de 1024 prototipos. El criterio de selección de PB utilizado es el MDM.

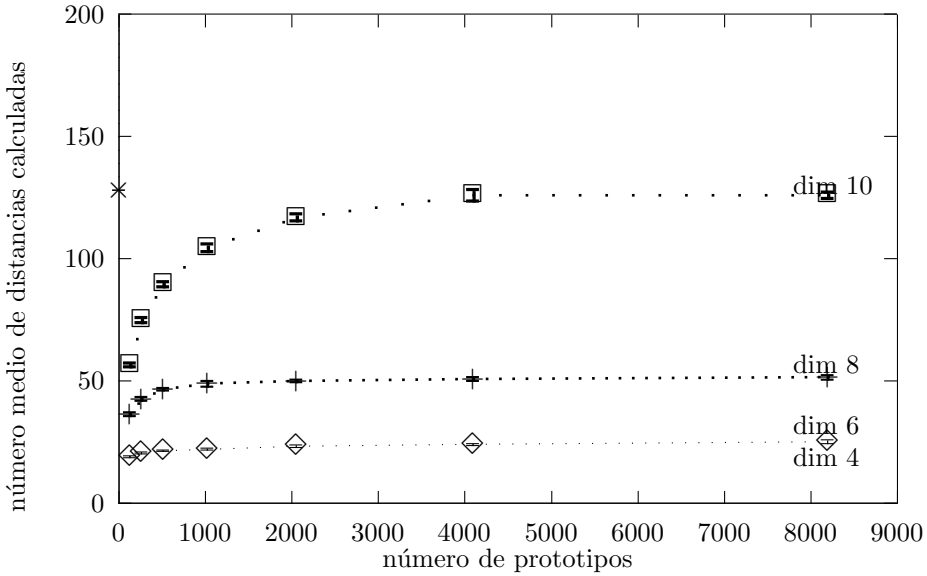

Figura 4.5: Número medio de distancias calculadas por el algoritmo, al variar el tamaño del conjunto de prototipos en un espacio vectorial euclídeo de N dimensiones usando el criterio E_{elim} para la nueva fase de aproximación de los PB elegidos con el criterio de selección de prototipos base MDM.

En el experimento presentado en la figura 4.4 se estudia el comportamiento del algoritmo LAESA cuando varía el número de prototipos $|P|$. Para este propósito, el número de PB, $|B|$, se fijó con el valor que aparece en la tabla 4.1. Este valor corresponde a la parte de la curva en la que la pendiente empieza a ser pequeña para la estrategia de eliminación E_{elim}. El procedimiento se repitió variando la dimensión y el número de prototipos, $|P|$. Los resultados muestran claramente la tendencia asintóticamente constante del número de distancias calculadas. Además, se observa una cierta mejoría en el número de distancias calculadas con respecto a la primera versión (conjunto de PB seleccionado aleatoriamente, no permitiendo la eliminación de los mismos en la fase de búsqueda).

Para poder observar mejor esta diferencia se han representado juntos los resultados de la primera versión del algoritmo (con los valores óptimos del número de PB que aparece en la tabla 4.1) con los resultados de la figura 4.5. Esta comparación se puede ver en la figura 4.6. Para la primera versión (con criterio de eliminación E_1) se ha utilizado el número óptimo de PB, mientras que para la nueva versión con el criterio de eliminación E_{elim}, se ha elegido un valor de $|B|$ mínimo perteneciente a la zona en la que el número de

DIMENSION	$\|B\|^*$ E_1 (azar)	$\|B\|^*$ E_{elim} (MDM)
4	8	7
6	20	16
8	36	30
10	60	60

Cuadro 4.1: Número óptimo de PB utilizados por los criterios de eliminación E_1 y E_{elim} con selección de los mismos siguiendo el método de selección azar y MDM, respectivamente.

distancias calculadas no sufría gran variación al aumentar este valor. Como se puede ver en la figura 4.5 el número medio de distancias sigue manteniéndose aproximadamente constante respecto al tamaño del conjunto de prototipos (para la primera versión respecto a la última) pero, además se ha reducido sustancialmente este número, sobre todo, cuando aumenta la dimensión de los datos.

Los resultados mostrados en la figura 4.5 sugieren que la eficiencia del algoritmo decrece exponencialmente con el aumento de la dimensión, como ya ocurría en la primera versión del algoritmo. Como ya se comentó en la sección 3.5.2, hay que tener en cuenta que para un número fijo de puntos uniformemente distribuidos, las distancias medias entre prototipos aumentan rápidamente con la dimensión. Este comportamiento no se espera para datos reales, ya que estos serán datos agrupados, donde el promedio de las distancias de las muestras a sus prototipos más cercanos no aumenta significativamente aunque se añadan más características a la representación de los datos (en el capítulo 5 se estudiará este hecho). También hay que tener en cuenta que el algoritmo trabaja directamente con espacios métricos, donde sólamente el concepto de dimensión intrínseca[1] [PBJD, 79] [Fuk, 90] podría tener sentido. Para una distancia adecuada, la dimensión intrínseca no aumenta necesariamente con una mayor (y mejor) representación de los datos.

[1]dimensión intrínseca de un conjunto de datos es el mínimo número de parámetros necesarios para generar ese conjunto de datos

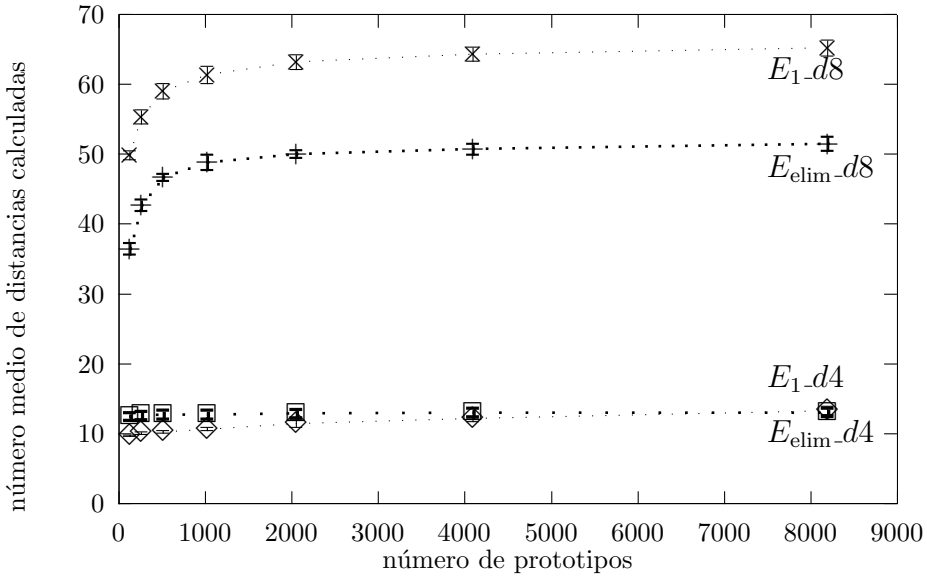

Figura 4.6: Número medio de distancias calculadas en un espacio vectorial euclídeo de N dimensiones por el algoritmo, al variar el tamaño del conjunto de prototipos usando el criterio E_{elim} para la nueva fase de aproximación de los PB utilizando el criterio de selección MDM y la primera versión del algoritmo (criterio de selección de PB SMD, criterio de eliminación E_1 y aproximación aleatoria de PB en la fase de búsqueda).

4.3. Experimentos con otras métricas y otros tipos de distribución.

En los experimentos que se muestran en esta sección las métricas utilizadas han sido, al igual que en el capítulo 3, L_1 y L_∞. Repitiendo el experimento de la figura 4.4 se ve que los valores óptimos del tamaño del conjunto de PB se mantienen para las diferentes métricas. En la figura 4.7, se puede comprobar este hecho para dimensión 4 y 8.

El comportamiento del criterio de eliminación E_{elim} es el mismo con las diferentes métricas, es decir, existe un tamaño del conjunto de PB a partir del cual el número de distancias calculadas permanece prácticamente constante. Repitiendo el experimento para un tamaño del conjunto de PB óptimo para cada dimensión (10 en dimensión 4 y 30 en dimensión 8) y variando ahora el número de prototipos, se ve que a su vez el comportamiento del algoritmo no difiere al cambiar de métrica, es decir, el número medio de distancias calculadas tiende a ser constante. Estos resultados se pueden ver en la figura 4.8.

Se han repetido los experimentos anteriores utilizando distribuciones gaussianas; concretamente, los prototipos generados pertenecen a cinco gaussianas situadas en el hipercubo unidad cuya matriz de covarianza tiene el valor 0.001 en los elementos de la diagonal principal y 0 en el resto. En este caso se ve también que el algoritmo, en general, tiene el mismo comportamiento que con distribuciones uniformes (ver figuras 4.9 y 4.10). En esta última figura no se observa claramente la tendencia asintóticamente constante del número de distancias respecto al número de prototipos. Esto puede ser debido a que esta tendencia se produce para conjuntos más grandes de prototipos o a que el número de PB utilizado tiene que ser mayor.

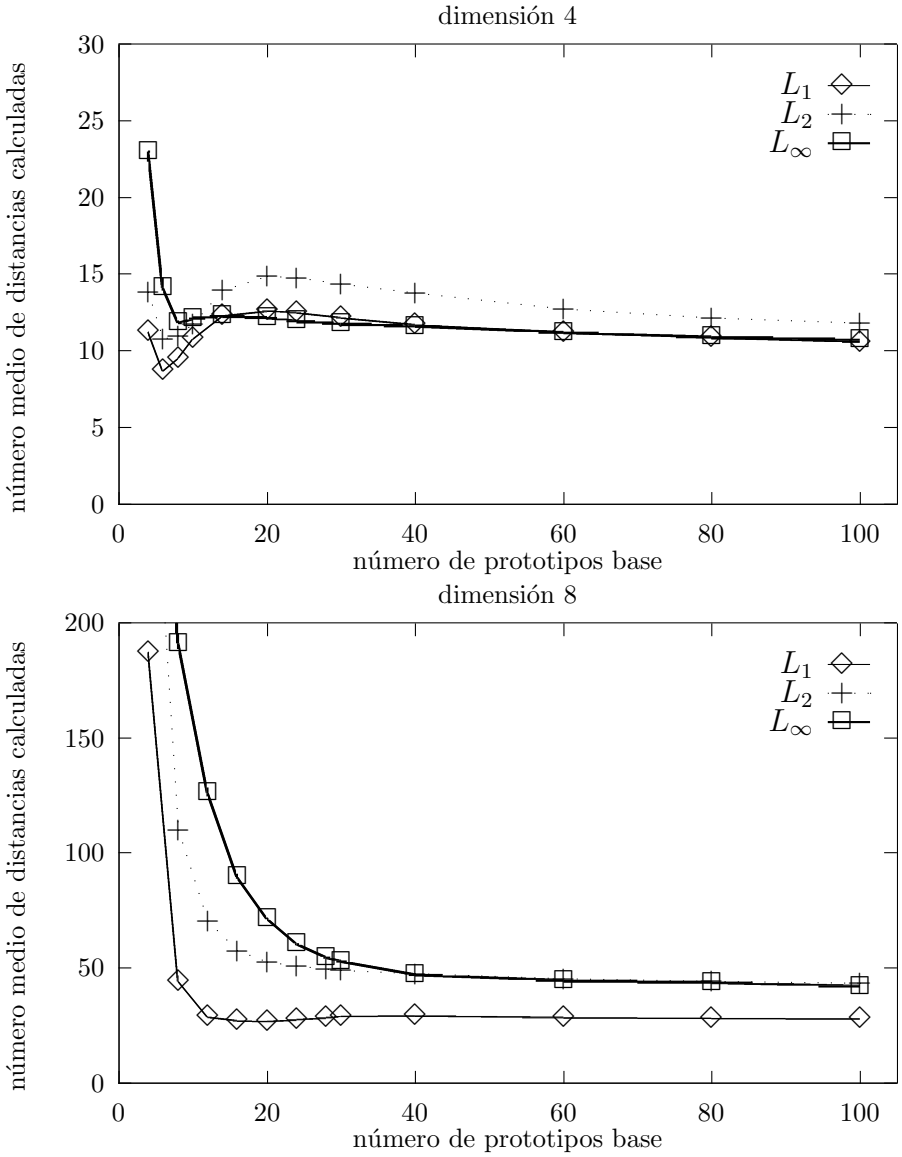

Figura 4.7: Número medio de distancias calculadas en un espacio de dimensión 4 y 8 por el algoritmo utilizando el criterio de eliminación E_{elim} para las métricas L_1, L_2 y L_∞ (MDM) al variar el número de PB para un conjunto de 1024 prototipos. La desviación típica de los datos representados es del 3 % para 8 PB, reduciéndose a medida que aumenta el número de PB.

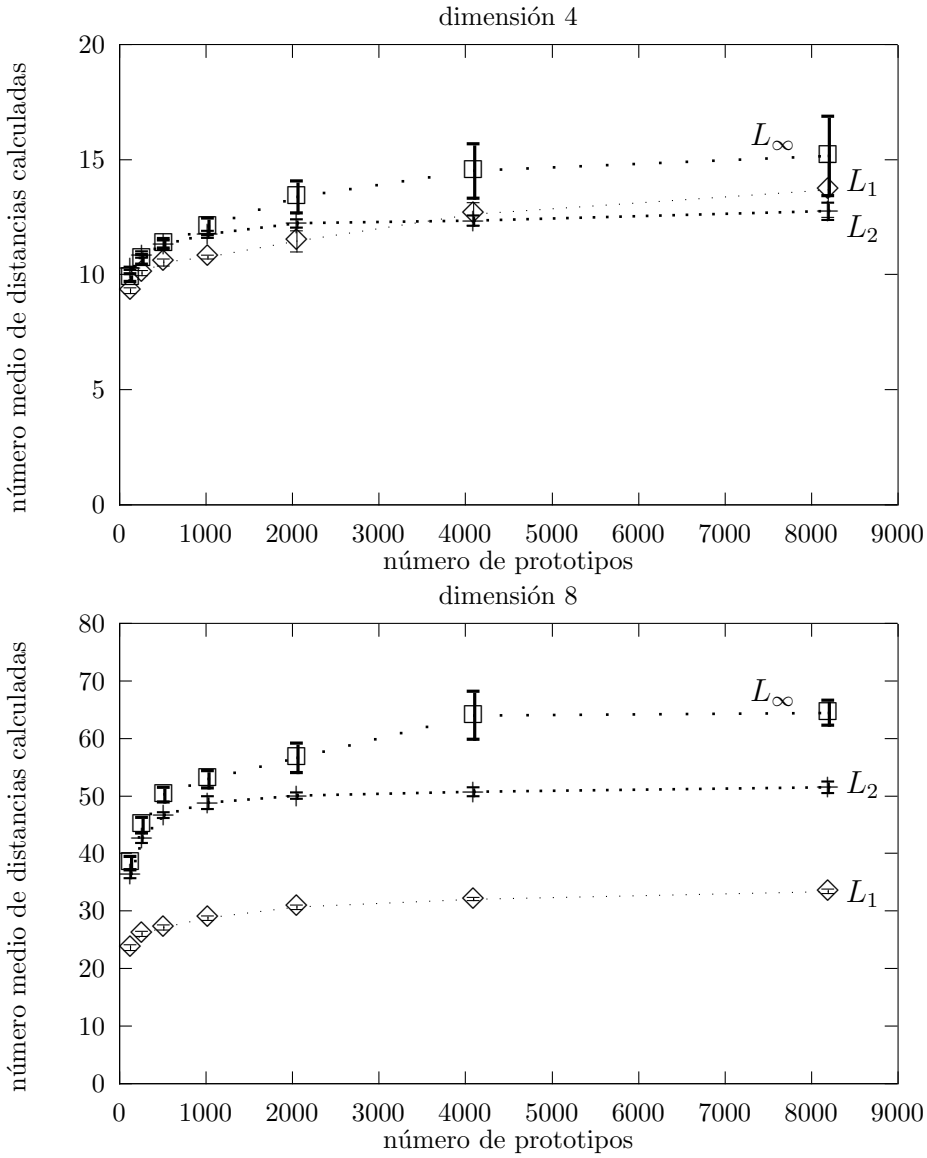

Figura 4.8: Número medio de distancias calculadas en un espacio de dimensión 4 y 8 por el algoritmo utilizando el criterio de eliminación E_{elim} para las métricas L_1, L_2 y L_∞ variando el tamaño del conjunto de prototipos.

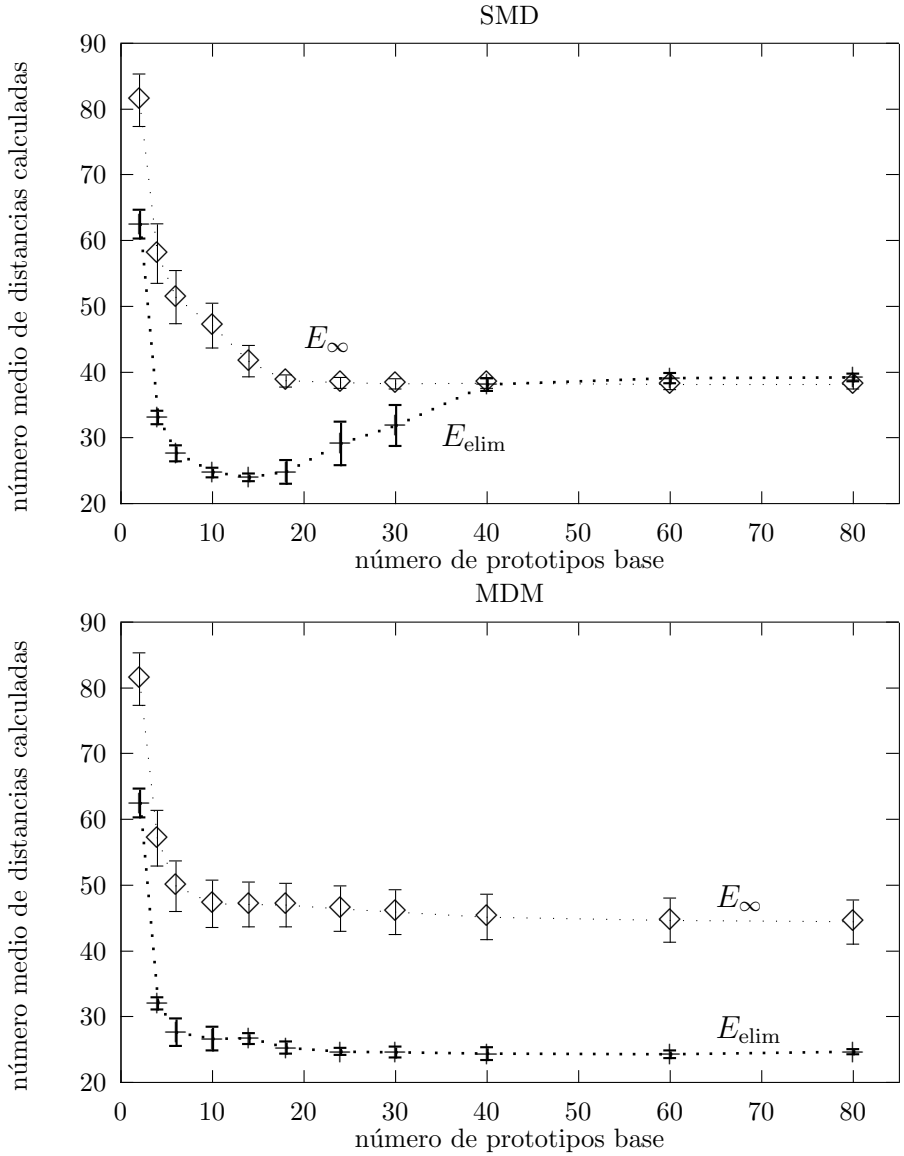

Figura 4.9: Número medio de distancias calculadas por el algoritmo al variar el número de prototipos base utilizando los criterios de eliminación E_{elim} y E_∞ y los criterios de selección de prototipos base SMD y MDM sobre un conjunto de 1024 prototipos de dimensión 6 extraídos de distribuciones gaussianas. La métrica utilizada ha sido la euclídea.

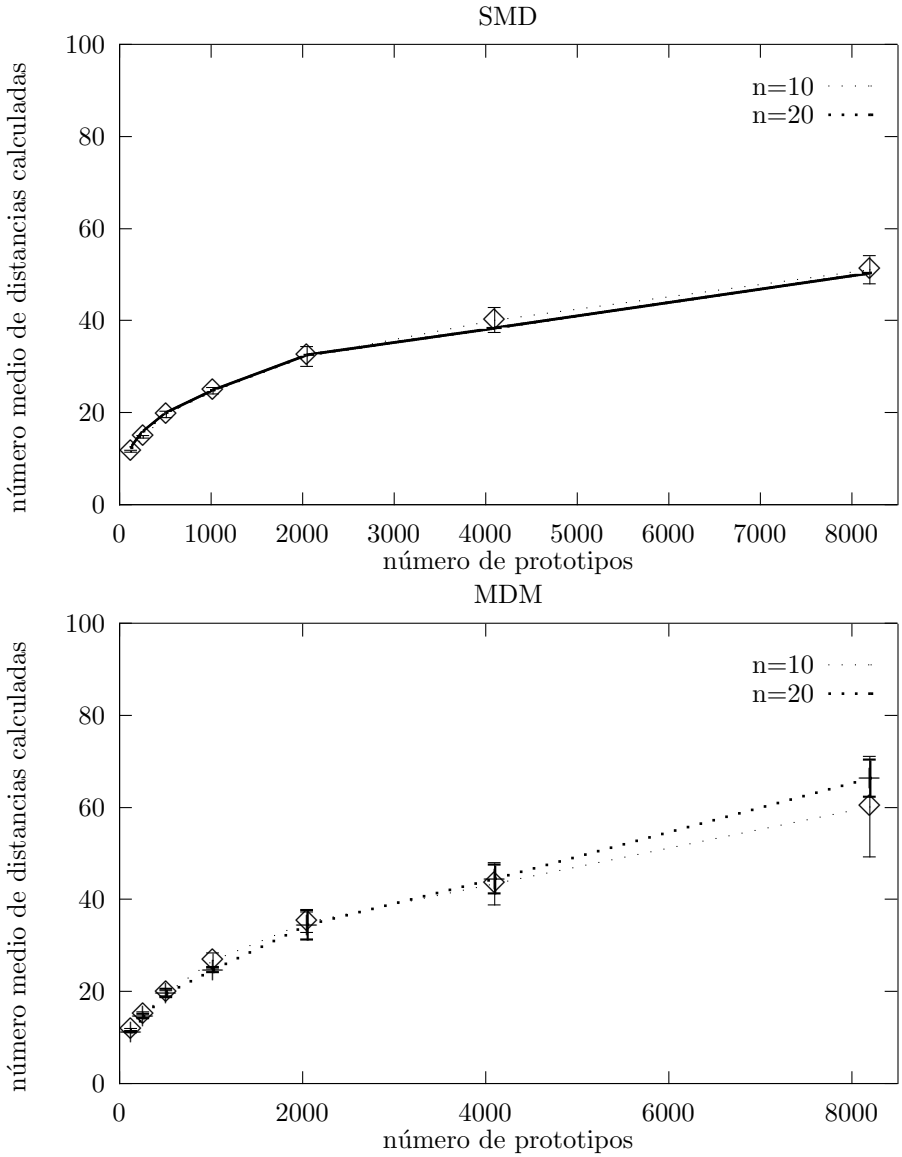

Figura 4.10: Número medio de distancias calculadas por el algoritmo al variar el número de prototipos utilizando el criterio de eliminación E_{elim} y los criterios de selección de prototipos base SMD y MDM para conjuntos de prototipos de dimensión 6 extraídos de distribuciones gaussianas. La métrica utilizada ha sido la euclídea.

4.4. Conclusiones

Los resultados presentados en las secciones anteriores demuestran claramente que el algoritmo con los nuevos criterios de manipulación de PB sigue comportándose como con la primera versión (en el sentido de calcular un número medio de distancias constante conforme aumenta el tamaño del conjunto de prototipos). En este capítulo, además, se han estudiado una serie de estrategias de manipulación simple de los PB en la fase de eliminación. El comportamiento del algoritmo con las aportaciones efectuadas en este capítulo ha sido estudiado a través de una serie de experimentos con datos sintéticos en espacios métricos. En la mayoría de los casos los prototipos y las muestras fueron extraídos de una distribución uniforme en hipercubos de N dimensiones. Las coordenadas de los puntos no fueron utilizadas nunca directamente y solamente sirvieron para calcular las distancias entre puntos usando una métrica apropiada que, en este caso, fue la norma euclídea.

La utilización de las estrategias propuestas (que han permitido relajar la elección del tamaño del conjunto de PB) combinada con una correcta elección de los PB en la de preproceso, hace que el número medio de distancias calculadas disminuya considerablemente respecto a los resultados anteriores. Experimentos con otras métricas y otros tipos de distribución también fueron realizados pero con similares resultados. Se podrían obtener aún algunas mejoras sobre los resultados presentados si se utilizan las ideas introducidas en [SW, 90]. Shasha y Wang calculan dos matrices aproximadas de distancias (MAD y MIN) que representan la mayor cota inferior (MAD) y la menor cota superior (MIN) de todos los caminos entre cada par de prototipos P. Esta información es utilizada en un algoritmo de búsqueda propuesto en el trabajo anterior. Algunos pequeños experimentos realizados utilizando las dos cotas no mejorar significativamente los resultados obtenidos con el LAESA (en el que solo se utiliza la cota inferior) y sin embargo la complejidad espacial es superior.

Capítulo 5

Criterios de selección de prototipos en la fase de búsqueda

Los experimentos del capítulo anterior muestran que el número medio de distancias calculadas aumenta exponencialmente con la dimensión de los datos (esto también ocurría con el algoritmo AESA). Este capítulo está dedicado al estudio de este hecho. Concretamente, se muestra que este aumento exponencial se produce no por el aumento de la dimensión, sino por el incremento de la distancia media entre las muestras y sus prototipos más cercanos. Esto ocurre en las distribuciones uniformes utilizadas cuando aumenta la dimensión y se mantiene constante el número de prototipos.

Para hacer independiente el aumento de la distancia media entre prototipos de la dimensión, se han generado muestras cuya distancia a su vecino más próximo está acotada por algún valor constante e independiente de la dimensión. Se han realizado dos tipos de experimentos para estudiar en este caso el comportamiento del algoritmo: 1) fijando el tamaño del conjunto de PB para cualquier dimensión y 2) fijando este valor a los valores óptimos obtenidos para cada dimensión en el capítulo anterior para los diferentes criterios de eliminación. Para el primer caso se quiere comprobar que reduciendo y fijando el tamaño del conjunto de PB en diferentes dimensiones, el número de distancias calculadas no aumenta exponencialmente con la dimensión. En el segundo caso, será necesario introducir un nuevo criterio de selección para demostrar que el número medio de distancias calculadas no aumenta con la dimensión.

Se va a comprobar que al reducir la distancia del vecino más próximo a una muestra dada ($D_{\min}$) manteniendo constantes el resto de parámetros del algoritmo (número de prototipos en P y número de PB), el número de

distancias calculadas no aumenta con la dimensión. En la fase de aproximación del algoritmo, hasta ahora siempre se han seleccionado primero los PB para calcular la distancia a la muestra en la fase de búsqueda. Esto se ha hecho así porque son los PB los que permiten actualizar la cota inferior de la distancia a la muestra del resto de prototipos no eliminados.

Para un número preestablecido de PB, el algoritmo se ha utilizado para clasificar muestras que se encuentran a diferentes distancias, $D_{\min}$, de su mismo vecino más próximo, *min*. Mientras quedan PB el algoritmo selecciona un PB para calcular su distancia a la muestra. Por lo tanto, el resultado parcial de la búsqueda de dos muestras que sólo se diferencian en la distancia a su prototipo más cercano es el mismo (en número de distancias calculadas y número de prototipos eliminados). Sólamente cuando el algoritmo empieza a seleccionar prototipos que no son base en la aproximación (porque ya no quedan PB, se observa que el número de distancias que se calculan para la muestra a menor distancia es ligeramente inferior. Sin embargo, esta reducción no es suficiente para reducir el incremento exponencial en el número de distancias calculadas con respecto a la dimensión cuando se utilizan los tamaños óptimos del conjunto de PB.

Para evitar este problema se propone un nuevo criterio que permite seleccionar prototipos que no son base desde el primer momento, sin tener que esperar a que el conjunto de PB esté vacío para ello [MO, 92]. Se comprobará experimentalmente que el número de distancias calculadas por el algoritmo con el nuevo criterio de selección no aumenta con la dimensión de los datos, sino que únicamente influye el incremento de la distancia de la muestra a su prototipo más cercano.

Se ha trabajado con dos criterios de selección diferentes, siendo el segundo una generalización del primero.

5.1. Criterio "siempre prototipos base", SPB

Este es en realidad el criterio que se ha venido utilizando hasta el momento. Consiste en escoger siempre los PB mientras este conjunto no esté vacío. En la fase de aproximación del algoritmo se van eligiendo como candidatos a la selección dos prototipos b y q, (uno base y otro no) respectivamente. Si el valor de b no está indeterminado (significa que el conjunto de PB no está vacío), el criterio seleccionará este prototipo. Este es el criterio utilizado en los capítulos anteriores, solamente que ahora este comportamiento se expresa a través de una función denominada SELECCIÓN (en la sección 5.4 se presenta el algoritmo LAESA con la incorporación de esta función):

$$\text{SELECCIÓN}(b, q) = (\textbf{si}\, b \neq \text{ indeterminado } \textbf{entonces}\, b\, \textbf{si no}\, q\, \textbf{fin si}) \tag{5.1}$$

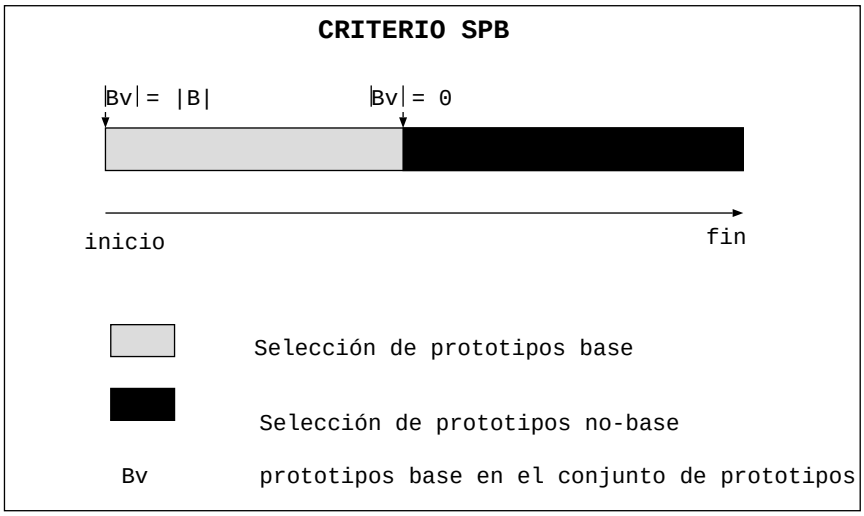

Figura 5.1: En la primera parte de la ejecución del algoritmo los prototipos seleccionados como candidatos a ser el vecino más próximo son siempre PB. Cuando este conjunto está vacío, se continúa seleccionando prototipos del resto del conjunto.

En los capítulos anteriores se ha observado experimentalmente que el número de distancias calculadas para los valores óptimos del tamaño del conjunto de PB, $|B|^*$, aumenta rápidamente con la dimensión. Esto ocurre porque en el LAESA no se permite calcular la distancia de la muestra a un prototipo que no es base hasta que el conjunto de PB esté vacío, siendo este hecho independiente de que las muestras se encuentren a mayor o menor distancia de sus prototipos más cercanos (ver figura 5.1). Además, también hay que tener en cuenta que no sólamente se está aumentando la distancia del vecino más próximo a una muestra dada cuando aumenta la dimensión, sino que también se ha ido aumentando el tamaño del conjunto de PB para obtener un mejor comportamiento.

5.2. Criterio "alterna prototipos base", APB

La función SELECCIÓN alternativa que se propone permite seleccionar prototipos que no son base desde que comienza el proceso de búsqueda.

Está claro que si se supiese que el prototipo seleccionado, y que no es base, está muy cercano a la muestra, sería muy útil utilizarlo en la fase de eliminación, pues permitiría eliminar un gran número de prototipos. Es evidente por otro lado que no se debe calcular indiscriminadamente en cada paso la distancia a un prototipo que no es base pues esto llevaría a incrementar el número de distancias calculadas sin actualizar para nada las aproximaciones (cotas inferiores de la distancia) del resto de prototipos. Por ello, se utilizará el siguiente criterio: se considerará que un prototipo que no es base es bastante cercano a la muestra cuando este prototipo haya sido elegido como tal en un cierto número de iteraciones consecutivas (ver figura 5.2). La función SELECCIÓN queda de la siguiente forma:

$$\text{SELECCIÓN}_{\text{r}}(b, q) = \textbf{si en las}\ r\ \text{iteraciones anteriores}\ q\ \text{no varía} \\ \textbf{entonces}\ q\ \textbf{si no}\ b\ \textbf{fin si} \qquad (5.2)$$

donde

$b \in B$
$q \in P - B$
$r \in [1 \ldots |P|]$

El caso límite de este criterio de selección se da cuando $r = |P|$. Esto significa que nunca se van a elegir prototipos que no son base antes de que el conjunto de PB esté vacío; en otras palabras, es el criterio de selección descrito en la sección 1.

Es de esperar que cuando la muestra se encuentre muy cercana a su vecino más próximo el nuevo criterio de selección, con $r < |P|$, permita llegar a él más rápidamente.

5.3. Experimentos

En el capítulo anterior se ha visto que tanto el número de distancias como el tamaño del conjunto de PB aumenta exponencialmente con la dimensión. En este capítulo se describe lo que ocurre con estos valores cuando se trabaja con datos agrupados. El estudio se hará de dos formas: en primer lugar se mantiene un tamaño fijo para el conjunto de PB cuando se aumenta la dimensión, y en segundo lugar se aplicarán los tamaños óptimos que se han obtenido en capítulos anteriores para cada dimensión. Para el criterio de selección SPB, $\text{SELECCIÓN}_{|P|}$, la estrategia de eliminación más interesante es la llamada E_{elim}, en la cual, a partir de un tamaño relativamente pequeño

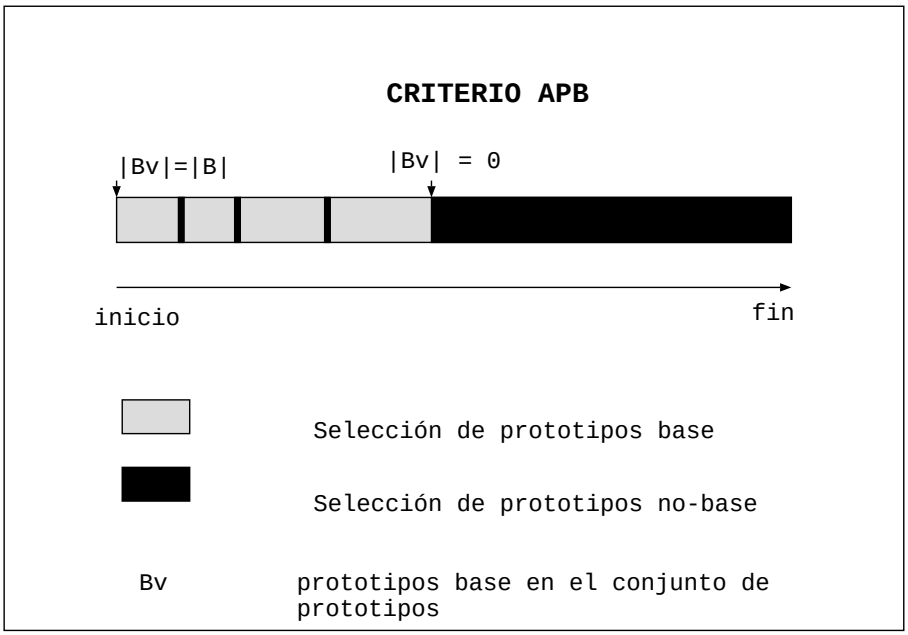

Figura 5.2: En la primera parte de la ejecución del algoritmo se permite seleccionar prototipos que no son base como candidatos a ser el vecino más próximo.

del conjunto de PB el aumento de este tamaño no afecta al número de distancias calculadas. El comportamiento del algoritmo en estas circunstancias ha sido ya estudiado en el capítulo anterior en el que, además, se ha podido observar el aumento del número de distancias calculadas con la dimensión. Sin embargo, se sospecha que este aumento en el número de distancias no es consecuencia directa del incremento de la dimensión, según un estudio efectuado por Vidal en [Vid, 85], sino de las distancias medias entre las muestras y sus vecinos más próximos.

Se ha supuesto que las muestras son versiones mas o menos distorsionadas de los prototipos. Para ello, se han realizado experimentos utilizando *intervalos de tolerancia* alrededor de cada prototipo (ver figura 5.3). Dentro de estos intervalos se han generado aleatoriamente las muestras sobre las que se aplicará posteriormente el algoritmo de búsqueda. En los distintos experimentos realizados se ha modificando el tamaño de estos intervalos, de forma que la distancia media de cada muestra a su vecino más próximo depende de este valor. Estos intervalos de tolerancia deberían ser en realidad esferas alrededor de los prototipos sobre los que se generan las muestras. Sin embargo, se han utilizado hipercubos por repetir exactamente el mismo experimento

realizado por Vidal en [Vid, 85].

Todos los experimentos de esta sección se han llevado a cabo utilizando únicamente la métrica euclídea y el criterio de selección de prototipos base MDM. En el primer experimento, manteniendo constante el tamaño del conjunto de PB, se estudia la dependencia del número medio de distancias calculadas respecto a la dimensión. El experimento se realizó utilizando diferentes valores para los intervalos de tolerancia y para el tamaño del conjunto de PB, $|B|$. Los resultados se ven en la figura 5.4. En vista de estos resultados, se puede afirmar que para un valor de $|B|$ fijo y muestras generadas dentro de unos intervalos de tolerancia bajos (inferiores al 12 %), la eficiencia del algoritmo, medida en términos del número de distancias calculadas, no aumenta exponencialmente con la dimensión y, además, si el número de PB utilizado es superior a 5 este número tiende a permanecer constante a medida que aumenta la dimensión.

En el siguiente experimento se ha repetido el anterior pero utilizando para cada dimensión el valor de $|B|$ óptimo de resultados del capítulo 4. En este caso también se trata de comprobar la dependencia del número de distancias calculadas respecto a la dimensión. Para ello se utilizaron los tamaños del conjunto de PB obtenidos en el capítulo 4 y que aparecen en la tabla 4.1 para cada dimensión e intervalos de tolerancia. El criterio de selección utilizado fue SPB (este es el criterio que se han utilizado hasta ahora en los capítulos 2, 3 y 4).

Como se puede observar en la figura 5.5, a medida que se reduce el tamaño de los intervalos de tolerancia se reduce el número medio de distancias en todas las dimensiones. Sin embargo, incluso para tamaños muy pequeños (12.5 %) de estos intervalos, sigue produciéndose el aumento del número de distancias con la dimensión. La explicación de este comportamiento es que la utilización de este criterio de selección hace que el algoritmo sólo empiece a seleccionar (y calcular distancias) de prototipos que no son base cuando el conjunto de PB está vacío. Debido a ello, el hecho de que las muestras estén agrupadas o no a su vecino más próximo es independiente en la primera parte de la ejecución (excepto cuando éste sea concretamente un PB). De hecho, se ha comprobado experimentalmente cuántos prototipos que no son base se seleccionan una vez que se ha quedado vacío el de PB. El resultado es que este número tiende a ser 1 cuando las tolerancias son del orden del 12,5 %. Por lo tanto, si se permite seleccionar este prototipo en iteraciones previas, se evitarían bastantes cálculos de la distancia.

Para el nuevo criterio de selección, APB, presentado en la sección anterior, llamado SELECCIÓN$_r$, ha sido inicialmente comprobada su eficiencia con los criterios de eliminación E_{elim} y E_∞ sin utilizar al principio las tolerancias. Los resultados con los dos criterios de eliminación son algo diferentes

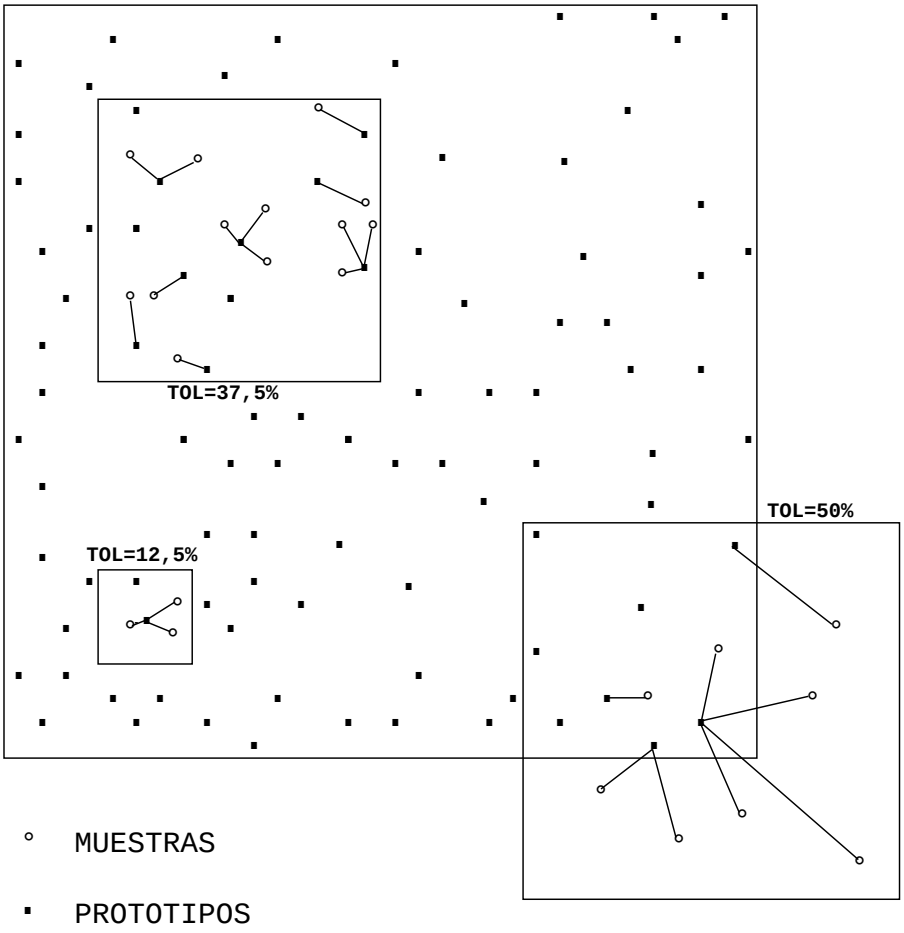

Figura 5.3: Distribución uniforme de prototipos y muestras con ilustración del concepto de intervalo de tolerancia. Los porcentajes representan el tamaño relativo de dichos intervalos (cuadros menores) respecto del lado del hipercubo mayor.

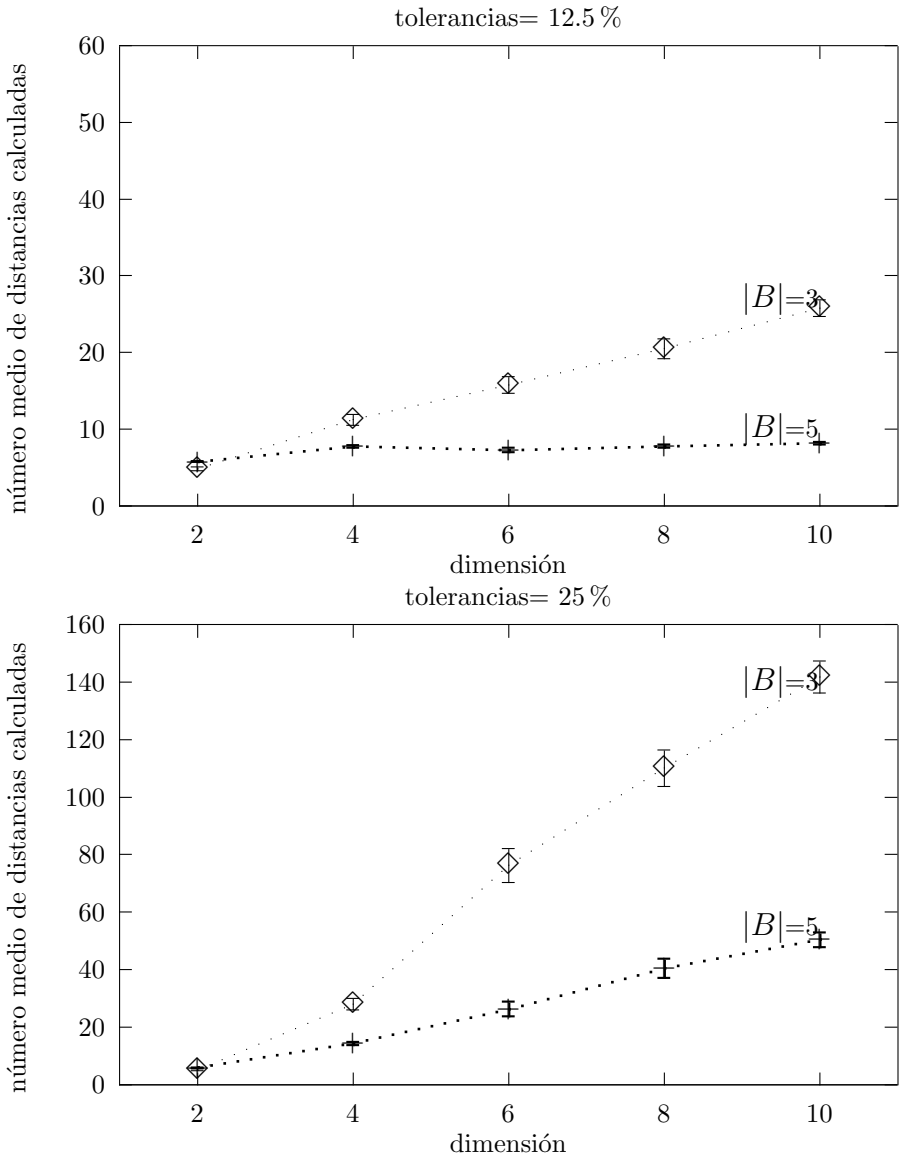

Figura 5.4: Número medio de distancias calculadas utilizando el criterio de selección SPB, variando la dimensión con tolerancias del 12.5 % y 25 % para un conjunto de 1024 prototipos. Los tamaños para el conjunto de PB utilizados han sido $|B| = 3$ y $|B| = 5$.

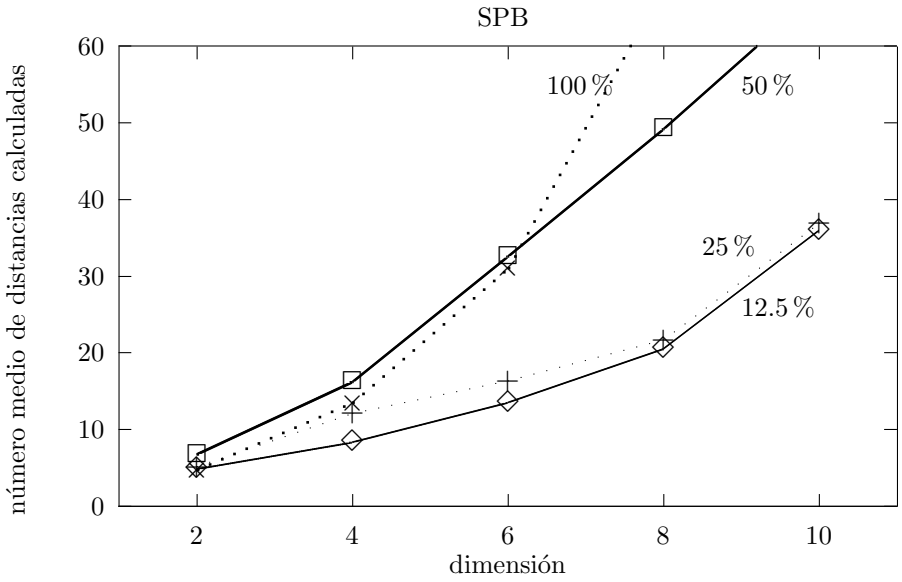

Figura 5.5: Número medio de distancias calculadas utilizando el criterio de selección SPB, variando la dimensión y la tolerancia (cercanía de las muestras a sus vecinos más próximos) para un conjunto de 1024 prototipos. La desviación típica de los datos es del 3 % en promedio. En este experimento se ha utilizado en cada dimensión el tamaño óptimo para el conjunto de PB dado por la tabla 4.1, a diferencia de la figura 5.4 en la que este valor era el mismo en todas las dimensiones.

aunque en ambos casos el número de distancias disminuye a medida que aumenta el tamaño del conjunto de prototipos (ver figura 5.6). Por ello, el resto de experimentos han sido realizados utilizando únicamente el criterio E_{elim}. Concretamente, para este criterio, los resultados de la figura 5.6 muestran que en dimensión 6 el algoritmo se comporta mejor al utilizar el criterio APB que el SPB cuando $r = 4$.

También se pone de manifiesto, no tanto en las gráficas por lo parecido de los resultados sino por la definición de los dos criterios, que los resultados del criterio SPB coinciden con los resultados del criterio APB para $r = |P|$. En general, utilizando el criterio E_{elim} de eliminación con muestras generadas aleatoriamente en el hipercubo unidad, no hay grandes diferencias de utilización de un criterio a otro: las diferencias entre ambos criterios de selección se notan cuando estas muestras estén muy cercanas a sus vecinos más próximos.

El hecho de seleccionar en las fases tempranas del algoritmo un prototipo que no es base beneficia en el sentido que que al encontrarse *probablemente* muy cercano a la muestra, el valor de $D_{\min}$ es muy pequeño, produciéndose muchas eliminaciones en la fase correspondiente. El propio criterio no permite que esta selección se realice dos veces seguidas y por lo tanto conseguimos en un paso eliminar muchos prototipos, pasando en el siguiente (con la selección de un PB) a seguir actualizando la cota inferior de la distancia al resto de prototipos no eliminados.

En la figura 5.7 se ha realizado un estudio del número de distancias calculadas conforme aumenta la dimensión del espacio para el criterio APB con $r = 4$. Para ello, se han utilizado varios intervalos de tolerancia (1.5 %, 12.5 %, 25 %, 50 %). Los experimentos realizados con este nuevo criterio con muestras agrupadas se han llevado a cabo utilizando para cada dimensión el número de PB obtenidos con el criterio de eliminación E_{elim} y el criterio de selección SPB que aparece en la tabla 4.1 del capítulo anterior.

Comparando las figuras 5.5 y 5.7, se puede ver que ambos criterios tienen un comportamiento similar cuando las muestras no están muy cercanas a sus vecinos más próximos, mientras que cuando sí lo están la utilización del nuevo criterio de selección hace que el número de distancias no aumente con la dimensión, sino que incluso llega a disminuir. Se puede concluir entonces, a la vista de los resultados, que con el criterio APB se ha conseguido mantener el número de distancias constante con la dimensión, incluso cuando se aumenta el tamaño del conjunto de PB.

Para finalizar esta sección, realizamos un último estudio referente al comportamiento del algoritmo con muestras agrupadas y diferentes valores del parámetro r. Como se puede ver en la figura 5.8 se ha repetido el mismo experimento con tolerancias del 12.5 % y del 25 %. La decisión de tomar un valor u otro para r no es crítica siempre que no sea muy grande (en este

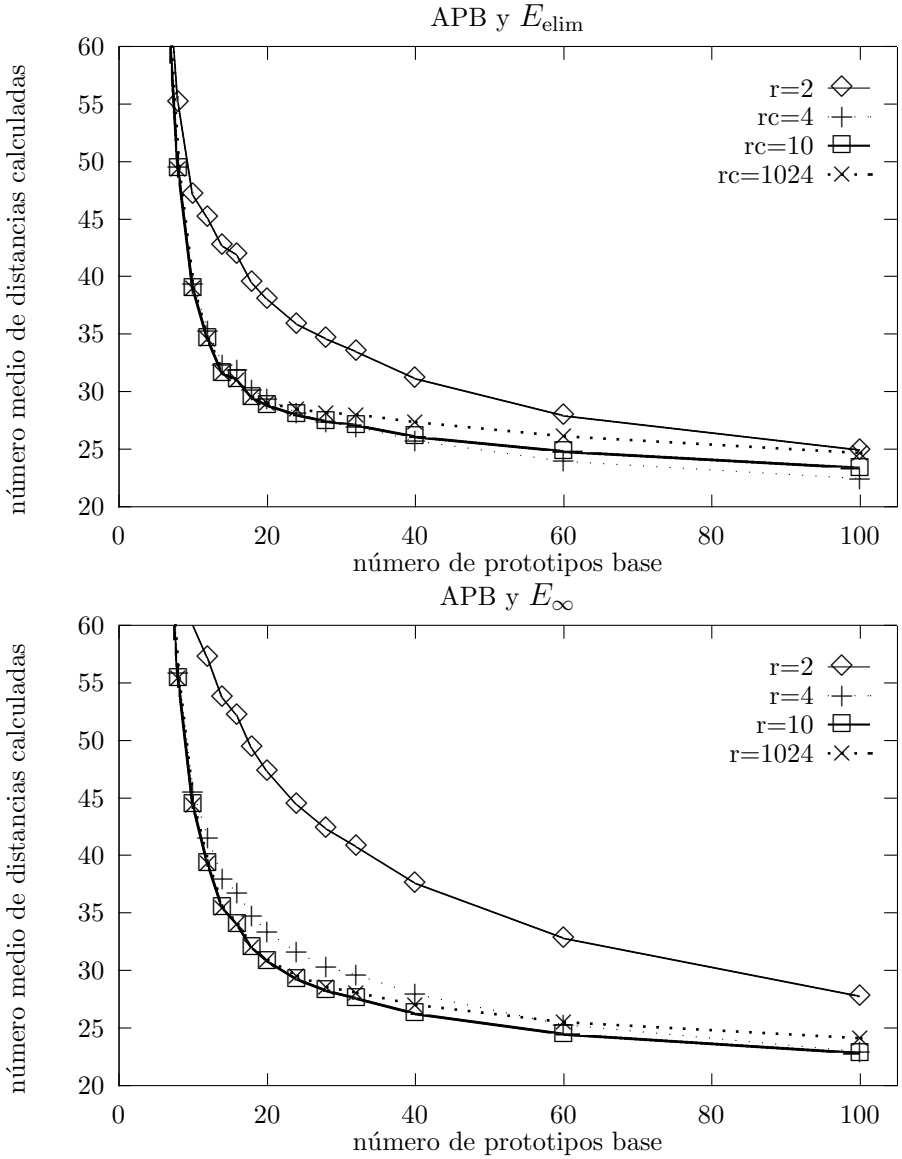

Figura 5.6: Número medio de distancias calculadas utilizando el criterio de selección APB, variando el número de prototipos base y el parámetro r. Los criterios de eliminación utilizados han sido E_{elim} y E_{∞} para un conjunto de 1024 prototipos pertenecientes a un espacio vectorial euclídeo de dimensión 6. La curva representada para el valor r=1024, corresponde al criterio SPB. La desviación típica de los datos es de un 10 % para tamaños pequeños del conjunto de PB, disminuyendo progresivamente hasta el 1 % para 100 PB.

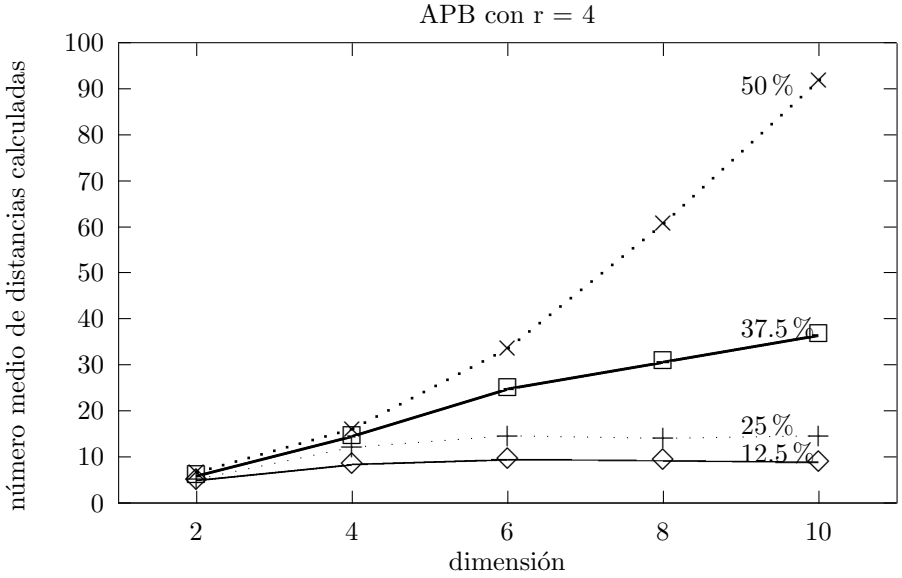

Figura 5.7: Número medio de distancias calculadas por el algoritmo utilizando el criterio de selección APB para $r = 4$, variando la dimensión y la tolerancia (cercanía de las muestras a sus vecinos más próximos, sección 5.3). La desviación típica de los datos es del 3 % en promedio.

último caso obtendríamos el comportamiento del criterio SPB).

5.4. El nuevo algoritmo LAESA

Para reunir todas las aportaciones de este capítulo y los anteriores, se ha reescrito el algoritmo para que de una forma flexible se pueda expresar cada uno de los diferentes criterios de eliminación y selección que se han ido introduciendo. Estas modificaciones consisten en la introducción de las funciones genéricas, CONDICIÓN y SELECCIÓN$_r$, que permitirán la gestión de los PB en las distintas fases del proceso de búsqueda. El algoritmo realiza, como ya se indicó en el capítulo 2, básicamente cinco pasos, que se repiten hasta que todos los prototipos han sido eliminados:

1. Cálculo de distancia de la muestra a un prototipo
2. Actualización del vecino más próximo a la muestra *min* y su distancia $D_{\min}$
3. Actualización de la cota inferior de la distancia g a cada prototipo
4. Aproximación
5. Eliminación

Alguna de estas fases ha cambiado ligeramente en el algoritmo para hacer que los cambios que se producen por la utilización de un criterio u otro sean más claros. La primera iteración empieza con el *cálculo de la distancia* (paso 1) de la muestra a un prototipo base arbitrariamente seleccionado. Si esta distancia es menor que la del vecino más próximo a la muestra hasta el momento se actualiza este valor (paso 2). A continuación se *actualiza la cota inferior de la distancia* (paso 3) al resto de prototipos. Este último paso es ejecutado solamente para los PB, ya que las distancias al resto de prototipos en la matriz precalculada M solo existen para estos prototipos. En función de la cota inferior de la distancia, el paso de *aproximación*, (paso 4), selecciona sucesivamente prototipos no eliminados del conjunto B siempre y cuando ciertas condiciones se cumplan; en otro caso, será seleccionado un prototipo del conjunto $P - B$. La función SELECCIÓN ayudará a decidir si va a ser seleccionado un prototipo base o no. La *eliminación*, (paso 5), se realiza para un prototipo p cuando se cumple

$$g_p \geq D_{\min} \ \wedge \ \text{CONDICIÓN} \tag{5.3}$$

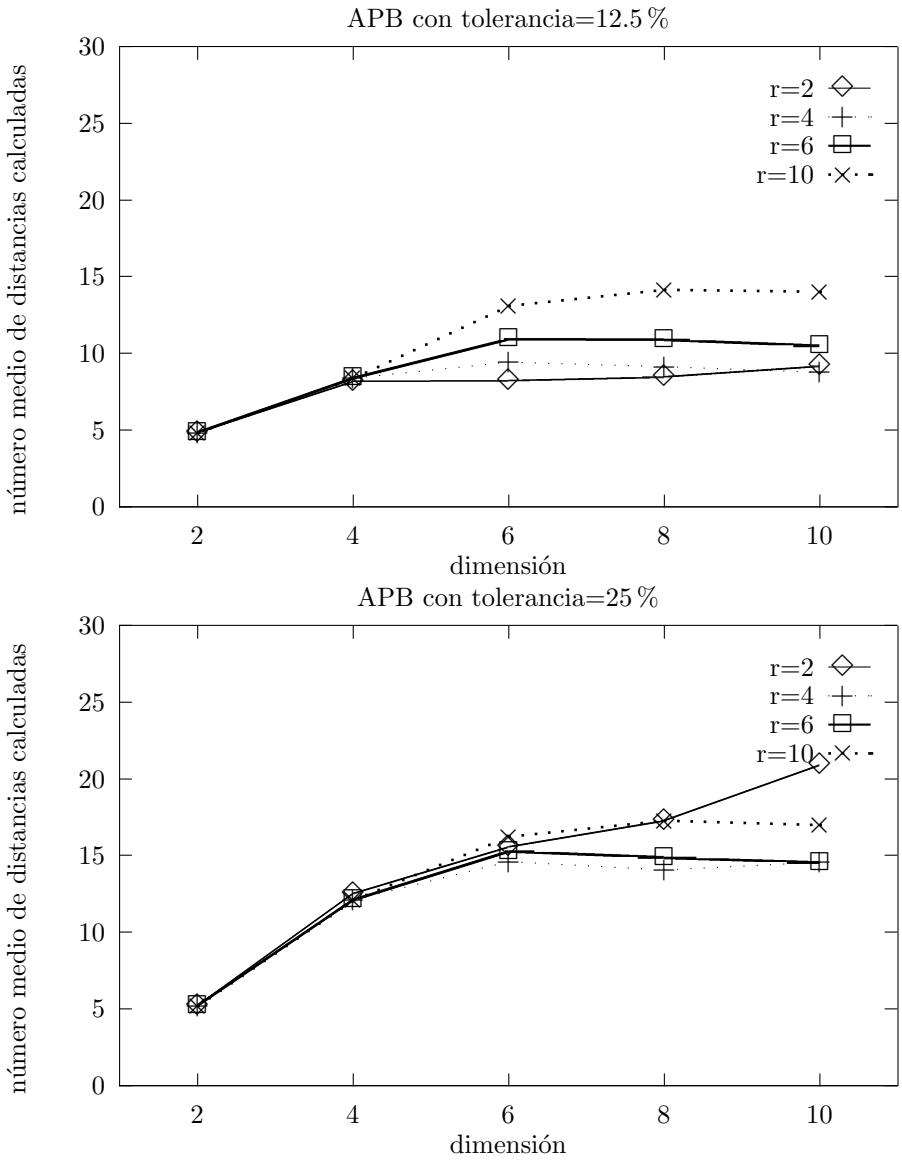

Figura 5.8: Número medio de distancias calculadas por el algoritmo utilizando el criterio de selección APB con tolerancias del 12.5 % y del 25 %, variando la dimensión para diferentes valores del parámetro r. La desviación típica de los datos es del 3 % en promedio.

donde g_p es el valor de la cota inferior de la distancia del prototipo p a la muestra y $D_{\min}$ es la distancia del prototipo más cercano a la muestra hasta el momento. La función CONDICIÓN siempre es cierta cuando el prototipo sobre el que se va a realizar la eliminación no es base. Sin embargo, cuando los prototipos son base, esta función tiene otro comportamiento. Diversas propuestas para estas funciones han sido realizadas en el capítulo 4, siendo su comportamiento probado a partir de experimentos. La versión definitiva del algoritmo se puede ver en la figura 5.9.

5.4.1. Coste del algoritmo LAESA

El coste computacional del algoritmo *LAESA* depende directamente del número de iteraciones del bucle principal **mientras**. Los experimentos realizados sugieren que para tamaños grandes del conjunto de prototipos y un número fijo de PB este número es asintóticamente constante en promedio (es decir, independiente del número total de prototipos). El coste temporal ("overhead"), por otro lado, también depende de otros factores no asociados con las distancias calculadas. Este coste es en promedio $O(|P|)$ (ya que el bucle externo se ejecuta un número constante en promedio de veces y el interior es lineal con $|P|$, y en el peor caso $O(|P|^2)$ (cuando se calculan las distancias de todos los prototipos a la muestra, caso muy improbable).

5.5. Conclusiones

En este capítulo se ha estudiado el incremento exponencial del número de distancias calculadas por el algoritmo con respecto a la dimensión. Se han repetido en las mismas condiciones los experimentos realizados con el algoritmo AESA (utilizando también intervalos de tolerancia) para estudiar este fenómeno con distintas variantes del algoritmo LAESA. En estos experimentos se han comprobado dos cosas: 1) el aumento exponencial del número de distancias calculadas con respecto a la dimensión no se observa cuando el número de PB utilizado para cada dimensión permanece constante, 2) si se aumenta el número de PB con respecto a la dimensión el número de distancias calculadas aumenta, es decir, el aumento en el número de distancias calculadas está directamente relacionado con el tamaño del conjunto de PB.

La introducción de un nuevo criterio de selección de prototipos, APB, ha permitido demostrar experimentalmente que, para muestras agrupadas, incluso cuando el tamaño del conjunto de PB elegido es elevado, el número medio de distancias calculadas no aumenta con la dimensión.

Algoritmo LAESA

Entrada:
$P \subset E$ // *subconjunto finito de prototipos*
$B \subset P$ //*conjunto de PB*
$M \in \mathbb{R}^{|P|\times|B|}$ // *matriz precalculada de* $|P| \times |B|$ *distancias*
$x \in E$ // *muestra*
$r \in \mathbb{R}$ // *parámetro para el criterio de selección*
Salida:
$min \in P$ // *prototipo de P más cercano a la muestra*
$D_{\min} \in \mathbb{R}$ // *distancia del prototipo más cercano a la muestra*
Funciones:
$d : E \times E \to \mathbb{R}$ // *función distancia*
CONDICIÓN: Booleano // *criterios de eliminación de PB*
SELECCIÓN$_r : B \times (P - B) \to P$ // *criterios de selección de PB*
Variables:
p, q, s, $b \in P$
$g \in \mathbb{R}^m$ // *vector para la cota inferior*
d_{xs}, g_p, g_q, $g_b \in \mathbb{R}$
Método:
$D_{\min} = \infty; min =$ indeterminado;
$g = \{0, \ldots, 0\}$
$s =$ elemento_aleatorio(B);

mientras $|P| > 0$ **hacer**
$dxs = d(x, s); P = P - \{s\}$; // *cálculo de la distancia*
si $dxs < D_{\min}$ **entonces** //*actualiza mejor solución*
$min = s; D_{\min} = d_{xs}$;
fin si
$q = b =$ indeterminado; $g_q = g_b = \infty$;
para todo $p \in P$ **hacer** // *bucle de aprox. y eliminación*
si $s \in B$ **entonces**
$g[p] = \max(g[p], |dxs - M(s, p)|)$;
fin si
$g_p = g[p]$;
si $p \in B$ **entonces**
si $g_p \geq D_{\min} \wedge$ CONDICIÓN **entonces**
$P = P - \{p\}$; //*eliminar en B*
si no // *aproximación: seleccionar en B*
si $g_p < g_b$ **entonces** $g_b = g_p; b = p$ **fin si**
fin si
si no
si $g_p \geq D_{\min}$ **entonces** $P = P - \{p\}$ //*eliminar en P-B*
si no // *aproximación: seleccionar en P-B*
si $g_p < g_q$ **entonces** $g_q = g_p; q = p$ **fin si**
fin si
fin si
fin para todo
$s =$ SELECCIÓN$_r(b, q)$;
fin mientras
fin Método
fin Algoritmo

Figura 5.9: Algoritmo LAESA, versión parametrizada.

Capítulo 6

Reducción del coste computacional no asociado al cálculo de distancias

El comportamiento de las técnicas de búsqueda de vecinos más próximos en espacios métricos depende directamente del tipo de estructura de datos utilizada para almacenar los prototipos. La mayoría de las técnicas cuyo coste computacional es sublineal utilizan una estructura arborescente para almacenar los prototipos sobre la que se aplica una técnica de ramificación y poda [HS, 78]. Esta técnica se utiliza para realizar una búsqueda en un árbol sobre el que se ha hecho una descomposición jerárquica del conjunto de prototipos. Esto permite evitar calcular la distancia a todos los prototipos. Utilizando reglas apropiadas, se decide si el vecino más próximo puede pertenecer o no a un nodo determinado. Si no es el caso, el nodo puede ser eliminado y con él todos los prototipos que pertenecen al mismo. La eficiencia de los algoritmos basados en esta técnica depende del tipo de descomposición jerárquica utilizada y de las reglas utilizadas para la eliminación de nodos. Un ejemplo lo constituye el algoritmo propuesto por Fukunaga y Narendra (ver sección 1.2.4) [FN, 75] en el que se utilizan dos reglas de eliminación, una para los nodos internos del árbol y otra para los nodos externos u hojas. La ramificación consiste en la incorporación a una lista de nodos vivos de los hijos del nodo que se va a expandir. Este algoritmo fue posteriormente mejorado marginalmente por Kamgar-Parsi y Kanal [KPK, 85] con la adición de dos reglas de eliminación.

Esta es una forma inmediata de reducir el coste computacional no asociado al coste de distancias u *overhead*, ya que se pueden dejar prototipos sin consultar en la fase de búsqueda si previamente se ha producido la poda de la rama que los contenía. El k-dtree es una estructura de datos arborescente

sobre la que, con ciertas modificaciones en su estructura original (comentadas en el capítulo 1), Friedman et al. en 1977 demostraron formalmente que el coste computacional asociado a la búsqueda del vecino más próximo es logarítmico respecto al tamaño del conjunto de prototipos, y además el número de distancias calculadas está acotado por una constante [FBF, 77]. Este es el mejor resultado obtenido para el coste computacional de un algoritmo de búsqueda de vecinos sobre el que el número de distancias permanece constante. Sin embargo, desafortunadamente la estructura de datos sobre la que se realiza la búsqueda requiere la *representación vectorial* de los datos.

En este capítulo se presenta una serie de algoritmos que hacen uso sólamente de las *propiedades métricas* del espacio (sin requerir una representación vectorial) y que utilizan estructuras arborescentes sobre las que se realiza la búsqueda. Se intenta reducir de esta manera el coste computacional lineal del algoritmo LAESA. Estos algoritmos utilizan, al igual que el algoritmo LAESA, un subconjunto de prototipos del conjunto total P, denominado *conjunto de prototipos base*, B, respecto del cual se calcula y se almacena la distancia al resto de prototipos de P. Además de reducir el coste computacional, se pretende que el número de distancias permanezca constante con el tamaño de P, como ocurría con el LAESA.

Los algoritmos presentados requieren un preproceso algo más complejo que el del LAESA. Se utilizan dos tipos de estructuras: un árbol binario T donde se almacenan todos los prototipos, y una matriz de distancias M. El algoritmo de búsqueda realiza un recorrido a través del árbol; la matriz de distancias es utilizada para poder evitar la exploración de algunas ramas del árbol.

6.1. El preproceso

La matriz de distancias M contiene las distancias calculadas entre cada prototipo perteneciente al conjunto de prototipos P y el conjunto de prototipos base (elegido utilizando en principio cualquiera de las estrategias propuestas en el capítulo 3). Para los experimentos con los nuevos algoritmos, se hará uso de las técnicas de selección de PB que mejores resultados presentaban para el algoritmo LAESA: SMD y MDM (ver sección 3.2).

El árbol de prototipos T se obtiene a partir de una partición recursiva y binaria del conjunto de prototipos P. Cada nodo $t \in T$ representa un subconjunto $S_t \subset P$. Si t es un nodo hoja, entonces t contiene solamente un prototipo. Cada nodo t almacena un prototipo representante $m_t \in S_t$ y el radio r_t de S_t. El radio se define como:

$$r_t = \max_{p \in S_t} d(p, m_t) \tag{6.1}$$

En el caso particular en que t es un nodo hoja, $S_t = \{m_t\}$ y por lo tanto $r_t = 0$.

El árbol de prototipos T es construido recursivamente de la siguiente manera: la raíz (ρ) representa $S_\rho = P$, y m_ρ es elegido aleatoriamente entre todos los prototipos pertenecientes a P. El representante del hijo derecho t_r de un nodo t es $m_{t_r} = m_t$ (ver figura 6.1). Sin embargo, para el hijo izquierdo, t_l, se elige el prototipo más alejado en S_t:

$$m_{t_l} = \text{argmax}_{p \in S_t} d(p, m_t) \tag{6.2}$$

Además, $S_{t_r} = \{p \in S_t : d(p, m_{t_r}) < d(p, m_{t_l})\}$ y $S_{t_l} = S_t - S_{t_r}$. El proceso recursivo termina en las hojas que representan subconjuntos con un único prototipo. Existen otros métodos para construir el árbol [DH, 73] pero con un mayor coste temporal.

6.2. Algoritmos de búsqueda

Antes de comenzar el proceso de búsqueda del vecino más próximo a una muestra x a través del árbol T, en los algoritmos que se describirán se calcula en primer lugar la distancia de la muestra a clasificar a todos los PB. Estas distancias son almacenadas para ser utilizadas posteriormente en las fases de aproximación y eliminación de los algoritmos.

Para realizar reducir el número de distancias calculadas, se va a utilizar la función cota inferior de la distancia, $g : P \times P \to \mathbb{R}$ definida como:

$$g(x, y) = \max_{b \in B} \{|d(y, b) - d(x, b)|\} \tag{6.3}$$

y que ya se utilizó con el mismo fin en el algoritmo LAESA.

La función $g(x, m_t)$ será calculada en los algoritmos de búsqueda a partir de:

- las distancias $d(b, m_t)$, calculadas en la fase de preproceso (cuando los prototipos base son seleccionados) y almacenadas en M, y
- las distancias del tipo $d(x, b)$, que no dependen de la partición que se realiza en la construcción del árbol y por lo tanto pueden ser calculadas y almacenadas en un vector D antes de comenzar la búsqueda en el árbol T.

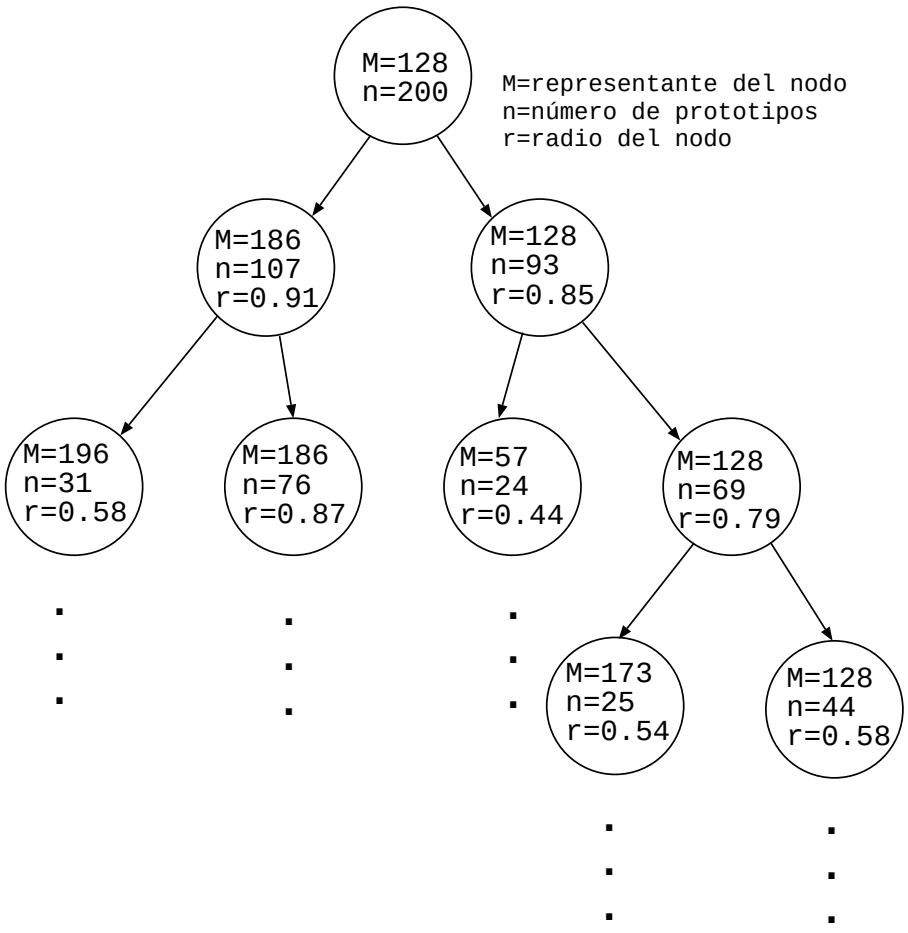

Figura 6.1: Ejemplo de creación de un árbol binario para almacenar 200 prototipos en dimensión 2.

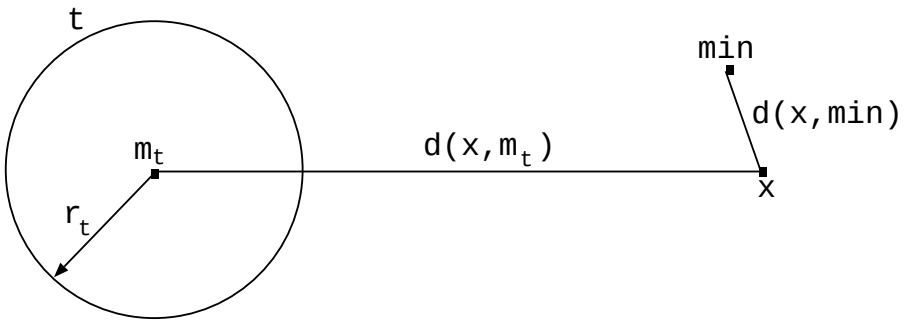

Figura 6.2: Interpretación geométrica de la regla de eliminación.

A continuación, se describe cómo se realiza la poda o eliminación en el árbol T. Sea x la muestra, t un nodo en T y min el prototipo candidato a vecino más próximo. No es difícil comprobar (ver figura 6.2) que si se cumple

$$d(x, min) + r_t < d(x, m_t) \tag{6.4}$$

entonces $d(x, p) > d(x, min)\ \forall p \in S_t$. En este caso, ningún prototipo que pertenezca a S_t está más cercano a x que min y, por lo tanto, la rama t puede ser podada.

Por la desigualdad triangular se cumple que $g(x, y) \leq d(x, y)\ \forall x, y \in P$, por lo que la ecuación 6.4 queda como:

$$d(x, min) + r_t < g(x, m_t) \tag{6.5}$$

y por lo tanto $d(x, p) > d(x, min)\ \forall p \in S_t$

Se han estudiado dos algoritmos: uno iterativo y otro recursivo. El proceso de búsqueda de los algoritmos es su diferencia principal. Concretamente, la estrategia de búsqueda utilizada en el algoritmo iterativo es la de *primero el mejor*, mientras que en el recursivo es una mezcla de *primero el mejor* y *primero en profundidad* (en las próximas secciones se comentarán con mayor detalle). En ambos casos solo se calculan distancias cuando se llega a las hojas del árbol.

Algoritmo TLAESA
Entrada:
T // *árbol de prototipos con raíz* ρ
$B \subseteq P$ //*conjunto de PB*
$M \in \mathbb{R}^{|B|\times|P|}$ //*matriz de distancias*
$x \in E$ // *muestra*
Salida:
$min \in P, D_{\min} \in \mathbb{R}$
Funciones:
$g : E \times E \to \mathbb{R}$ // *función cota inferior de la distancia*
$d : E \times E \to \mathbb{R}$ // *función distancia*
Variables: $D \in \mathbb{R}^{|B|}, g_\rho \in \mathbb{R}, b \in B, \rho \in T$;
Método:
$D_{\min} = \infty, g_\rho = 0$;
para todo $b \in B$ **hacer**
$D[b] = d(x, b)$;
si $D[b] < D_{\min}$ **entonces**
$D_{\min} = D[b]$;
$min = b$;
fin si
$g_\rho = \max(g_\rho, |M(b, \rho) - D[b]|)$;
fin para todo
BÚSQUEDA (ρ, g_ρ);
fin Método
fin Algoritmo

Figura 6.3: Algoritmo TLAESA. La búsqueda tiene dos versiones: iterativa y recursiva.

6.2.1. Algoritmo iterativo de búsqueda

En este primer método, la exploración del árbol T comienza en la raíz, ρ, colocando sus hijos ρ_t y ρ_r en una *lista de nodos vivos*, l_{nv}, inicialmente vacía. Este método es un procedimiento iterativo. Si el nodo que se está examinando en una iteración determinada, q, es una hoja, entonces si se cumple que $g(x, q) < D_{\min}$ se calcula la distancia a x. En este caso, si $d(x, q) < D_{\min}$ se actualiza min (prototipo más cercano hasta el momento) con el valor de q.

Si q no es una hoja, se calcula $g(x, m_{t_l})$ (no es necesario calcular $g(x, m_{t_r})$ ya que $m_{t_r} = m_t$) poniendo los dos hijos en la lista l_{nv} si $g(x, m_{t_i}) < D_{\min}$. Entre aquellos nodos que no son eliminados se elige un candidato a explorar

en la siguiente iteración según la fórmula:

$$q = \text{argmin}_{\forall t \in l_{\text{nv}}} g(x, t) \tag{6.6}$$

El procedimiento finaliza cuando la lista l_{nv} está vacía (ver figura 6.4). Esta lista se implementa eficientemente mediante una cola de prioridad (*heap*) sobre los valores de $g(x, .)$, y la estrategia de ramificación y poda resultante es la llamada *primero el mejor*. En el algoritmo iterativo de búsqueda se utilizan las siguientes funciones:

- eshoja: tiene como parámetro de entrada un nodo y como parámetro de salida un valor booleano. Esta función devuelve un valor verdadero cuando el nodo es un nodo hoja.
- hijoizquierdo(hijoderecho): tiene como parámetro de entrada y salida un nodo. Esta función devuelve el nodo sucesor izquierdo (derecho) de un nodo que pertenece a un árbol binario.

6.2.2. Algoritmo recursivo de búsqueda

La búsqueda recursiva sigue una estrategia *primero en profundidad* combinada con un recorrido *raíz–izquierda–derecha* y *raíz–derecha–izquierda* dependiendo de los valores de la función g para los hijos del nodo t, $g(x, m_{t_r})$ y $g(x, m_{t_l})$. El hijo con el menor valor de g es el primero de los dos que se explora (excepto si es eliminado). Cuando el nodo explorado q es una hoja, si se cumple que $g(x, q) < D_{\min}$ se calcula $d(x, q)$. Si, para este último caso, se cumple que $d(x, q) < D_{\min}$, q será el nuevo vecino más cercano a la muestra hasta el momento [MOC, 95] (ver figura 6.5).

6.3. Experimentos

Para estudiar el comportamiento de ambos algoritmos se han realizado experimentos con datos sintéticos. En estos experimentos se ha utilizado la métrica euclídea sobre un conjunto de datos uniformemente distribuido sobre el hipercubo unidad. Para estudiar el coste computacional de los algoritmos se contabilizó el número de *pasos de programa*[1] que se producían en el proceso de búsqueda.

El primer experimento fue realizado para obtener el valor del único parámetro que aparece en el algoritmo, el número de PB, $|B|$. Los resultados que

[1]Paso de programa es una instrucción simple en el programa

Algoritmo BÚSQUEDA
Entrada:
$t \in T$ // *nodo del árbol*
$gt \in \mathbb{R}$ //*valor de g(x,t)*
Variables: $t_r, t_l, n \in nodo; d, g_l, gaux \in \mathbb{R}$;
Método:
Repetir
si eshoja(t) **entonces**
si $g_t < D_{\min}$ **entonces**
$d = d(x, m_t)$; //*cálculo de la distancia*
si $d < D_{\min}$ **entonces** // *actualiza mejor solución*
$min = m_t$;
$D_{\min} = d$;
fin si
fin si
si no
$t_r = \text{hijoderecho}(t)$;
$t_l = \text{hijoizquierdo}(t)$;
$g_l = g(x, t_l)$;
si $D_{\min} + r_l > g_l$ **entonces** //*expansión del hijo izquierdo*
Añadir(l_{nv}, t_l);
fin si
si $D_{\min} + r_r > g_t$ **entonces** //*expansión del hijo derecho*
Añadir(l_{nv}, t_r);
fin si
$gaux = \infty$;
para todo $n \in l_{\text{nv}}$ **hacer**
si $g_n < gaux$ **entonces**
$t = n$;
$gaux = g_n$;
fin si
fin para todo
Borrar(l_{nv}, t);
fin si
fin repetir (hasta $l_{\text{nv}} = \emptyset$)
fin Método
fin Algoritmo

Figura 6.4: Algoritmo BÚSQUEDA en su versión iterativa.

Algoritmo BÚSQUEDA
Entrada:
$t \in T$ // *nodo del árbol*
$g_t \in \mathbb{R}$ //*valor de g(x,t)*
Variables: $t_r, t_l \in$ nodo; $d \in \mathbb{R}$;
Método:
si eshoja(t) **entonces**
si $g_t < D_{\min}$ **entonces** //*cálculo de la distancia*
$d = d(x, m_t)$;
si $d < D_{\min}$ **entonces** //*actualiza de la mejor solución*
$min = m_t$;
$D_{\min} = d$;
fin si
fin si
si no
$t_r =$ hijoderecho(t);
$t_l =$ hijoizquierdo(t);
$g_l = g(x, t_l)$;
si $g_l \leq g_t$ **entonces** //*primera rama a explorar*
si $D_{\min} + r_l > g_l$ **entonces**
BÚSQUEDA (t_l, g_l);
fin si
si $D_{\min} + r_r > g_t$ **entonces**
BÚSQUEDA (t_r, g_t);
fin si
si no
si $D_{\min} + r_r > g_t$ **entonces**
BÚSQUEDA (t_r, g_t);
fin si
si $D_{\min} + r_l > g_l$ **entonces**
BÚSQUEDA (t_l, g_l);
fin si
fin si
fin si
fin Método
fin Algoritmo

Figura 6.5: Algoritmo BÚSQUEDA en su versión recursiva.

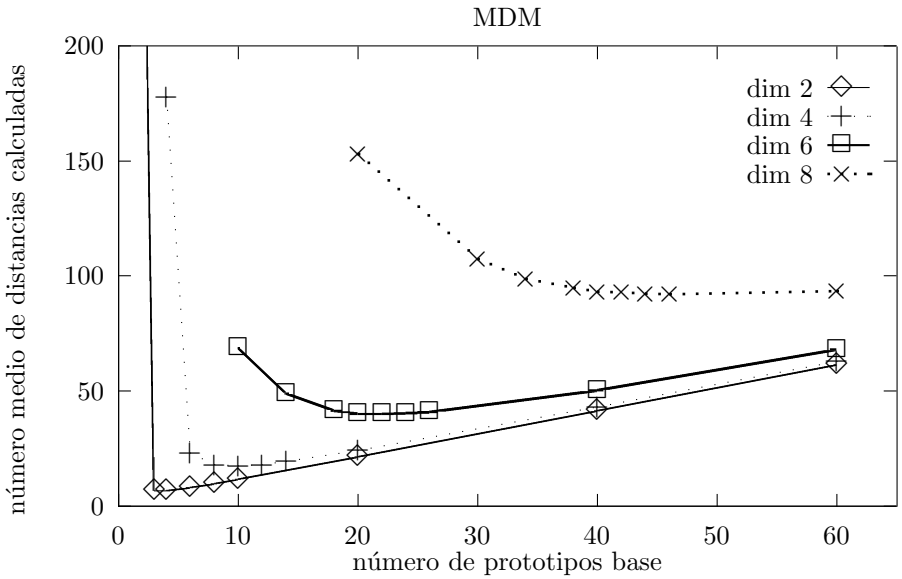

Figura 6.6: Número medio de distancias calculadas por el algoritmo (recursivo) en función del número de PB y diferentes dimensiones, usando la métrica euclídea y un conjunto de 8192 prototipos. El método de selección de PB fue el MDM. La desviación típica media de todas las estimaciones fue del 2 %.

se muestran en la figura 6.6 son los obtenidos con el algoritmo recursivo (los resultados son similaares a los obtenidos con el algoritmo iterativo). Este experimento se hizo utilizando los métodos de selección de prototipos base SMD (suma máxima de distancias) y MDM (máximo de las distancias mínimas). Como en capítulos anteriores, el método MDM daba unos resultados algo mejores (en número de distancias) que el SMD (alrededor del 10 % mejores). Este experimento se hizo para varias dimensiones en un espacio vectorial euclídeo (2, 4, 6, 8) con un conjunto de 8192 prototipos.

Los resultados de la figura 6.6 muestran que existe un valor óptimo para $|B|$ para el cual el número medio de distancias calculadas es mínimo. Cuando $|B|$ es muy bajo, el hecho de añadir más PB hace que la fase de eliminación en el algoritmo sea más efectiva, lo que reduce aún más el número de distancias a calcular. Sin embargo, cuando $|B|$ tiene un valor suficientemente grande, se produce un aumento lineal del número de distancias a calcular con $|B|$. Esto es debido a que en el algoritmo se calculan las distancias de todos los PB a la muestra. Como se puede ver en la figura 6.6 el número óptimo de PB depende de la dimensión del espacio. Concretamente, los valores óptimos $|B|^*$ para dimensiones 2, 4, 6 y 8 (usando la distancia euclídea) son 3, 10, 22

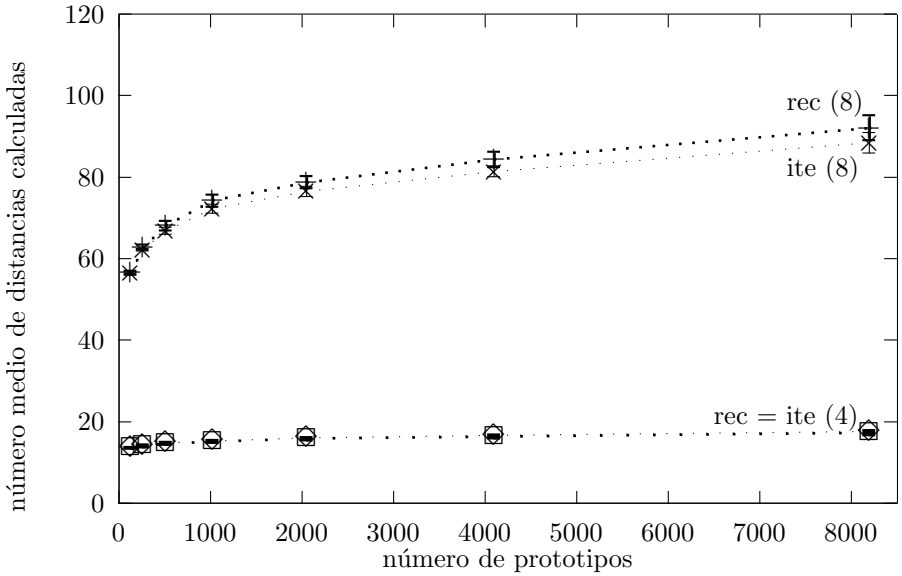

Figura 6.7: Número medio de distancias para los valores óptimos de $|B|$ de los algoritmos recursivo e iterativo en función del número de prototipos en dimensión 4 y 8, usando la distancia euclídea.

y 46, respectivamente. También se ha comprobado que, al igual que ocurría para el algoritmo LAESA, el valor óptimo ($|B|^*$) para cada dimensión no depende del tamaño del conjunto de prototipos.

Una vez obtenidos los valores óptimos del conjunto de PB, se ha realizado un experimento para ver qué ocurre con el número de distancias calculadas al aumentar el tamaño ($|P|$) del conjunto de prototipos. Este experimento ha sido realizado utilizando los dos algoritmos propuestos para diferentes dimensiones. Los resultados con ambos algoritmos se pueden ver en la figura 6.7, donde se puede comprobar que, además de que los resultados son prácticamente idénticos, el número de distancias a calcular tiende a ser independiente del número de prototipos. A su vez, el comportamiento del algoritmo recursivo medido en número medio de pasos de programa para calcular el coste computacional era mejor que en el iterativo (ver figura 6.8). Por ello, el resto de experimentos se han realizado utilizando únicamente el algoritmo recursivo.

Para demostrar que el número de distancias tiende a una constante, se ha repetido el experimento utilizando el k-dtree (ver sección 1.2.3). La comparación entre el número de distancias calculadas por el TLAESA (recursivo) y el k-dtree se presenta en la figura 6.9.

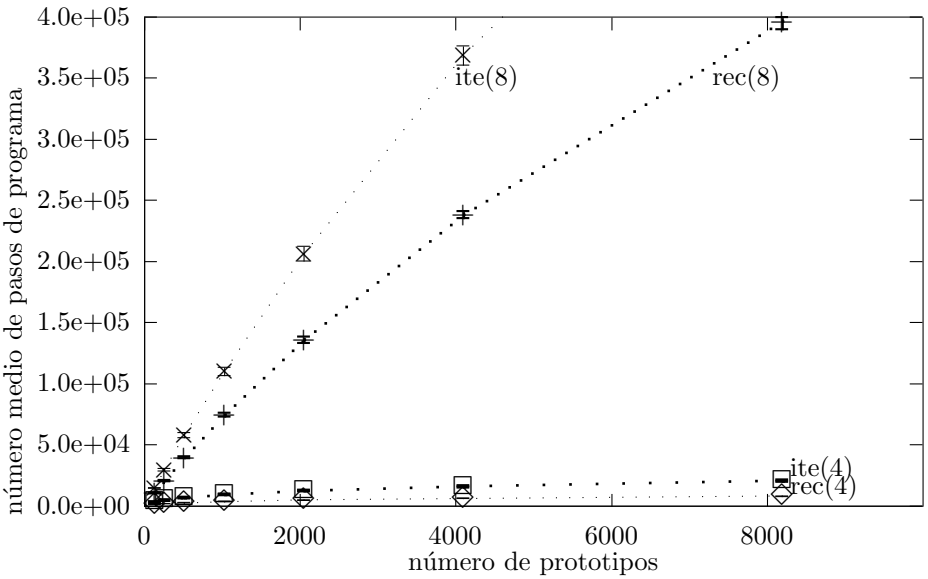

Figura 6.8: Número medio de pasos de programa para los valores óptimos de $|B|$ de los algoritmos de búsqueda recursivo e iterativo en función del número de prototipos en dimensión 4 y 8, usando la distancia euclídea.

Friedman et al. demuestran en [FBF, 77] que el número de distancias que se requieren para encontrar el vecino más próximo en un k-dtree está limitado por una constante que depende de la dimensión del espacio. Como se puede ver en la figura 6.9 la relación disminuye cuando aumenta el número de prototipos, lo que permite afirmar que el número de distancias calculadas en el TLAESA deberá estar también limitado superiormente por una constante.

Obviamente, el algoritmo TLAESA no puede hacer uso de la información de las distancias entre los prototipos de la matriz de distancias de forma tan eficiente como se realiza con el algoritmo LAESA (debido a que la aproximación y eliminación se realizan en función de los representantes de cada nodo, y no para cualquier prototipo), y por lo tanto calcula más distancias que éste (ver figura 6.10). Sin embargo, el coste computacional del algoritmo TLAESA es sublineal, como se puede comprobar en las figuras 6.11 y 6.12. La elección entre los dos algoritmos depende del tamaño del conjunto de prototipos y del coste de la distancia. Si los conjuntos de prototipos son de tamaño mediano o las distancias utilizadas son muy caras, el algoritmo LAESA será preferible al TLAESA. Pero a medida que el tamaño del conjunto de prototipos empiece a ser elevado, será preferible elegir el algoritmo TLAESA. En los experimentos realizados en las figuras 6.11 y 6.12 se contabilizan los

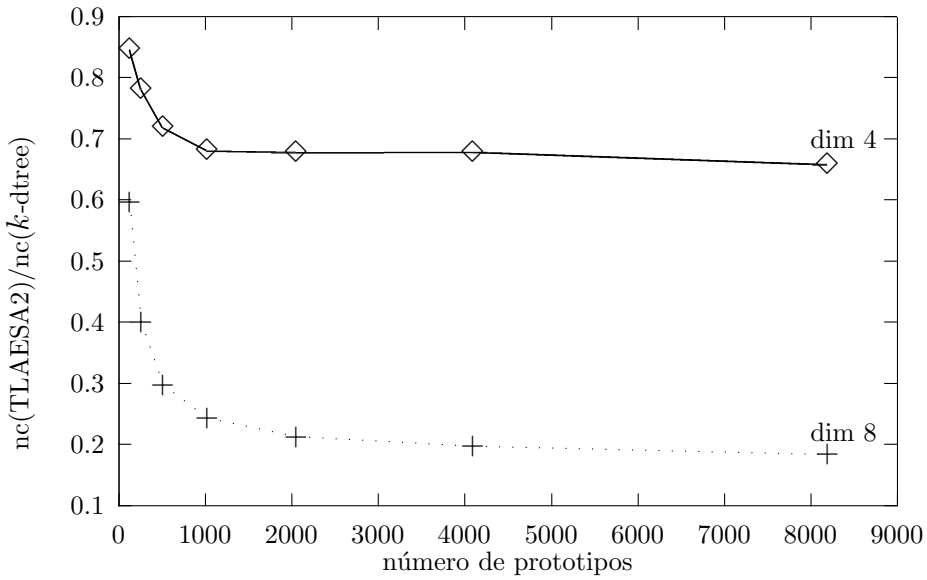

Figura 6.9: Cada una de las curvas representa, para una dimensión dada, el cociente entre el número de distancias (nc) calculadas con el algoritmo TLAESA y el número de distancias calculadas usando el k-dtree. El número de PB utilizado en cada dimensión ha sido el óptimo, 10 PB en dimensión 4 y 46 PB en dimensión 8.

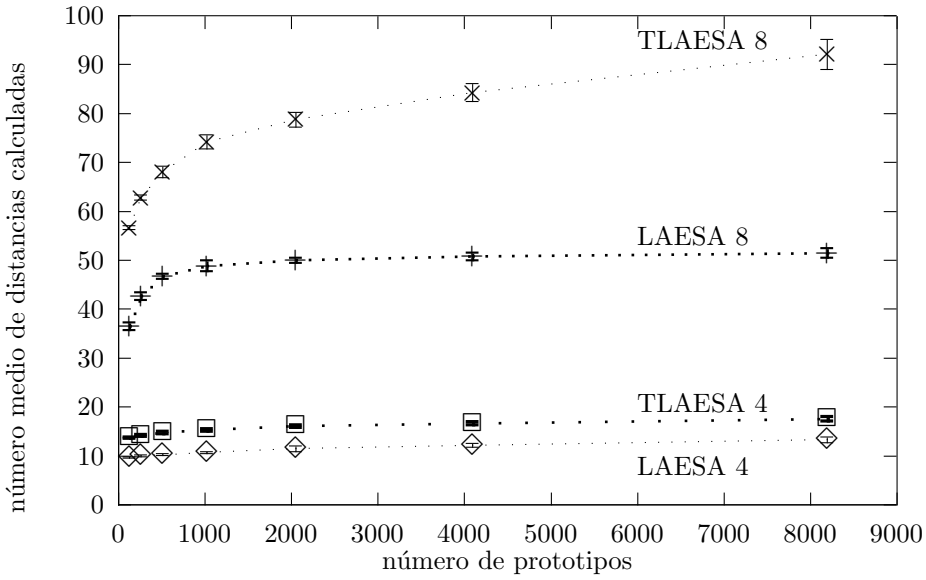

Figura 6.10: Número medio de distancias calculadas por los algoritmos LAESA y TLAESA para dimensiones 4 y 8 en un espacio vectorial euclídeo. El tamaño del conjunto de PB fue el óptimo en cada caso. Para el LAESA $|B|^*$=7 en dimensión 4 y $|B|^*$=30 en dimensión 8. Para el TLAESA, $|B|^*$=10 en dimensión 4 y $|B|^*$=46 en dimensión 8.

pasos de programa que necesita cada algoritmo para encontrar el vecino más próximo. En estos experimentos se estudia la influencia de dos características importantes de funciones distancia definidas en una situación de espacio métrico, no necesariamente vectorial: 1) coste computacional de la distancia y 2) propiedades geométricas que vienen caracterizadas por la dimensión. En los experimentos se ha asociado un coste determinado al cálculo de la distancia medido en pasos de programa, y la dimensión simula la complejidad estructural del espacio de búsqueda ("dimensión intrínseca").

En las gráficas de las figuras 6.11 y 6.12, se puede ver que las cuatro gráficas representan el coste total[2] del algoritmo en dos dimensiones distintas (4 y 8) de los prototipos en un espacio vectorial euclídeo. El coste en pasos de programa asociado a la distancia ha sido 1, 100, 1000, 10000, respectivamente.

Para dimensión 4, el coste en pasos de programa es claramente menor con el nuevo algoritmo que con el LAESA y este valor es, además, independiente del tamaño del conjunto de prototipos. Cuando el coste de la distancia es muy elevado ($D = 10000$ pasos), el TLAESA empieza a ser mejor que el LAESA para tamaños del conjunto de prototipos superiores a 1000. Esto significa que a medida que aumenta el coste asociado a la distancia para una dimensión determinada, el tamaño del conjunto de prototipos para el cual es mejor el TLAESA que el LAESA es cada vez mayor.

En dimensión 8 también se observa un comportamiento similar al de dimensión 4, aunque en este caso el tamaño del conjunto de prototipos para el que el algoritmo TLAESA empieza a ser mejor que el LAESA es superior que en dimensión 4. Concretamente para costes de distancia muy elevados ($D = 10000$ pasos) es mejor el algoritmo TLAESA con conjuntos superiores a 3500 prototipos.

Se puede deducir entonces que para tamaños pequeños/medianos del conjunto de prototipos es mejor utilizar el algoritmo LAESA, mientras que a medida que estos conjuntos se hacen muy grandes es mejor utilizar el TLAESA.

6.3.1. Experimentos con otros tipos de distribución

El algoritmo TLAESA no ha sido probado únicamente con la técnica de agrupamiento propuesta en la sección 6.1, sino que también se han utilizado otras técnicas, entre ellas la de las c-medias(sección 3.1). Los resultados comparando las dos formas de agrupamiento de los prototipos en el árbol son prácticamente idénticos cuando la distribución de prototipos es uniforme.

[2]El *coste total* se define como el número de pasos de programa (sin contabilizar las distancias) más el número de distancias multiplicado por el coste de la distancia

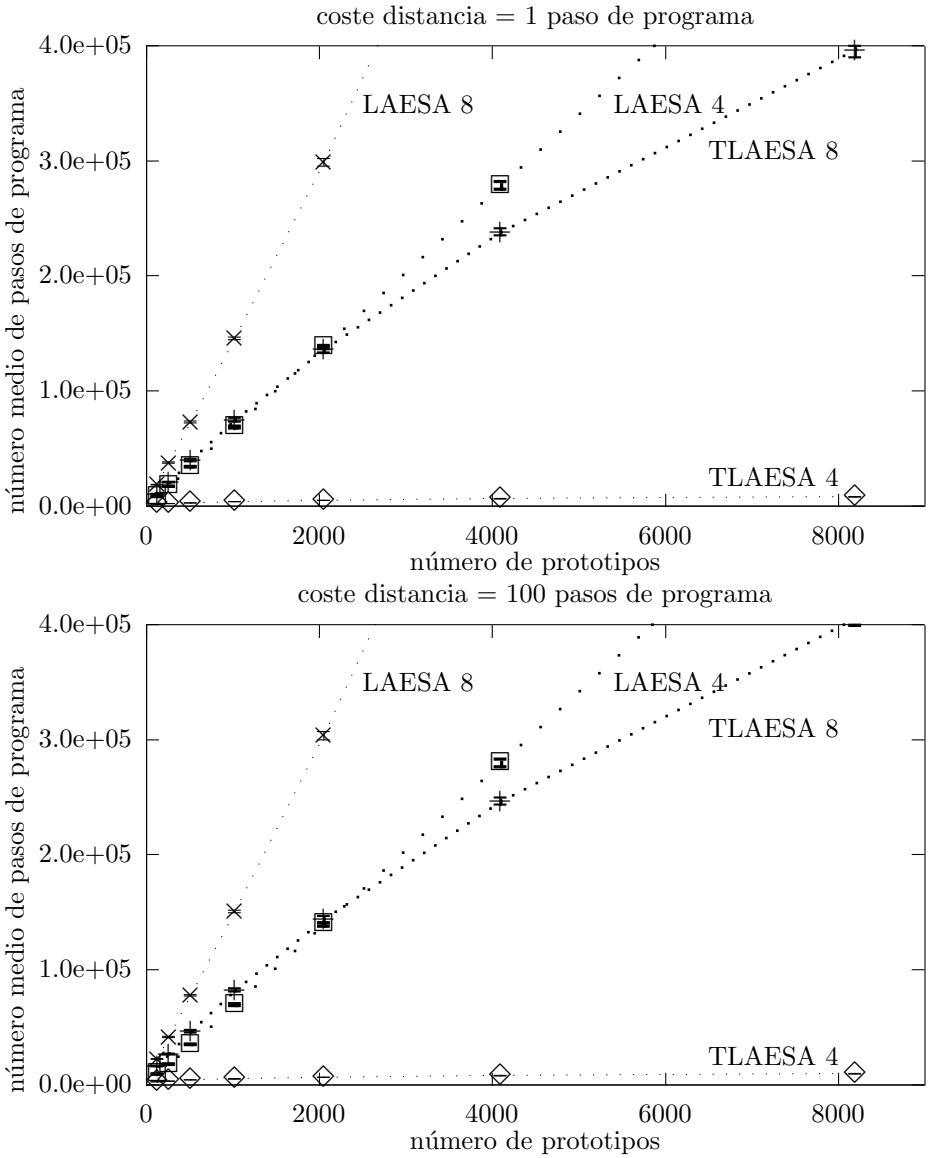

Figura 6.11: Comparación entre el LAESA y TLAESA en número de pasos de programa en un espacio vectorial euclídeo. El coste asociado al cálculo de la distancia en cada caso es de 1 y 100 pasos de programa, respectivamente.

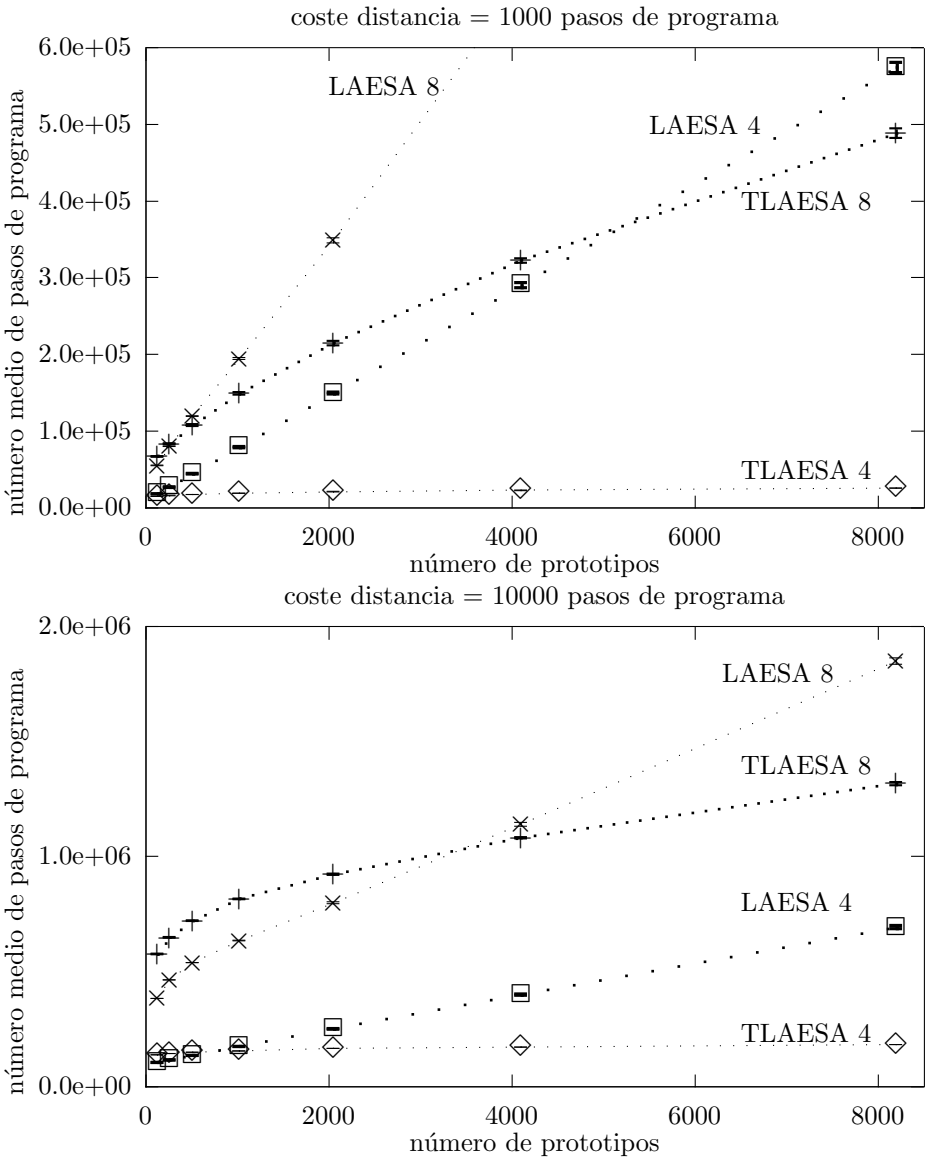

Figura 6.12: Comparación entre el LAESA y TLAESA en número de pasos de programa en un espacio vectorial euclídeo. El coste asociado al cálculo de la distancia en cada caso es de 1000 y 10000 pasos de programa, respectivamente.

Cabe recordar en este punto que por lo tanto es mucho más eficiente utilizar la técnica de agrupamiento propuesta en la sección 6.1, de coste lineal, que la técnica de las c-medias con un coste claramente superior. Este experimento fue repetido utilizando distribuciones gaussianas (igual que se hizo en la sección 4.2.2). En este caso el algoritmo continúa teniendo el mismo comportamiento (ver figura 6.13).

6.4. Conclusiones

El algoritmo TLAESA, como el LAESA, solamente hace uso de las propiedades métricas de la distancia. Además mantiene las otras propiedades interesantes del mismo: calcula un número medio de distancias constante con la talla del conjunto de prototipos y el coste espacial es lineal. Sin embargo con este algoritmo se ha conseguido reducir el *overhead*, y con él el coste computacional total del algoritmo a cotas sublineales, de forma que para un coste asociado a la distancia, el TLAESA se comporta mejor que el LAESA a medida que aumenta el tamaño del conjunto de prototipos.

Se puede resumir la parte II diciendo que los algoritmos LAESA y TLAESA son dos algoritmos rápidos de búsqueda de vecinos más próximos en espacios métricos. Dependiendo del tamaño del conjunto de prototipos y del coste computacional del cómputo de la distancia, será más conveniente utilizar uno u otro método (LAESA para conjuntos pequeños/medianos, TLAESA para conjuntos muy grandes).

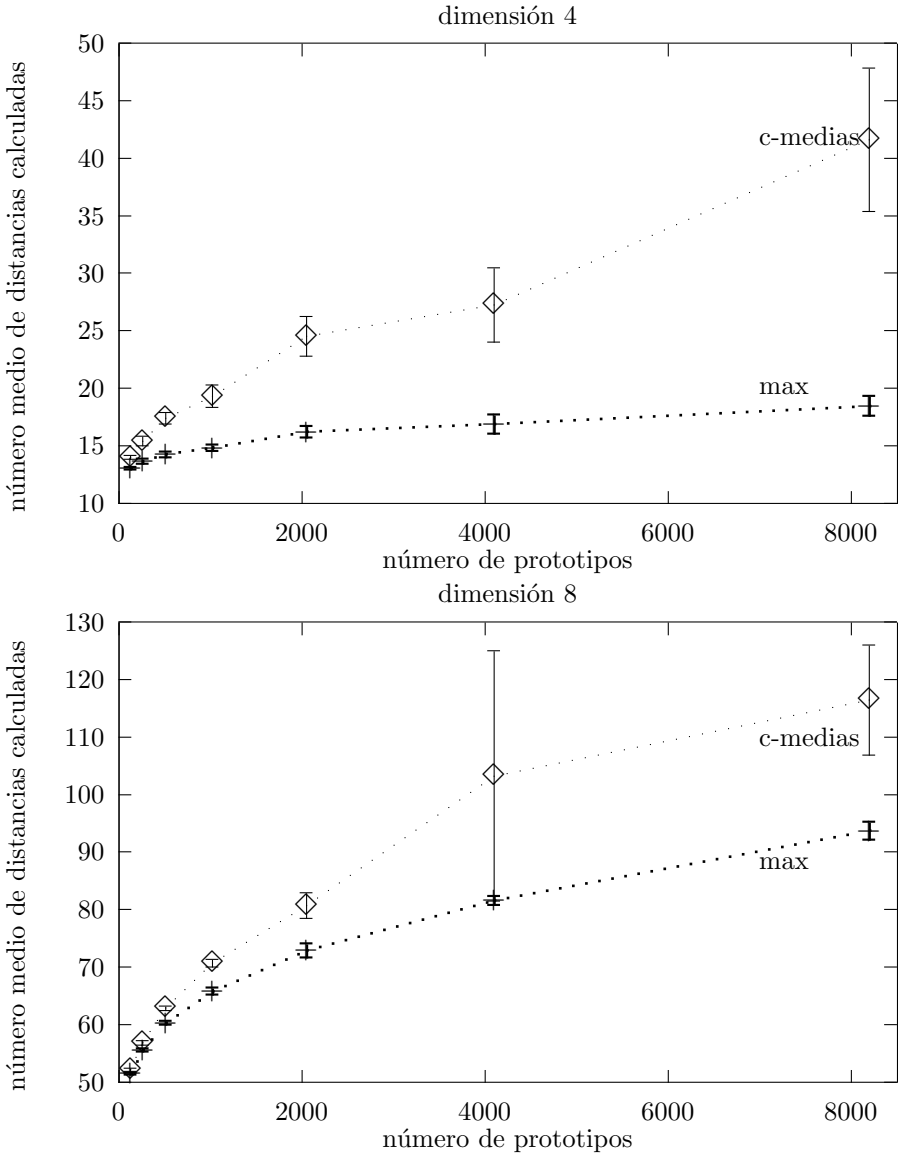

Figura 6.13: Número medio de distancias calculadas por el algoritmo TLAE-SA utilizando el algoritmo *c*-medias para la creación del árbol y prototipos extraídos de distribuciones gaussianas para dimensiones 4 y 8 usando la métrica euclídea. El tamaño del conjunto de PB fue el óptimo en cada caso, $|B|^*$=10 en dimensión 4 y $|B|^*$=46 en dimensión 8.

Parte III

Comparación de métodos y aplicación al reconocimiento de caracteres manuscritos

Capítulo 7

Comparación de algoritmos

En este capítulo se comparan experimentalmente los algoritmos propuestos con otros aparecidos anteriormente en la literatura y que también trabajan en espacios métricos. Esta comparación se realizará en función del coste temporal de los algoritmos. Concretamente, con el algoritmo LAESA (capítulo 2 al 5) se ha reducido el coste espacial del algoritmo AESA manteniendo la característica principal del mismo (el número medio de distancias calculadas es constante en promedio respecto al tamaño del conjunto de prototipos). El coste temporal del LAESA es lineal por lo que la primera comparación será realizada respecto al algoritmo AESA.

En el capítulo 6 se ha pasado a reducir el coste computacional total ("overhead") del LAESA (lineal) con la propuesta de un nuevo algoritmo, el TLAESA. Este algoritmo, basado en la técnica de ramificación y poda, utiliza reglas de eliminación que evitan visitar todos los prototipos del conjunto. Otras técnicas que utilizan el mismo esquema son las propuestas por Fukunaga y Narendra [FN, 75], y Kalantari y MacDonald [KM, 83] con resultados similares a los de Fukunaga y Narendra. La segunda comparación realizada se centra en el estudio del coste temporal del algoritmo propuesto, TLAESA, con los algoritmos que utilizando la misma técnica de programación tienen un coste temporal sublineal.

7.1. Comparación de algoritmos con coste temporal lineal

Como su propio nombre indica, el algoritmo LAESA es una versión *lineal* del AESA en referencia al coste espacial del algoritmo (en el caso del AESA es cuadrático) respecto al tamaño del conjunto de prototipos.

Se puede afirmar que el LAESA con los criterios de eliminación de pro-

dimensión	número de distancias (AESA)
4	7.35
8	27.42

Cuadro 7.1: Número medio de distancias calculadas por el algoritmo AESA en dimensión 4 y 8.

totipos base E_{elim} y E_∞, tiene como un caso especial el AESA en el que el tamaño del conjunto de PB, $|B|$ coincide con el tamaño del conjunto de prototipos, $|P|$.

$$\text{AESA} = lim_{|B|\rightarrow|P|}\text{LAESA}$$

Esto se ha comprobado realizando un experimento en el que se ha ido aumentando el tamaño de B hasta que alcanza el tamaño del conjunto de prototipos P. En el mismo se han utilizado los dos criterios de eliminación de PB que hacían que el número de distancias a calcular fuese independiente de este tamaño a medida que aumentaba el número de PB. El resto de criterios de eliminación no tienen este comportamiento, sino que en éstos el número de distancias aumenta linealmente a media que lo hace el número de PB. Comparando los resultados de la figura 7.1 para el valor de $|B| = 1024$ con la tabla 7.1 donde se muestra el número de distancias calculadas por el AESA para un conjunto de 1024 prototipos, se puede ver que los resultados del LAESA con los criterios E_{elim} y E_∞ son idénticos.

En la figura 7.2 se compara el número medio de distancias calculadas por el AESA y por el LAESA (utilizando los tamaños óptimos del conjunto de PB en cada dimensión, ver sección 3.5.2). Se puede observar que este valor siempre es menor para el AESA que para el LAESA. Para estos resultados los tamaños utilizados para el conjunto de PB en dimensión 4 y 8 fueron, respectivamente, 7 y 24.

Cuando se ha trabajado con datos agrupados para demostrar que el número de distancias a calcular no dependía exponencialmente de la dimensión, ha sido necesario introducir en el LAESA un nuevo criterio de selección de prototipos en la fase de aproximación o mantener constante el número de PB utilizados en cada dimensión (capítulo 5). Sin embargo, en el AESA no es necesario modificar nada ya que todos los prototipos del conjunto P son PB a su vez, y por lo tanto no existe distinción entre prototipos base y prototipos que no son base.

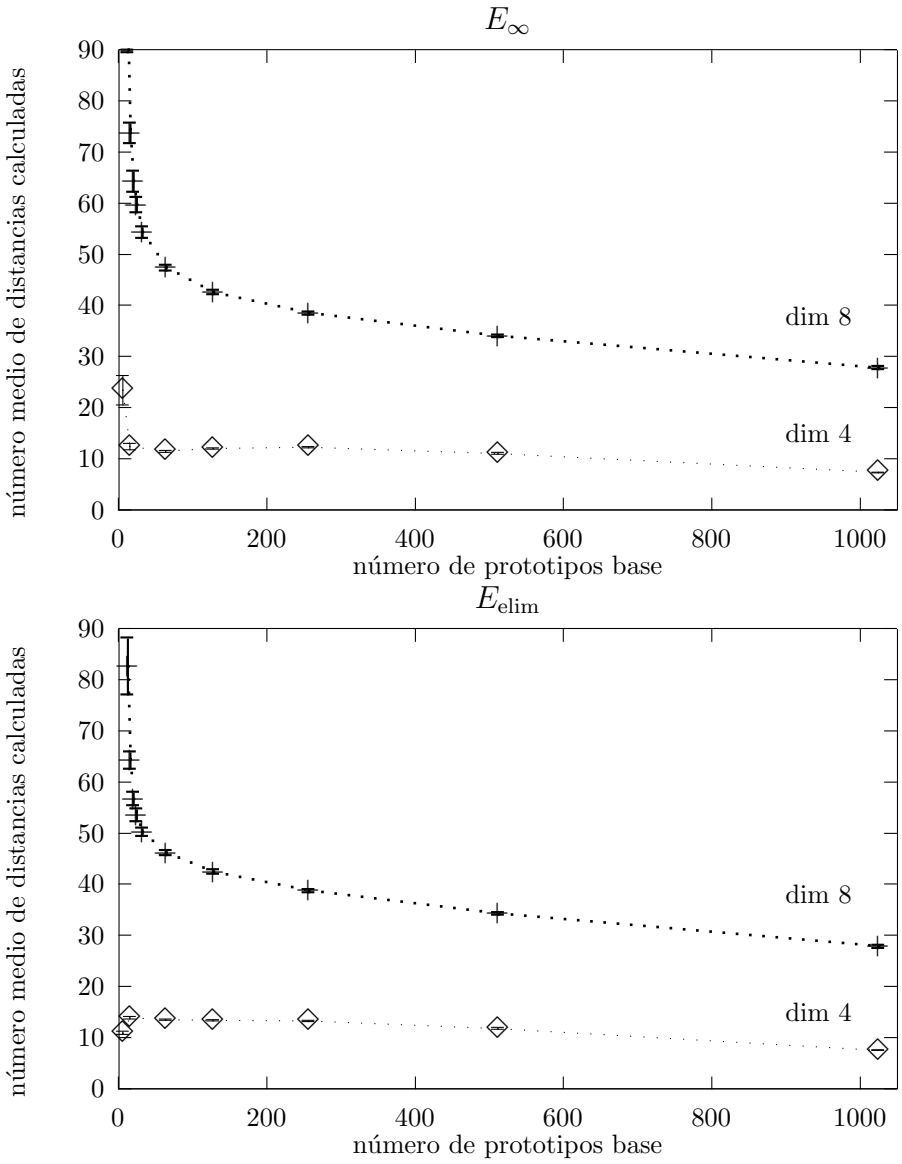

Figura 7.1: Número medio de distancias calculadas por el LAESA cuando aumenta el número de PB hasta que coincide con el tamaño del conjunto total (1024), en espacios vectoriales euclídeos de dimensión 4 y 8, respectivamente.

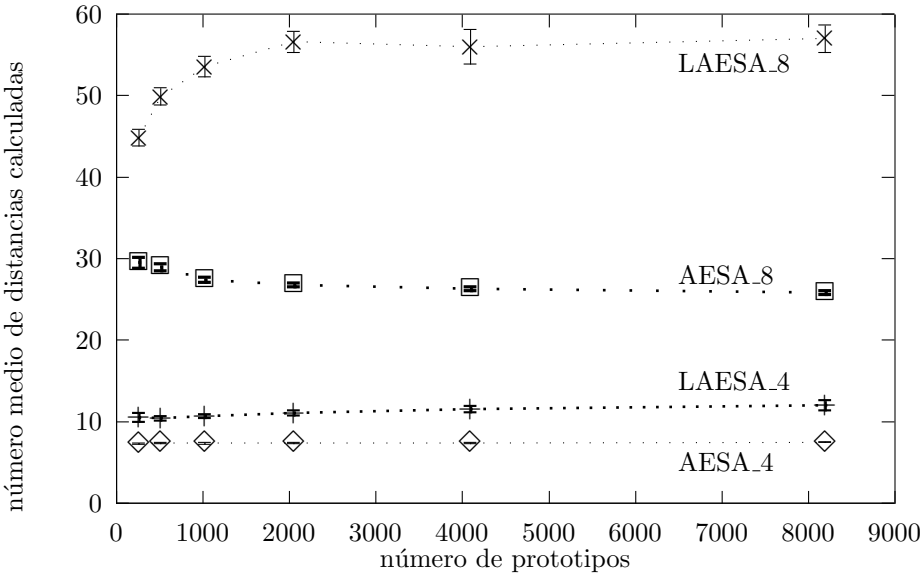

Figura 7.2: Número medio de distancias calculadas por LAESA y AESA cuando se aumenta el número de prototipos en un espacio vectorial euclídeo de dimensión 4 y 8, respectivamente. El criterio de selección de PB ha sido en ambos casos el MDM, y el de eliminación en el LAESA ha sido el E_{elim}.

7.2. Comparación de algoritmos con coste temporal sublineal

El algoritmo TLAESA se ha introducido (capítulo 6) para reducir el coste computacional, no asociado al coste de la distancia, de los algoritmos AESA y LAESA. Para ello, se utiliza una estructura arborescente que permite realizar la búsqueda del vecino más próximo de forma que se evita visitar algunas ramas del árbol. Este tipo de estructura ha sido utilizada en otros métodos anteriormente propuestos, como por ejemplo el algoritmo de Fukunaga y Narendra [FN, 75] y el de Kalantari y McDonald [KM, 83]. Concretamente, en este segundo caso, los autores demostraron experimentalmente que el coste temporal promedio del algoritmo propuesto era $O(\log |P|)$.

En esta sección se compara el algoritmo TLAESA con estos dos algoritmos. Una característica común a los tres es que las reglas de eliminación de nodos utilizadas son las mismas. Las diferencias son principalmente dos: el preproceso y la búsqueda en el árbol. En el preproceso del algoritmo TLAESA se obtiene, además de la estructura arborescente en la que se almacenan los prototipos, una matriz de distancias entre el conjunto de prototipos y el conjunto de PB (previamente seleccionados). Sin embargo, en el algoritmo de

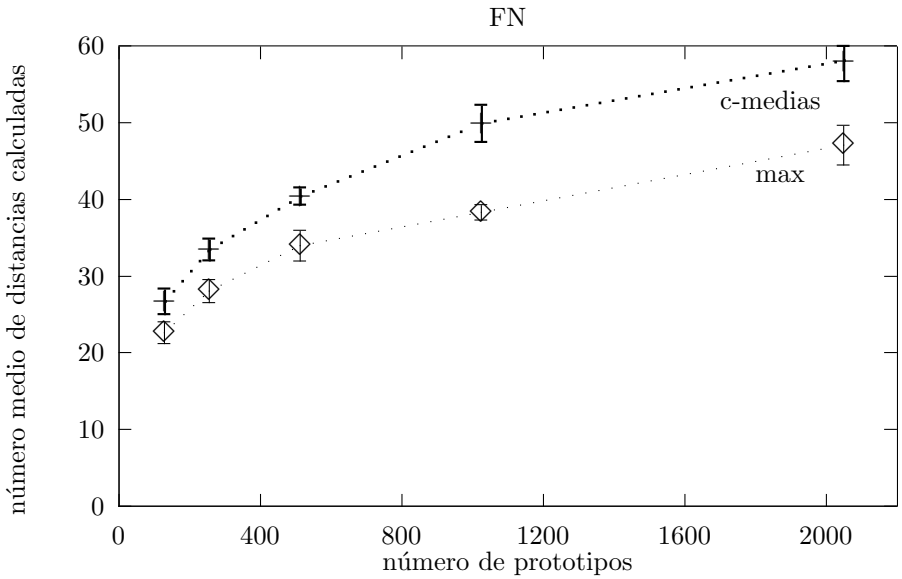

Figura 7.3: Número medio de distancias calculadas por el algoritmo FN (1975), al aumentar el número de prototipos en un espacio vectorial euclídeo de dimensión 2 sobre una distribución uniforme. El algoritmo FN es aplicado a dos tipos de agrupamiento de prototipos: el de las c-medias y el propuesto en el capítulo 6 para el TLAESA.

Fukunaga y Narendra (1975), FN, y en el de Kalantari y McDonald (1983), KM, solamente se utiliza la estructura arborescente. La búsqueda en el árbol es la otra diferencia, ya que cada uno de los algoritmos utiliza una estrategia (o combinación de estrategias) adecuada a la información almacenada en la estructura arborescente. El algoritmo FN ha sido utilizado sobre un árbol binario, con un número de niveles suficiente para que en los nodos hojas no hubiesen más de un elemento (se ha comprobado experimentalmente que esta elección para el número de niveles y la ramificación hace que el algoritmo tenga un comportamiento similar, o mejor, que la utilización de otros valores). A su vez, para el almacenamiento de los prototipos en el árbol, se ha utilizado el mismo criterio que el propuesto para el algoritmo TLAESA (capítulo 6). De hecho, se ha comprobado experimentalmente que la utilización de este criterio (de coste computacional lineal respecto al tamaño del conjunto de prototipos) sobre el algoritmo FN da resultados similares en la fase de búsqueda que si se hubiese utilizado el algoritmo de las c-medias (ver figura 7.3 y 7.4, en las que se ha denominado *max* al criterio de agrupamiento de prototipos visto en la seccin 6.1).

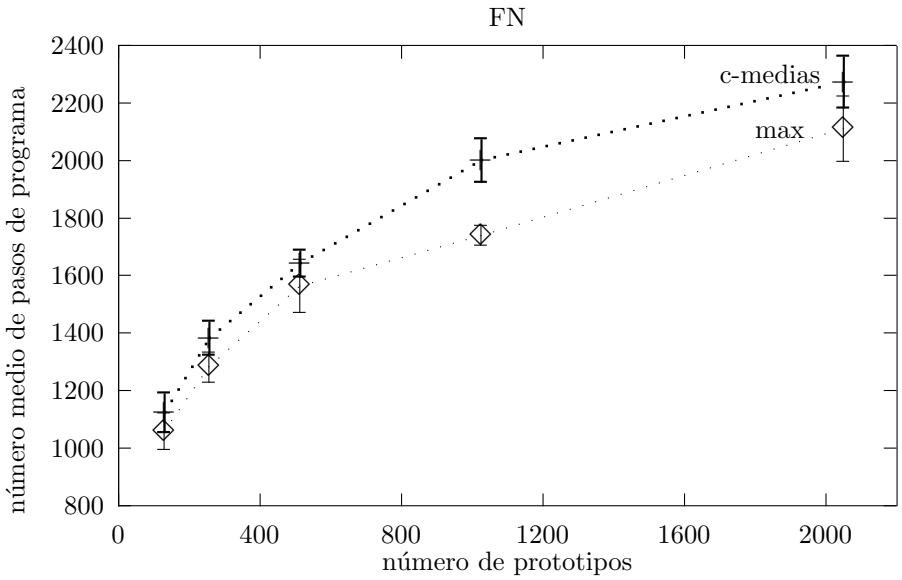

Figura 7.4: Número medio de pasos de programa del algoritmo FN, al aumentar el número de prototipos en un espacio vectorial euclídeo de dimensión 2 sobre una distribución uniforme. El algoritmo FN es aplicado a dos tipos de agrupamiento de prototipos: el de las c-medias y el propuesto en el capítulo 6 para el TLAESA.

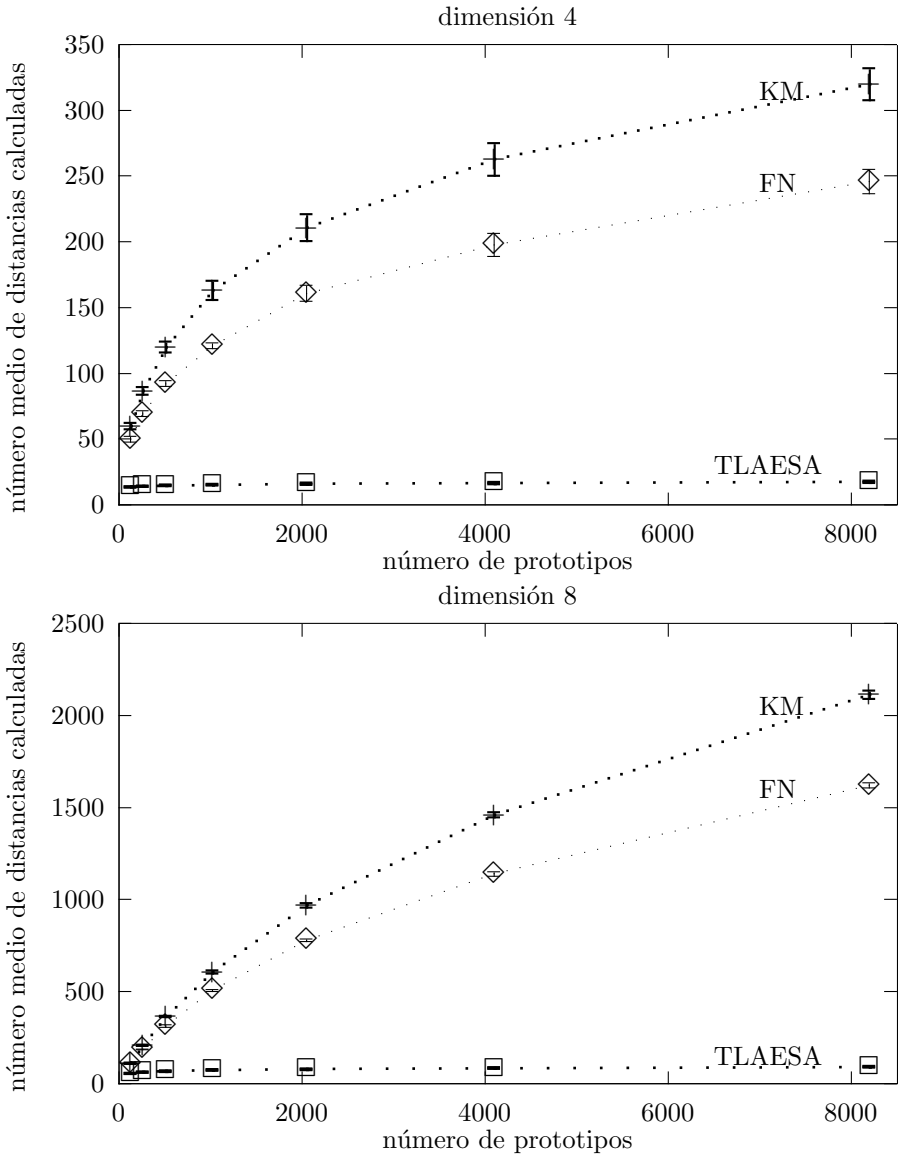

Figura 7.5: Número medio de distancias calculadas para el TLAESA y los algoritmos FN y KM cuando aumentamos el número de prototipos en un espacio vectorial euclídeo de dimensión 4 y 8, respectivamente.

Los algoritmos se han comparado teniendo en cuenta las dos características más importantes que se están estudiando: el número medio de distancias que calculan y su coste computacional medido en número de pasos de programa. Al igual que ya se hizo en el capítulo 6, también se ha estudiado el comportamiento de los tres algoritmos en el caso en el que se asocia un coste a la distancia. Esto contribuirá a elegir el algoritmo a utilizar según el coste de calcular la distancia utilizada en una aplicación concreta.

En la figura 7.5 se compara el número medio de distancias calculadas por los tres algoritmos en espacios vectoriales de dimensión 4 y 8, respectivamente. Para el algoritmo TLAESA se ha utilizado en cada dimensión el número óptimo de PB calculado en el capítulo 6 (10 PB en un espacio vectorial de dimensión 4 y 46 PB en un espacio vectorial de dimensión 8). El criterio de selección de PB utilizado fue en todos los casos el MDM (ver sección 3.2). En la figura 7.5 se puede observar que el algoritmo TLAESA es sustancialmente mejor que los algoritmos FN y KM para cualquier dimensión y tamaño del conjunto de prototipos.

En la figura 7.6 se compara el número medio de pasos de programa de los tres algoritmos. En este caso, el algoritmo TLAESA es peor que los otros dos en dimensión 8. Hay que recordar que los algoritmos FN y KM tienen un coste temporal que, en promedio, es proporcional al logaritmo del tamaño del conjunto de prototipos. En el capítulo 6 se demostró experimentalmente que el algoritmo TLAESA tenía un coste temporal sublineal. Sin embargo, si se representa esta misma figura asociando un coste al cálculo de las distancias, que es donde se comporta mejor el algoritmo TLAESA, se puede ver cuándo empieza a ser mejor este algoritmo respecto a los algoritmos FN y KM. Estos resultados se pueden ver en la figura 7.7 para un coste asociado a la distancia de 100 pasos de programa (ver capítulo 6), y en la figura 7.8 para un coste asociado a la distancia de 1000 pasos de programa (en realidad las figuras 7.7 y 7.8 son el resultado de aplicar a la figura 7.6 un coste a las distancias calculadas). Se puede observar en estas dos últimas figuras que a partir de un valor no muy elevado del coste de la distancia (menor que el que suele tener en un caso real) el algoritmo TLAESA es mejor que los otros dos (utiliza menos pasos de programa, equivalente a decir que es más rápido).

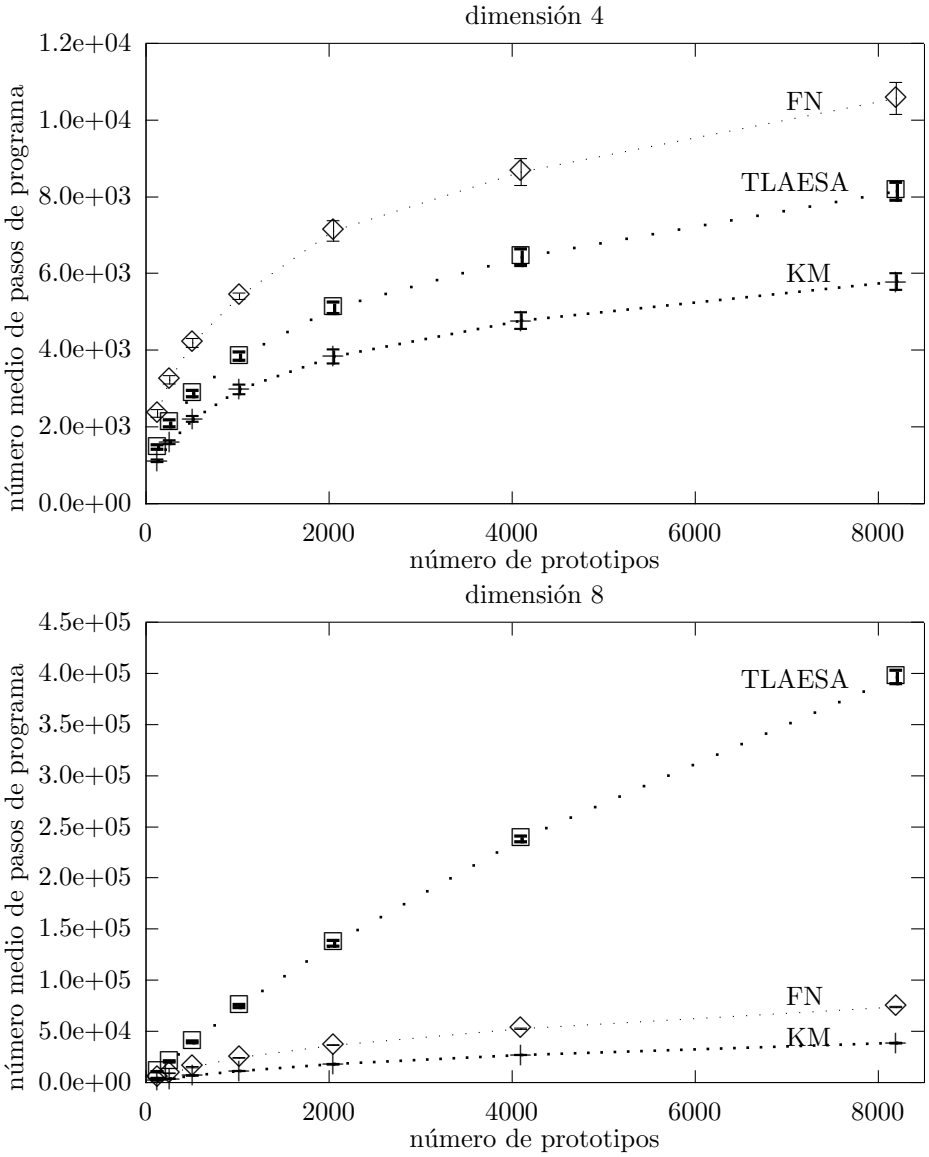

Figura 7.6: Número medio de pasos de programa consumidos por TLAESA y por los algoritmos FN y KM cuando aumenta el número de prototipos para la búsqueda en un espacio vectorial euclídeo de dimensión 4 y 8, respectivamente.

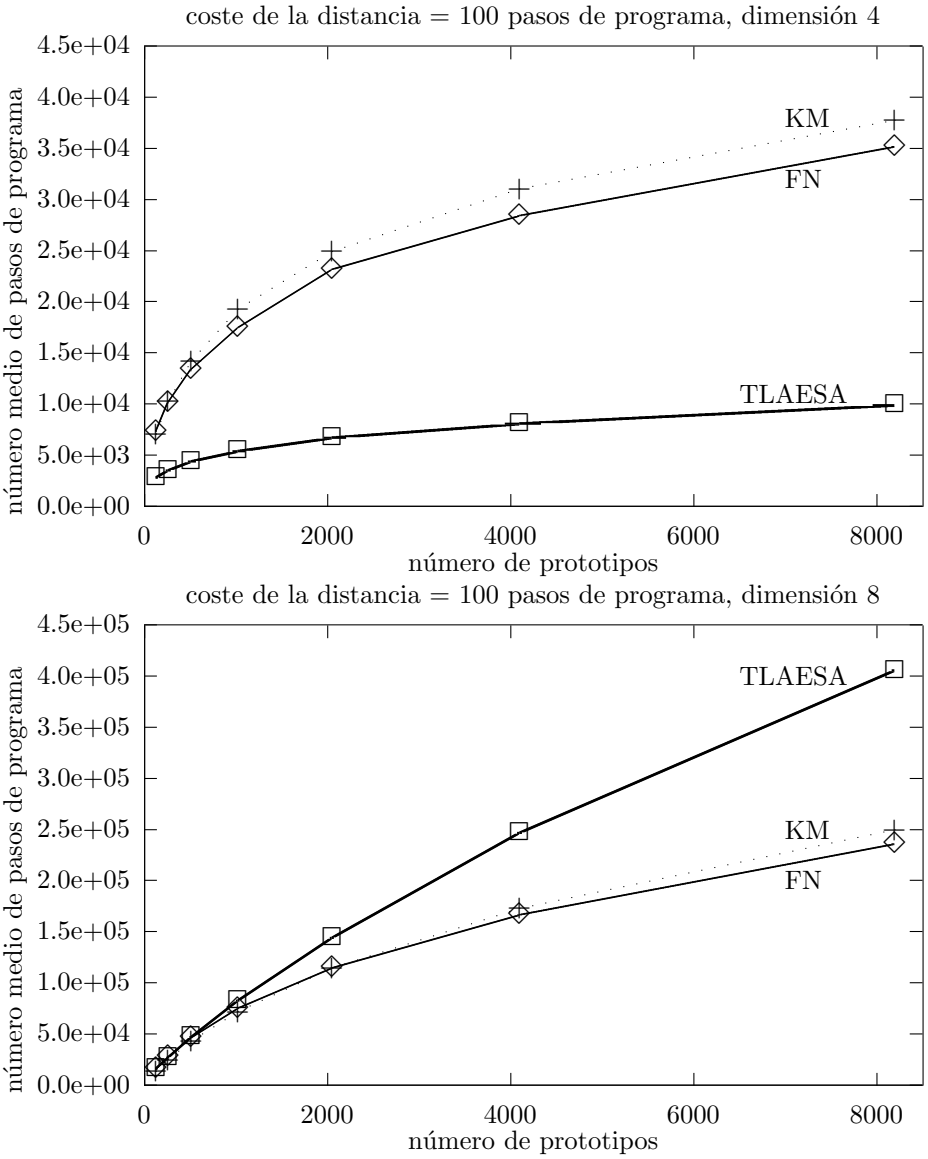

Figura 7.7: Número medio de pasos de programa consumidos por TLAESA y por los algoritmos FN y KM cuando aumenta el número de prototipos para la búsqueda en un espacio vectorial euclídeo de dimensión 4 y 8, respectivamente. La desviación típica de los datos es la misma que la de la figura 6. Se ha asociado un coste relativo de 100 pasos a las distancias.

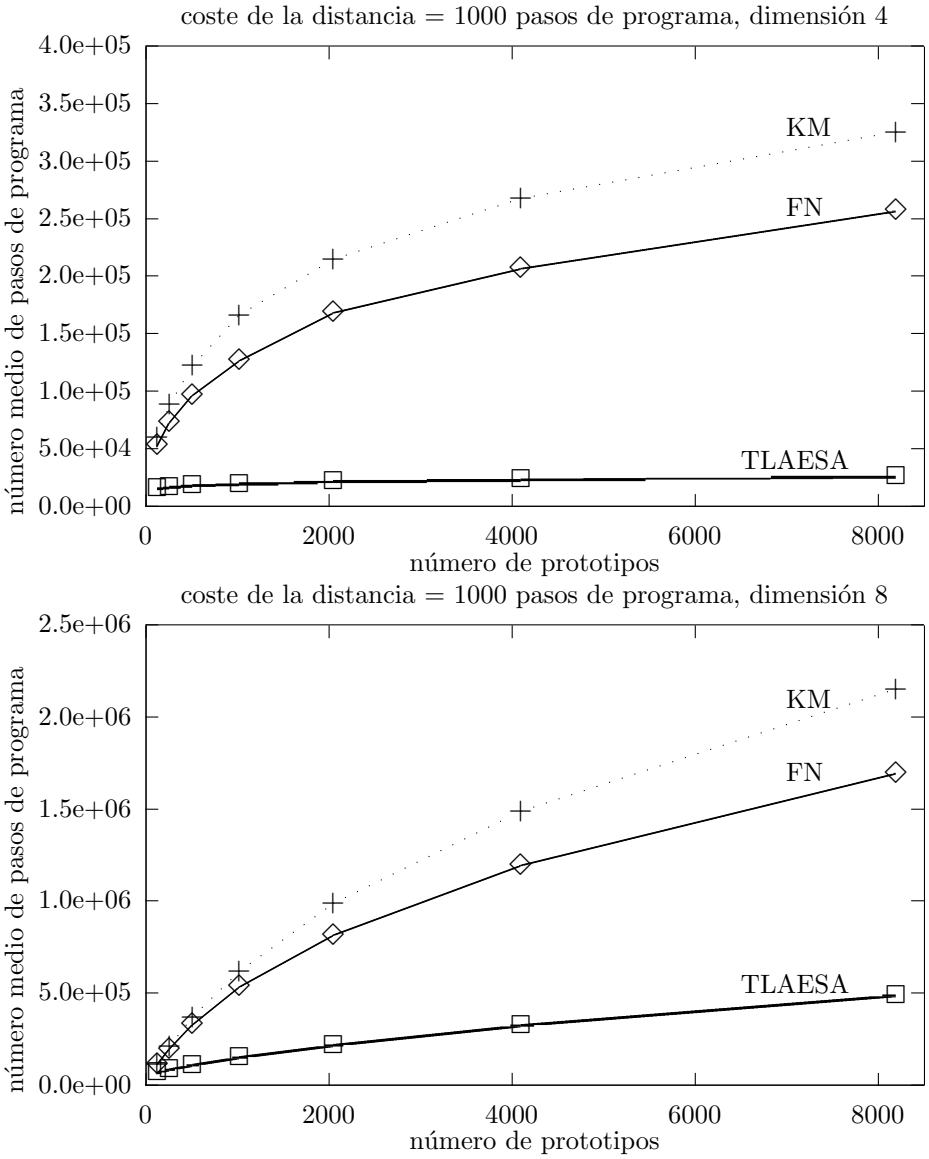

Figura 7.8: Número medio de pasos de programa consumidos por TLAESA y por los algoritmos FN y KM cuando aumenta el número de prototipos para la búsqueda en un espacio vectorial euclídeo de dimensión 4 y 8, respectivamente. La desviación típica de los datos es la misma que la de la figura 6. Se ha asociado un coste relativo de 1000 pasos a las distancias.

7.3. Conclusiones

En este capítulo se ha comparado el comportamiento de los algoritmos propuestos en este trabajo con otros aparecidos previamente en la literatura y que también trabajan en espacios métricos. Se ha comparado por separado aquellos que tenían características similares en cuanto al coste computacional. Así se ha podido comprobar que el algoritmo LAESA presentado como alternativa al algoritmo AESA, a pesar de calcular un número de distancias un poco mayor que éste, este valor tiende a ser constante. Pero lo más importante es que este resultado se ha conseguido utilizando un coste espacial *lineal* con el tamaño del conjunto de prototipos, frente a un coste cuadrático en el caso del algoritmo AESA. Ambos algoritmos, sin embargo, tienen un coste computacional total lineal con el tamaño del conjunto de prototipos.

Por otro lado, se ha comparado el algoritmo TLAESA con otros algoritmos de coste computacional sublineal, concretamente con coste logarítmico en promedio con respecto al tamaño del conjunto de prototipos. El coste computacional de TLAESA es superior al de estos algoritmos, aunque se ha demostrado experimentalmente que es sublineal (como se comprobó en el capítulo 6). Sin embargo, si se asocia un coste a la distancia, el algoritmo TLAESA se comporta mejor que los algoritmos FN y KM.

Capítulo 8

Experimentos con datos reales

Existe una serie de problemas en reconocimiento de formas para los cuales, a pesar de no disponer de una representación adecuada de los datos u objetos en un espacio vectorial, se dispone al menos de una medida de disimilitud entre cada par de datos. El cálculo de esta medida de disimilitud es, en muchos casos, especialmente caro. Algunos ejemplos de estas medidas son la distorsión basada en alineamiento temporal no lineal (dynamic time warping, DTW) que se utiliza en el reconocimiento de palabras aisladas [RL, 84], [CV, 87], las medidas de disimilitud (exponenciales con la talla de los grafos) aplicadas al reconocimiento de descripciones relacionales (grafos con atributos) [SF, 83], [SH, 85], o la disimilitud entre cadenas de símbolos basada en distancia de edición, la cual se utiliza en muchas aplicaciones de reconocimiento sintáctico de formas [WF, 74].

En este capítulo se estudia el comportamiento de los algoritmos propuestos en un problema real de reconocimiento de formas. Concretamente, se aplicarán al reconocimiento de caracteres manuscritos, utilizando como representación cadenas de símbolos que representan el contorno de los mismos [GMO, 95]. Esta elección ha sido realizada por los buenos resultados de clasificación que se obtienen con una técnica tan sencilla de extracción de características. Por otro lado, el elevado coste computacional de la distancia a aplicar (distancia de edición) hace que los algoritmos que se proponen estén especialmente indicados para esta tarea de reconocimiento ya que, a medida que aumenta el tamaño de los conjuntos de prototipos sobre los que se va a realizar la clasificación, el número de distancias resulta independiente de dicho tamaño. Aunque estos resultados ya han sido estudiados y comentados en los capítulos anteriores, ahora se va a comprobar el comportamiento de los algoritmos (número medio de distancias calculadas, tasa de error en la clasificación, valor óptimo para el tamaño del conjunto de PB, comparación con otros métodos) en una aplicación con datos reales.

8.1. Extracción de características

El procedimiento utilizado para la extracción de características consiste en la obtención del contorno de los caracteres (representados sobre matrices de 128 × 128 píxeles blancos y negros, de forma que los caracteres están en negro sobre un fondo blanco) utilizando para ello ocho vectores de dirección. El método comienza buscando el primer píxel negro en un recorrido de izquierda a derecha, empezando por la parte superior de la matriz (es la forma de leer en las lenguas occidentales). Una vez localizado este primer píxel se recorre el contorno del carácter hasta que se vuelve a aquel por el que se ha comenzado. Durante el recorrido se va completando la cadena que dependerá de las direcciones que se han ido siguiendo para obtener el contorno. Un ejemplo de este recorrido se puede ver en la figura 8.1.

A este procedimiento se le ha añadido una modificación para reducir con ello la longitud de las cadenas obtenidas. Esta modificación consiste en obtener la cadena de contorno de una matriz de 64 × 64 píxeles, a partir de la sustitución de 4 píxeles por uno en la matriz original. El método de sustitución ha consistido simplemente en considerar que el píxel es negro si al menos uno de los cuatro que representa lo es.

8.2. Distancia de edición

La distancia de edición entre dos cadenas se define como el mínimo coste que se requiere para realizar un conjunto de transformaciones que permiten obtener una cadena a partir de otra. Estas transformaciones son inserción, borrado y sustitución de un símbolo de la cadena. Bajo ciertas condiciones [WF, 74] esta distancia cumple la siguiente ecuación recursiva:

$$d(\lambda,\lambda) = 0$$

$$d(xa,yb) = \text{mín} \begin{cases} d(x,yb) + P_b & \\ d(xa,y) + P_i & \\ d(x,y) + P_s & \text{si} \quad a \neq b \\ d(x,y) & \text{si} \quad a = b \end{cases}$$

donde P_b, P_i y P_s son, respectivamente, los costes de las operaciones de borrado, inserción y sustitución, $a, b \in \sum$, $x, y \in \sum^*$, y λ es la cadena vacía.

En este trabajo se han utilizado la siguiente función de costes:

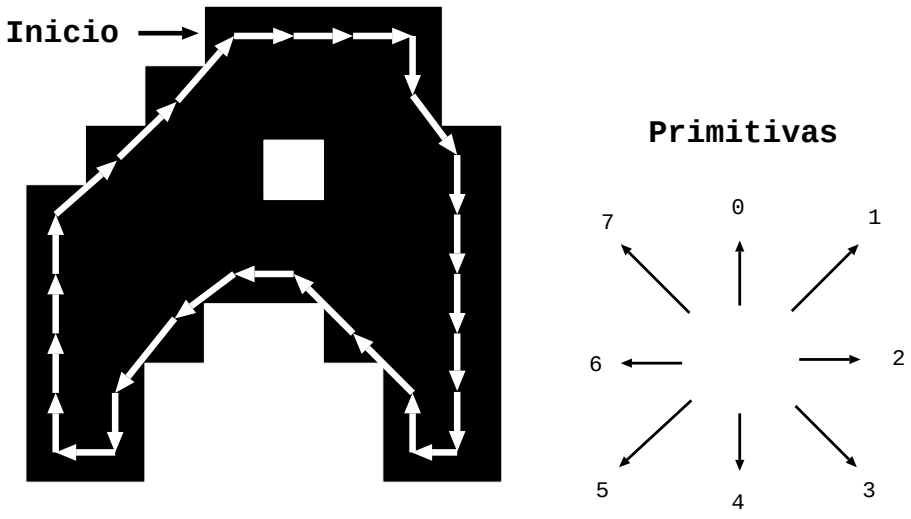

Figura 8.1: Ejemplo de extracción de la cadena de contorno para un ejemplo de la letra “A”.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	λ
0	0	1	2	3	4	3	2	1	1
1	1	0	1	2	3	4	3	2	1
2	2	1	0	1	2	3	4	3	1
3	3	2	1	0	1	2	3	4	1
4	4	3	2	1	0	1	2	3	1
5	3	4	3	2	1	0	1	2	1
6	2	3	4	3	2	1	0	1	1
7	1	2	3	4	3	2	1	0	1
λ	1	1	1	1	1	1	1	1	

Esta distancia se puede calcular en un tiempo $O(|x| \times |y|)$ (donde $|x|$ e $|y|$ son las longitudes de las cadenas x e y) usando una técnica de programación dinámica [WF, 74]. El valor de la distancia utilizado en todos los experimentos fue normalizado respecto a la suma de las longitudes de las dos cadenas.

8.3. Preparación de los experimentos

Para comprobar el comportamiento de los algoritmos se utilizó la base de datos NIST Special Database 3 (National Institute of Standards and Technology), de la cual se extrajeron los dígitos escritos por 100 escritores. Como cada escritor había escrito 10 repeticiones de cada dígito, los conjuntos extraídos para realizar los experimentos constan de un número determinado de escritores. Por ejemplo, se ha podido extraer un conjunto del orden de 1000 dígitos a partir de todas las repeticiones de dígitos escritos por 10 escritores. Siguiendo el mismo criterio se extrajeron los diferentes tamaños para el conjunto de prototipos que se utilizan en los experimentos.

Estos tamaños de conjuntos fueron utilizados como conjuntos de muestras, mientras que para el conjunto de prototipos se crearon varios subconjuntos de diferentes tamaños.

En estos experimentos se ha utilizado, además de los parámetros propios de cada algoritmo, otro denominado *holgura*, H. Este parámetro se puede estimar a partir de la llamada "holgura de la desigualdad triangular"[VRCB, 88], definida como:

$$H(x, y, z) = d(x, y) + d(y, z) - d(x, z) \quad \forall x, y, z \in E, \tag{8.1}$$

donde E es el espacio de representación. Para cada $x, y, z \in E$, este valor representa la holgura con la que la desigualdad triangular se cumple para el

triplete (x, y, z) con la medida de disimilitud utilizada [Vid, 85]. Si se calcula un histograma de valores $H(x, y, z)$ para un gran número de tripletes (x, y, z), este histograma puede usarse como una estimación de la probabilidad con la que se cumplirá la desigualdad triangular con distintas holguras. Mediante esta estimación puede determinarse un valor H tal que la probabilidad de holguras menores que H sea despreciable [VCR, 85].

Este valor fue utilizado en la condición de eliminación de prototipos en el algoritmo AESA para mejorar el comportamiento de dicho algoritmo en un caso real. Tomando un valor adecuado para este parámetro el algoritmo calcula muchas menos distancias, sin variación apreciable de la tasa de error en la clasificación con respecto a la búsqueda exhaustiva. El valor de la holgura se puede calcular a partir de métodos propuestos en [Vid, 85] o experimentalmente como veremos más adelante. En el algoritmo LAESA, la holgura se aplica también en la condición de eliminación de un prototipo p (ver sección 4.1) de la siguiente forma:

$$g_p \geq D_{\min} - H \tag{8.2}$$

donde g_p es la cota inferior de la distancia de la muestra al prototipo p y $D_{\min}$ es la distancia del prototipo más cercano a la muestra hasta el momento.

Así mismo, para el algoritmo TLAESA la condición de eliminación o poda para cada nodo t del árbol (ver sección 6.2) es:

$$g_{m_t} \geq D_{\min} + r_t - H \tag{8.3}$$

donde g_{m_t} es la cota inferior de la distancia de la muestra al nodo con representante m_t, $D_{\min}$ es la distancia del prototipo más cercano a la muestra hasta el momento, y r_t es el radio del nodo en cuestión.

La ecuación 8.3 ha sido aplicada también en los algoritmos FN y KM, ya que la regla de eliminación de nodos no terminales en estos algoritmos es la misma que la utilizada en el algoritmo TLAESA. A su vez, la regla de eliminación de prototipos en los nodos terminales (hojas) es:

$$d(x, m_t) \geq D_{\min} + d(p, m_t) - H \tag{8.4}$$

donde p es un prototipo perteneciente a las hojas del árbol.

El número medio de dígitos para cada escritor obtenidos de la base de datos fue de 100, por lo que el tamaño elegido para los ficheros de prototipos y muestras es un múltiplo de este valor. Concretamente, el número mínimo de dígitos que contienen los ficheros con los que se han hecho los experimentos fue de 1000, que corresponde a los dígitos obtenidos de 10 escritores. El tamaño máximo de fichero con el que se ha trabajado ha sido de 8000 dígitos.

En todos los experimentos se han utilizado como conjunto de muestras ficheros correspondientes a 10 escritores ($\sim$ 1000 dígitos). El tamaño de los conjuntos de prototipos ha dependido del experimento concreto. Sin embargo, en todos los casos se ha repetido cada experimento 9 veces, de forma que para la obtención de las prestaciones medias también se ha utilizado el *muestreo en dos etapas* [Coc, 77]. Además, para que los experimentos no dependiesen del número de escritores utilizado en cada caso, se utilizaron siempre 100 escritores, entre prototipos y muestras. Así, por ejemplo, en los experimentos con 1000 dígitos las 9 repeticiones consistían en utilizar como prototipos los escritores en grupos de 10, y como muestras otros 10 escritores distintos pertenecientes siempre al grupo de 100.

Utilizando grupos de 10 escritores como prototipos y muestras, se realizó un estudio experimental para comprobar cuál era el mejor valor para la holgura en cada uno de los algoritmos que se han propuesto en esta tesis. Este experimento también se realizó para los algoritmos FN y KM, ya que va a compararse temporalmente el comportamiento de estos algoritmos con el TLAESA. Los resultados obtenidos se presentan en la figura 8.2. Al igual que en el capítulo 7, los árboles utilizados para el algoritmo FN eran binarios. El número de niveles utilizado fue de 10 para conjuntos de 1000 prototipos aumentando hasta 13 para conjuntos del orden de 8000 prototipos.

En los experimentos que se presentan en las siguientes secciones se ha utilizado un valor para el parámetro de la holgura en el que la tasa de error en la clasificación empezaba a crecer en todos los algoritmos. Este valor ha sido, concretamente, H=0.1.

Los experimentos presentados son similares a los realizados con datos sintéticos en los capítulos 3, 4 y 5: estudio del comportamiento del algoritmo para los criterios de eliminación más representativos, comparación de criterios de selección y comportamiento del algoritmo respecto al número de distancias calculadas y al tiempo consumido a medida que aumenta el número de prototipos.

8.3.1. Experimentos con el LAESA

En primer lugar se estudia el comportamiento del algoritmo para los criterios de eliminación con los que se obtuvieron resultados más satisfactorios (ver sección 4.1). El objetivo es, en este caso, encontrar el tamaño óptimo del conjunto de PB para el criterio E_1, y a partir de qué tamaño es interesante elegirlo para los criterios E_∞ y E_{elim}.

Para este experimento se han utilizado conjuntos de 10 escritores que corresponden a un total de 1000 dígitos de media. Cada experimento fue repetido 9 veces de la forma mencionada en el apartado anterior.

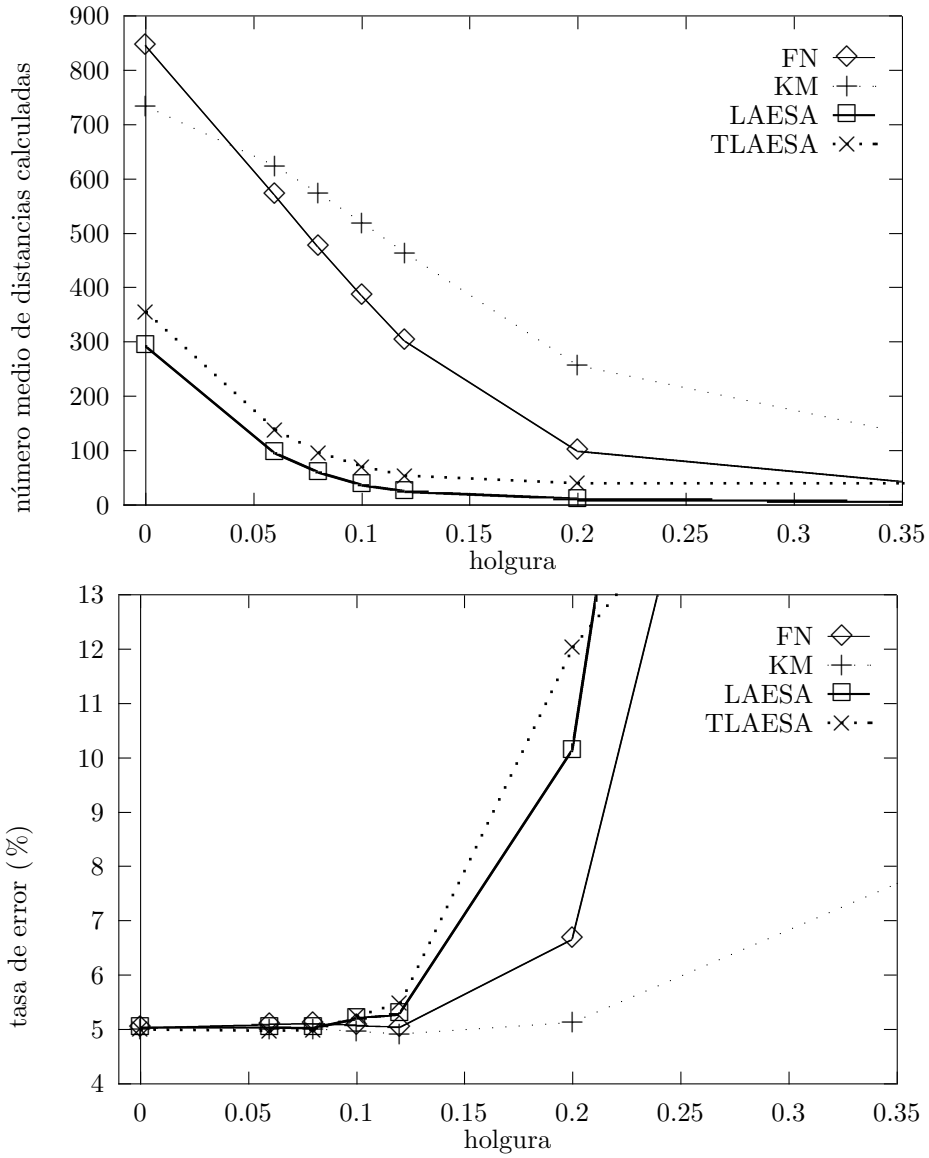

Figura 8.2: Número medio de distancias calculadas y tasa de error asociada en función de la holgura para los algoritmos LAESA, TLAESA, FN y KM utilizando un conjunto de 1000 prototipos. El criterio de selección de PB utilizado en los algoritmos LAESA y TLAESA fue el MDM y el número de PB fue 40. El criterio de eliminación utilizado en el LAESA fue E_{elim}. La desviación típica media de todas las estimaciones fue del 4 %.

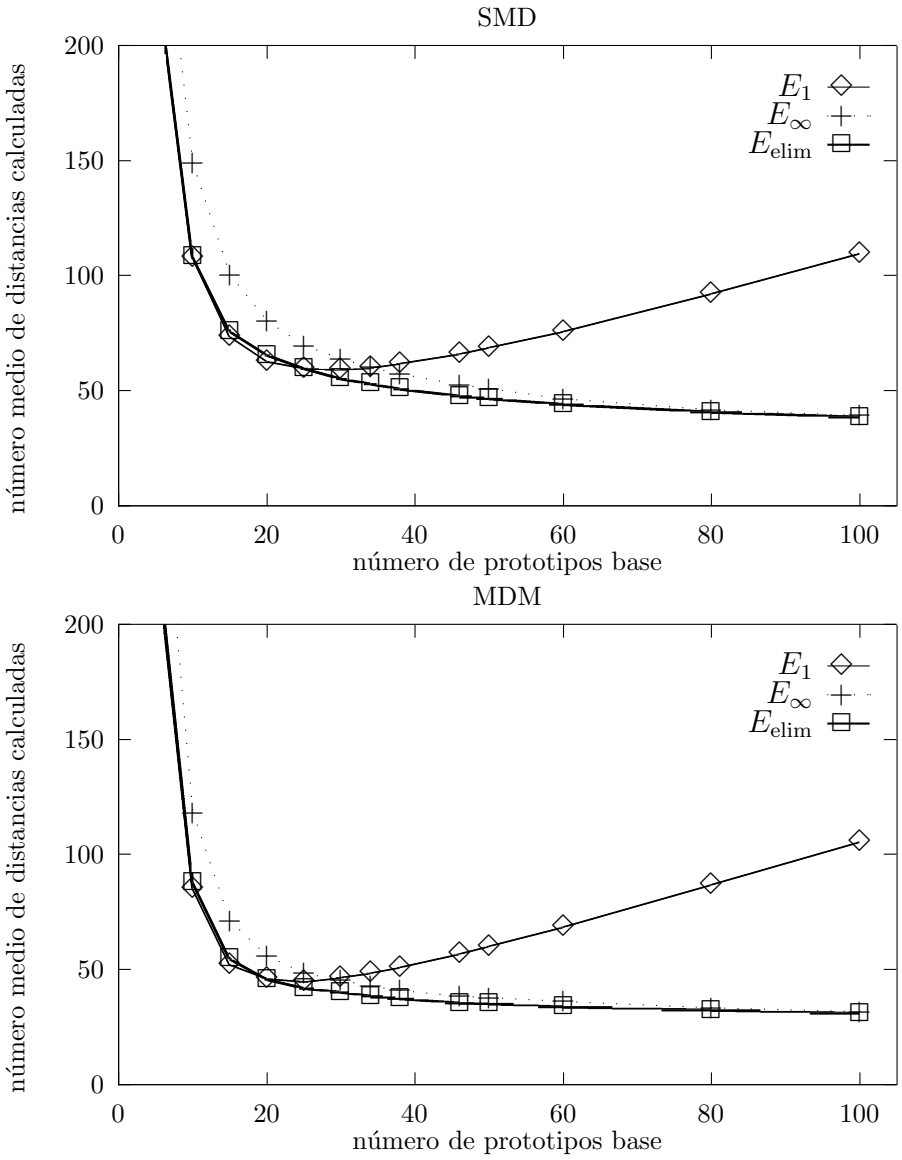

Figura 8.3: Número medio de distancias calculadas por el LAESA al variar el número de PB con los diferentes criterios de eliminación de PB, para un conjunto de 1000 prototipos de media. La desviación típica media de los datos fue del 4 %

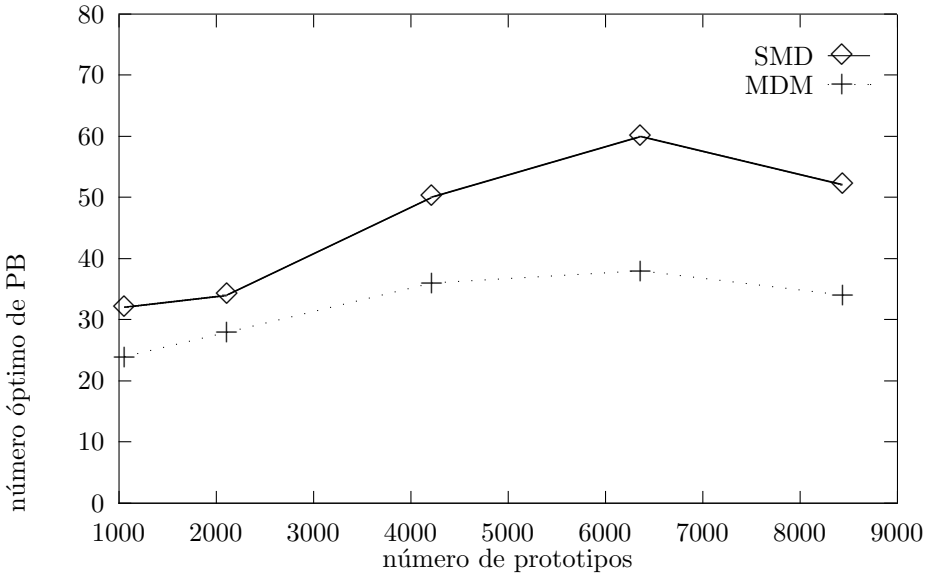

Figura 8.4: Tamaño óptimo del conjunto de PB para el algoritmo LAESA, usando el criterio de eliminación E_1, a medida que aumenta el tamaño del conjunto de prototipos.

Observando la figura 8.3 se puede ver que el tamaño óptimo usando el criterio E_1 es de 25 PB utilizando el criterio de selección de PB MDM y un poco superior ($\sim$ 30) en el caso de utilizar el criterio SMD. También, para este valor, la pendiente en los otros dos criterios empieza a ser menos pronunciada.

Otro experimento realizado ha sido el estudio de la variación del tamaño óptimo del conjunto de PB respecto al tamaño del conjunto de prototipos. Este valor llegaba a mantenerse constante conforme aumentaba el tamaño del conjunto de prototipos cuando se trabajaba con datos sintéticos (ver sección 3.5.2). En la figura 8.4 se han representado los valores óptimos para $|B|$ al variar el tamaño del conjunto de prototipos. Como se puede observar, el comportamiento es el mismo que el obtenido con datos sintéticos.

Como consecuencia, conjuntos de PB de tamaños similares a los óptimos y superiores han sido utilizados para estudiar el comportamiento del algoritmo a medida que aumenta el tamaño del conjunto de prototipos. El experimento se ha realizado utilizando el criterio de eliminación E_{elim}.

En la figura 8.5 se puede ver que, aunque no se observa aún claramente cómo el número de distancias tiende a ser constante en promedio, parece que sí tiende a este comportamiento.

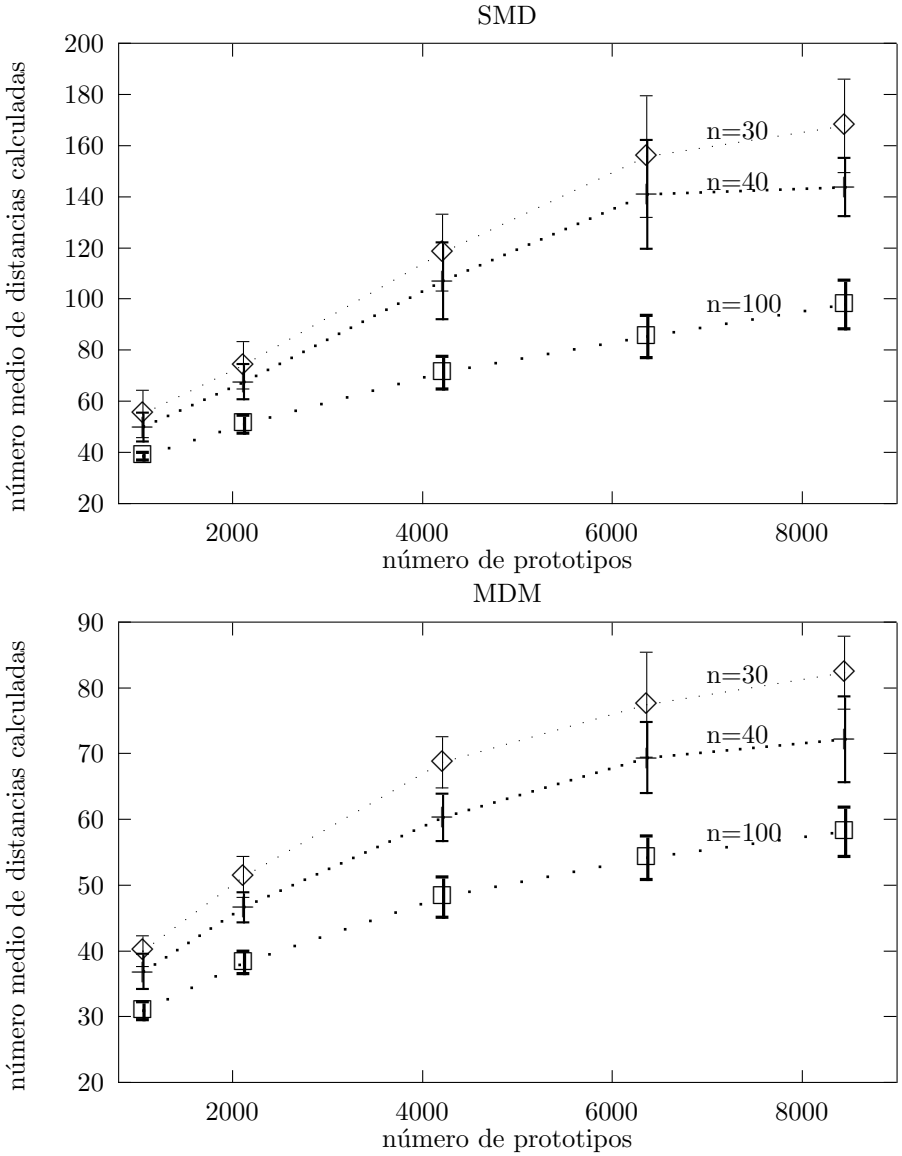

Figura 8.5: Número medio de distancias calculadas por el LAESA al variar el número de prototipos, usando el criterio de eliminación E_{elim} y los criterios de selección de PB MDM y SMD, para diferentes tamaños del conjunto de PB, concretamente, los valores utilizados han sido 30, 40 y 100.

Además, al igual que ocurría con datos sintéticos, los resultados obtenidos al utilizar el método de selección de prototipos base MDM son mejores que con SMD; de hecho, en este caso, las diferencias son aún mayores. En la figura 8.6 se presentan los resultados de la tasa de error del experimento presentado en la figura 8.5. Se puede observar que, como era de esperar, la tasa de error disminuye a medida que aumenta el número de prototipos. En este caso, la tasa de error disminuye hasta un valor de 3 %.

También se ha comprobado el comportamiento del algoritmo utilizando el criterio de selección SELECCIÓN$_r$ (secciones 5.1 y 5.2). En la figura 8.7 se ha representado el comportamiento del algoritmo utilizando la condición de eliminación E_{elim} para los dos criterios de selección de prototipos base: MDM y SMD. Los resultados obtenidos demuestran que el algoritmo se comporta prácticamente igual con los dos criterios de selección estudiados cuando se aplica a datos reales. Algo similar ocurría en los experimentos con datos sintéticos, aunque en ese caso las diferencias para los distintos valores del parámetro r eran algo mayores. Al igual que en los experimentos previos de este capítulo se obtienen mejores resultados cuando se utiliza el criterio MDM para la selección de los PB.

8.3.2. Experimentos con el TLAESA

En primer lugar se trata de determinar el tamaño óptimo del conjunto de PB. Este experimento se ha repetido en las mismas condiciones que el de la figura 8.3 en la sección 8.3.1. En la figura 8.8 aparece el comportamiento del TLAESA para los dos criterios de selección de prototipos base, SMD y MDM. Existe un valor óptimo para $|B|$ en el que el número de distancias calculadas es mínimo. Este mínimo coincide en el caso de utilizar cualquiera de los dos criterios de selección de PB, aunque el número de distancias calculadas es claramente inferior si se utiliza MDM.

Repitiendo el experimento de la figura 8.4 para el algoritmo TLAESA, se llega a la misma conclusión que se había llegado para el LAESA, es decir, el número medio de distancias parece que tiende a una constante (aunque haría falta comprobarlo para tamaños más grandes del conjunto de prototipos). Los resultados con el método MDM son bastante mejores que con el método SMD.

Por último, se presenta un experimento en el que se compara el número medio de distancias calculadas y el tiempo consumido por los algoritmos LAESA, TLAESA, FN y KM en función del tamaño del conjunto de prototipos. En la figura 8.10 se observa la diferencia apreciable que hay entre

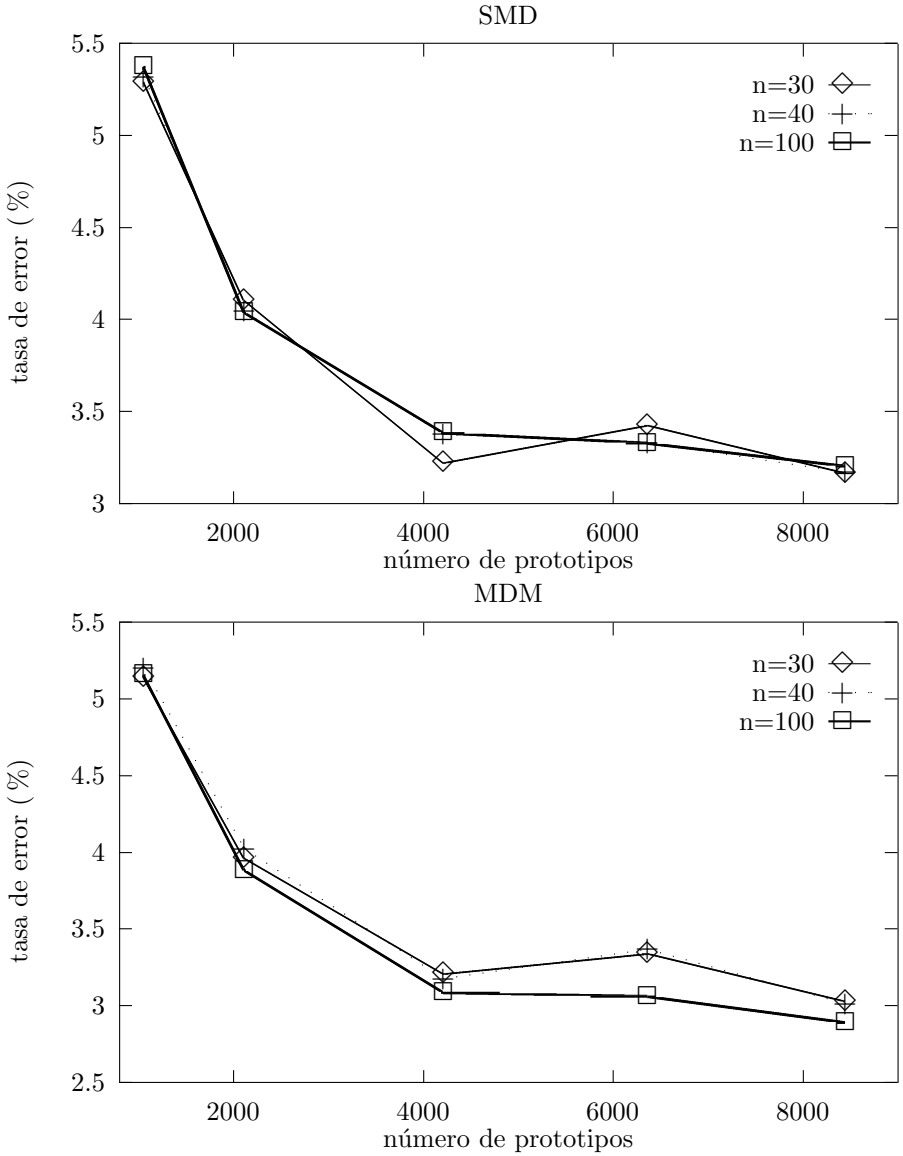

Figura 8.6: Tasa de error en la clasificación utilizando el algoritmo LAESA al variar el número de prototipos, usando el criterio de eliminación E_{elim} y los criterios de selección de PB MDM y SMD, para diferentes tamaños del conjunto de PB, concretamente, los valores utilizados han sido 30, 40 y 100.

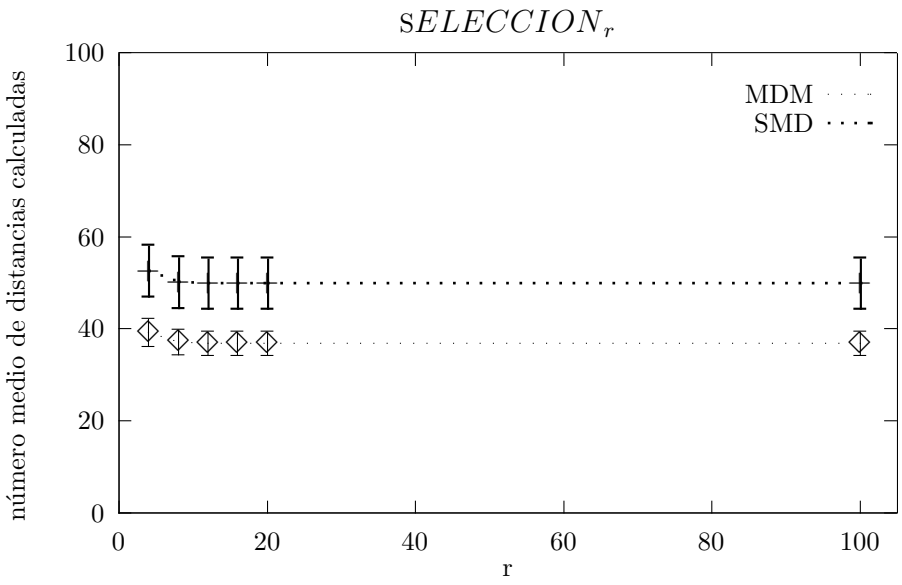

Figura 8.7: Número medio de distancias calculadas por el LAESA al variar el parámetro r del criterio de selección, usando el criterio de eliminación E_{elim} para un conjunto de 1000 prototipos. El número de PB utilizado fue de 40.

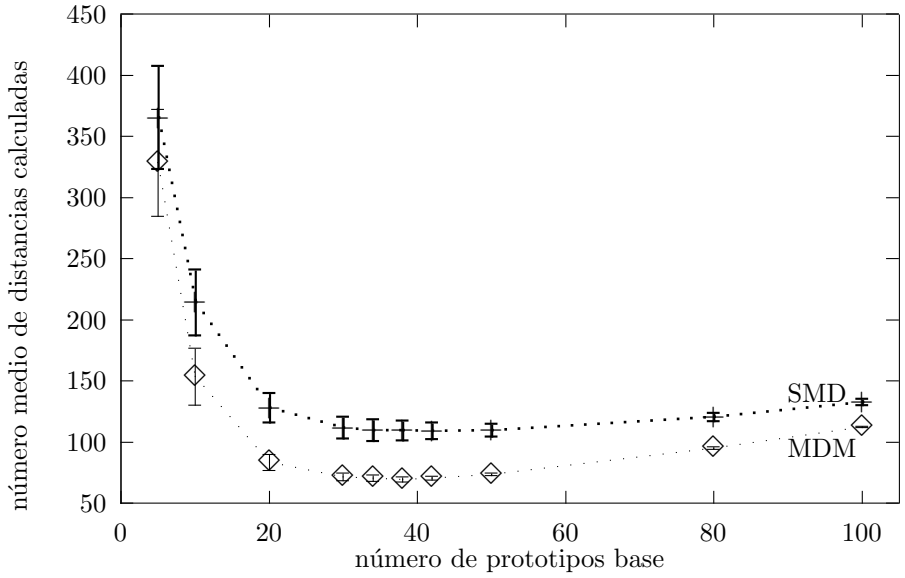

Figura 8.8: Número medio de distancias calculadas por el TLAESA al variar el número de prototipos base, usando los criterios de selección de PB SMD y MDM.

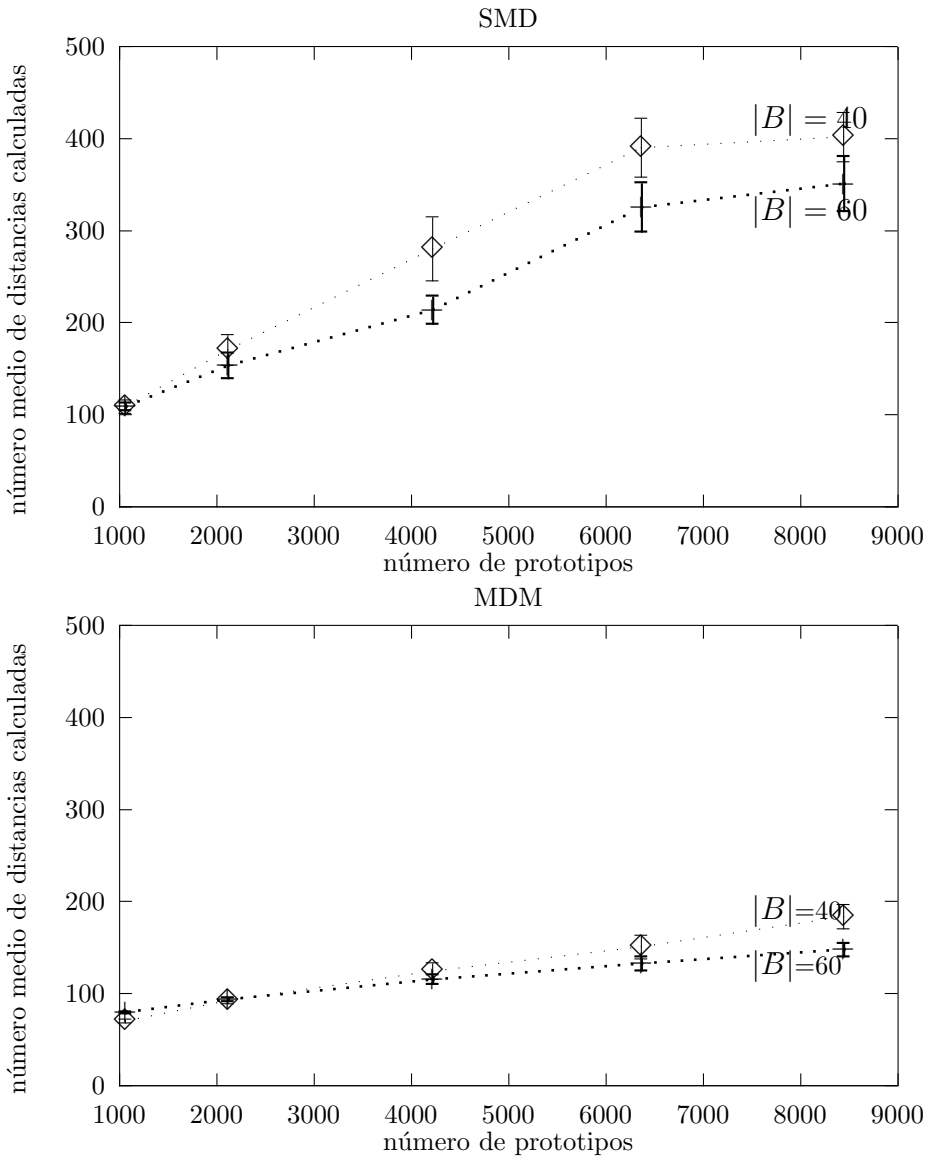

Figura 8.9: Número medio de distancias calculadas por el TLAESA al variar el número de prototipos para los criterios de selección de prototipos base MDM y SMD.

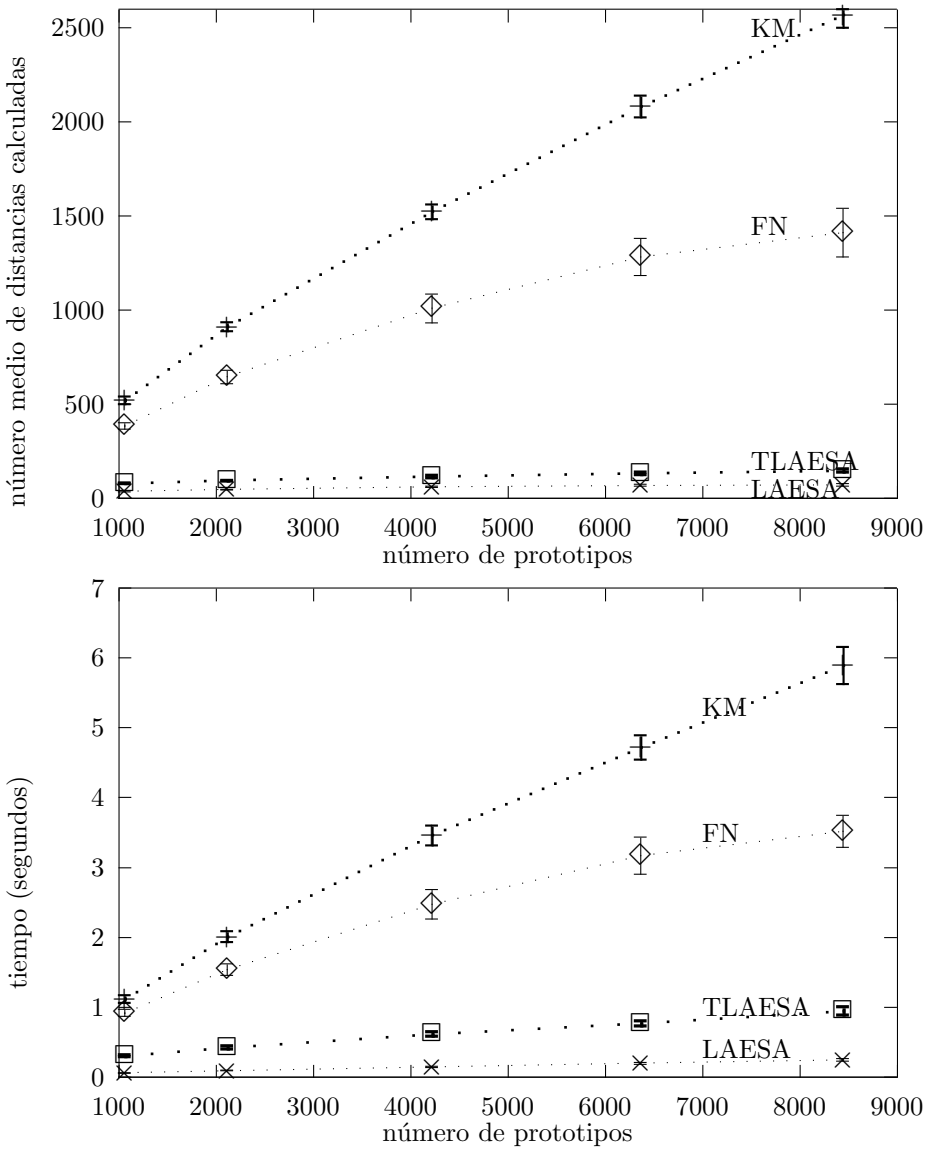

Figura 8.10: Número medio de distancias calculadas y tiempo consumido por los algoritmos FN, KM, TLAESA y LAESA al variar el número de prototipos. El criterio de selección de PB utilizado en LAESA y TLAESA ha sido el MDM. El criterio de eliminación utilizado en LAESA fue el E_{elim} para 40 PB, mientras que para el TLAESA se utilizaron 60 PB.

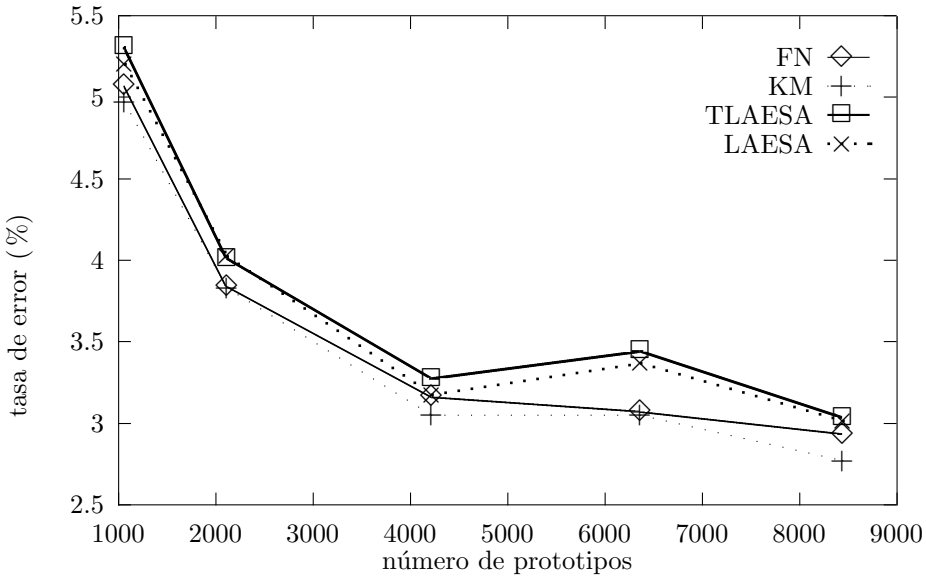

Figura 8.11: Tasa de error en la clasificación para los algoritmos FN, KM, TLAESA y LAESA al variar el número de prototipos. El criterio de selección de PB utilizado en LAESA y TLAESA ha sido el MDM. El criterio de eliminación utilizado en LAESA fue el E_{elim} para 40 PB, mientras que para el TLAESA se utilizaron 60 PB.

los métodos propuestos, LAESA y TLAESA, respecto a FN y KM, tanto en número de distancias como el tiempo. Este resultado permite decir que cuando se trabaja con datos reales en los que el cálculo de la distancia tiene un coste apreciable, es más importante el peso asociado a este coste que al del propio coste computacional (overhead) de los diferentes algoritmos si se usan conjuntos de prototipos de tamaño mediano. El coste computacional empieza a ser más importante a medida que aumenta dicho tamaño. La tasa de error asociada al experimento de la figura 8.10 se presenta en la figura 8.11.

8.4. Conclusiones

En este capítulo se ha estudiado el comportamiento de los algoritmos propuestos en los capítulos 5 y 6 en un caso real, concretamente en el reconocimiento de caracteres manuscritos. Se ha repetido con datos reales la mayoría de los experimentos realizados con datos sintéticos realizados en los capítulos anteriores constatando conclusiones similares. Concretamente, se puede decir que en este caso los algoritmos se comportan de la forma esperada. En el caso concreto del número medio de distancias calculadas, se ha podido comprobar la tendencia asintóticamente constante a medida que aumenta el número de prototipos, sobre todo en el algoritmo LAESA. Esta tendencia se debería observar mejor con el TLAESA para tamaños más grandes del conjunto de prototipos. Otro hecho interesante que se ha observado es que, como en datos sintéticos, los algoritmos funcionan mejor si se utiliza el método de selección de PB denominado MDM (máximo de las distancias mínimas). Sin embargo, en un caso real las diferencias entre utilizar este método o el SMD (máximo de la suma de distancias) han resultado mucho mayores que con datos sintéticos. Por último, también se ha podido comprobar la importancia que tiene en un experimento con datos reales calcular un número reducido de distancias, debido al elevado coste que conllevan y que hace que los algoritmos LAESA y TLAESA sean más rápidos que FN y KM. También se ha podido observar en este caso que para los tamaños que se han utilizado en el conjunto de prototipos (hasta 8000 prototipos) siempre ha sido mejor (más rápido) el LAESA que el TLAESA. Es de esperar que esta tendencia cambie cuando aumente considerablemente el tamaño del conjunto de prototipos.

Parte IV

Conclusiones y bibliografía

Capítulo 9

Conclusiones

En este trabajo se han presentado nuevos algoritmos de búsqueda de vecinos más próximos en espacios métricos. El objetivo principal ha sido que estos algoritmos tuviesen un comportamiento similar al del algoritmo AESA con respecto a la característica más importante del mismo: calcular un número de distancias constante en promedio a medida que aumenta el tamaño del conjunto de prototipos. Estos algoritmos, además de cumplir esta importante propiedad, se caracterizan porque la complejidad temporal y espacial del preproceso son lineales con el tamaño del conjunto de prototipos; es decir, se reducen en un orden los costes del algoritmo AESA. Esta reducción en la complejidad se ha obtenido gracias a la utilización de un subconjunto de prototipos al que se ha denominado *conjunto de prototipos base*. Se han propuesto varios métodos para la selección de este subconjunto de forma que cumpliesen dos objetivos: buen comportamiento del algoritmo (en lo referente al número medio de distancias calculadas) y coste temporal bajo.

En el capítulo 3 se ha realizado una serie de experimentos para comprobar que los métodos propuestos MDM (máximo de distancias mínimas) y SDM (suma máxima de distancias), de coste $O(|P|)$, siendo $|P|$ el tamaño del conjunto de prototipos P, se comportan mejor que otros métodos de coste superior como es el algoritmo de las c-medias, e incluso se comporta de forma similar a un método propuesto por Ramasubramanian de coste $O(|P|^3)$. Se da la circunstancia de que este método, a pesar de tener este coste tan elevado, ha servido para comprobar que la mejor disposición de los PB dentro del espacio de representación es aquella en la que los PB están máximamente separados.

A continuación el trabajo se ha centrado en mejorar el comportamiento del algoritmo realizando la gestión de los PB, tanto en la fase de aproximación como en la de eliminación. Para ello, se han introducido dos funciones, llamadas CONDICIÓN y SELECCIÓN para las que se han presentado varias

definiciones en los capítulos 4 y 5.

En el capítulo 6 se han presentado dos algoritmos que permiten reducir el coste computacional no asociado al coste de la distancia del algoritmo LAESA. En el mismo, se han estudiado las condiciones para las cuales es mejor utilizar uno u otro algoritmo.

En el capítulo 7 se han comparado experimentalmente los algoritmos propuestos con otros aparecidos anteriormente capaces también de trabajar en espacios métricos (no necesariamente vectoriales). Por último, los algoritmos han sido aplicados a un caso real en el capítulo 8, concretamente al reconocimiento de caracteres manuscritos. Aunque el LAESA ha sido utilizado ya en varias aplicaciones reales como, por ejemplo, el reconocimiento automático del habla [JV, 94] en este trabajo se han presentado resultados de su aplicación al reconocimiento de caracteres manuscritos. En este capítulo se ha comprobado que todos los comportamientos que presentaban los algoritmos con datos sintéticos se han manifestado similarmente en un caso real.

9.1. Desarrollos futuros

Este trabajo tiene una continuación en la profundización del algoritmo TLAESA. Por un lado, es necesario evitar que el comportamiento del algoritmo en número de distancias calculadas dependa del número de prototipos base seleccionados. Este problema se resolvió en el algoritmo LAESA con la introducción de criterios de eliminación de PB. En este caso, esto se podría conseguir en principio modificando la fase de preproceso. Por otro lado, se trataría de estudiar más detenidamente el algoritmo TLAESA, para demostrar el coste, aparentemente logarítmico, que tiene.

En estos momentos también se está investigando para reducir la tasa de error en la clasificación a partir de la obtención automática de la función de costes asociada a la distancia de edición. Una vez obtenida una función de costes óptima es de esperar que los algoritmos presentados en esta tesis mejoren su comportamiento, ya que se ha comprobado experimentalmente que el número de distancias que se calculan en un proceso de clasificación de una muestra disminuye cuando se aplica una función de costes que reduce la tasa de error en la clasificación.

Bibliografía

[AJV, 93] *Aibar, P., Juan, A. y Vidal, E.*
Extensions to the approximating and eliminating search algorithm (AESA) for finding k-nearest-neighbours

New Advances and Trends in Speech Recognition and Coding, 23–28, Granada, 1993.

[Ben, 75] *Bentley, J.L.*
Multidimensional Binary Search Trees Used for Assocciative Searching

Communications of the ACM, **18**:9, 509–517, Septiembre 1975.

[BK, 73] *Burkhard, W.A. y Keller, R.M.*
Some approaches to best match file searching

Comm. Ass. Comput. Mach. 16, 239–236.

[CGRS, 84] *Cheng, D.Y., Gersho, A. Ramamurthi, B. y Shoham, Y.*
Fast search algorithms for vector quantization and pattern matching
ICASSP, 1984.

[CH, 67] *Cover, T.M. y Hart, P.E.*
Nearest Neighbor Pattern Classification

IEEE Transactions on Information Theory, **IT-13**, 21–27, Enero 1967.

[CMC, 94] *Chaudhuri, D., Murthy, C.A. y Chaudhuri, B.B.*
Finding a Subset of Representative Points in a Data Set

IEEE Transactions on Systems, man, and Cybernetics, **24**:9, 1416–1424, Septiembre 94.

[Coc, 77] *Cochram, W.*
Sampling Techniques

Wiley, New York, 1977.

[CV, 87] *Casacuberta, F. y Vidal, E.*
Reconocimiento Automático del Habla

Marcombo, 1987.

[Das, 91] *Dasarathy, B.*
Nearest Neighbour(NN) norms: NN Pattern Classification Techniques

IEEE Computer Society Press, 1991.

[DH, 73] *Duda, R. y Hart, P.*
Pattern Classification and Scene Analysis

Wiley, 1973.

[DK, 82] *Devijver, P.A. y Kittler, J.*
Pattern Recognition: a Statistical Approach

Prentice Hall, 1982.

[FBF, 77] *Friedman, J., Bentley, J. y Finkel, R.A.*
An Algorithm for Finding Best Matches in Logarithmic Expected Time

ACM Transactions on Mathematical Software, **3**:3, 209–226, Septiembre 1977.

[FBS, 75] *Friedman, J.H., Baskett, F. y Shustek, L.J.*
An Algorithm for Finding Nearest Neighbors

IEEE Transactions on Computers, **c-24**:10, 1000–1006, Octubre 1975.

[FLL, 93] *Faragó, A., Linder, T. y Lugosi, G.*
Fast Nearest-Neighbor Search in Dissimilarity Spaces

IEEE Transactions on Pattern Analysis and Machine Intelligence. **15**:9, 957–962, 1993.

[FN, 75] *Fukunaga, K. y Narendra, M.*
A Branch and Bound Algorithm for Computing k-Nearest Neighbors

IEEE Transactions on Computers, 750–753, Julio 1975.

[FP, 70] *Fischer, F.P. y Patrick, E.A.*
A preprocessing algorithm for nearest neighbor decision rules

Proceedings of the National Electronic Conference, **26**, 481–485, Diciembre 1970.

[Fuk, 90] *Fukunaga, K.*
Introduction to Statistical Pattern Recognition

Academic Press Inc, 1990.

[GMO, 95] *Gómez, E., Micó, L. y Oncina, J.*
Testing the linear approximating eliminating search algorithm in handwritten character recognition tasks

Actas del VI Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes, 212–217, Córdoba 1995.

[Har, 68] *Hart, P.E.*
The condensed nearest neighbor rule

IEEE Transactions Information Theory, **IT-14**, 515–516, May 1968.

[HS, 78] *Horowitz, E. y Sahni, S.*
Fundamentals of computer algorithms

Computer Science Press, 1978.

[JV, 94] *Juan, A. y Vidal, E.*
Fast K-means-like clustering in metric spaces

Pattern Recognition Letters, **15**:1, 19–25, Enero 1994.

[KM, 83] *Kalantari, I. y McDonald, G.*
A Data Structure and an Algorithm for the Nearest Point Problem

IEEE Transactions on Software Engineering, **SE-9**:5, 631–634, Septiembre 1983.

[KPK, 85] *Kamgar-Parsi, B. y Kanal, L.*
An Improved Branch and Bound Algorithm for Computing k-Nearest Neighbors

Pattern Recognition Letters, **3**:1, 7–12, Enero 1985.

[MO, 92] *Micó, L. y Oncina, J.*
A new criterion for approximating in a recent version with linear preprocessing of the AESA algorithm

Advances in pattern recognition and applications (selected papers from the Vth spanish symposium on pattern recognition and image analysis), Ed. World Scientific, 3–11, Valencia 1992.

[MOV, 91] *Micó, L. Oncina, J. y Vidal, E.*
Algoritmo para encontrar el vecino más próximo en un tiempo medio constante con una complejidad espacial lineal

Tech. Report DSIC II/14-91, Universidad Politécnica de Valencia, 1991.

[MOV, 92] *Micó, L., Oncina, J. y Vidal, E.*
An algorithm for finding nearest neighbours in constant average time with a linear space complexity

Int. Conference on Pattern Recognition (ICPR), Le Hague, **II**, 557–560, Septiembre 1992.

[MOV, 94] *Micó, L., Oncina, J. y Vidal, E.*
A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing-time and memory requirements

Pattern Recognition Letters, **15**:1, 9–17, Enero 1994.

[MOC, 95] *Micó, L., Oncina, J. y Carrasco, R.C.*
A fast branch and bound nearest neighbour classifier in metric spaces

aceptado, pendiente de publicación en Pattern Recognition Letters.

[MS, 86] *Murphy, O.J. y Selkow, S.M.*
The efficiency of using k-d trees for finding nearest neighbors in discrete space

Information Processing Letters 23, 215–218, Noviembre 1986.

[NG, 88] *Niemann, H. y Goppert, R.*
An efficient branch-and-bound nearest neighbour classifier

Pattern Recognition Letters **7**, 67–72, Febrero 1988.

[PBJD, 79] *Pettis, K.W., Bailey, T.A., Jain, A.K. y Dubes, R.C.*
An intrinsic dimensionality estimator from near-neighbour information

IEEE Trans. Pattern Anal. Machine Intell. 1, 41.

[Ram, 92] *Ramasubramanian, V.* Comunicación privada.

[RL, 84] *Rabiner, L. y Levinson, S.E:*
Isolated and connected word recognition - Theory and selected aplications

IEEE Trans. Comm. 29, 621–659.

[RP, 88] *Ramasubramanian, V. y Paliwal, K.*
An Optimized K-d Tree Algorithm for Fast Vector Quantization of Speech

Signal Processing IV: Theories and Applications, 875–878 (Proc. EUSIPCO '88, Grenoble, France, Septiembre 1988)

[RP, 89] *Ramasubramanian, V. y Paliwal, K.*
A Generalized Optimization of the K-d Tree for Fast Nearest-Neighbour Search

Proc. of 4th IEEE Region 10 International Conference, TENCON'89, 565–568, Noviembre 1989.

[RP, 90] *Ramasubramanian, V. y Paliwal, K.*
An Efficient Approximation-Elimination Algorithm for Fast Nearest-Neighbour Search Based on a Spherical Distance Coordinate Formulation

Signal Processing V:Theories and Applications, 1323–1326, 1990.

[Ram, 91] *Ramasubramanian, V.*
Fast Algorithms for Nearest-Neighbour Search and Application to Vector Quantization

PhD dissertation, University of Bombay, 1991.

[RP, 92] *Ramasubramanian, V. y Paliwal, K.K.*
An efficient approximation-elimination for fast nearest-neigbor search

ICASSP-92.

[Sha, 77] *Shapiro, M.*
The Choice of Reference Points in Best-Match File Searching

Artificial Intelligence/Language Processing., **20**, 339–343, 1977.

[SH, 85] *Shapiro, L.G. y Haralik, R.M.*
A metric for comparing relational descriptions

IEEE Trans. Pattern Anal. Machine Intell. **7**, 90–94.

[SF, 83] *Sanfeliu, A. y Fu, K.S.*
A distance measure between attributed graphs for pattern recognition

IEEE Transactions on systems, man and cybernetics. **13**, 353–362.

[Set, 81] *Sethi, I.K.*
A fast algorithm for recognizing nearest neighbors

IEEE Transactions on systems, man, and cybernetics. **SMC-11**:3, Marzo 1981.

[SW, 90] *Shasha, D. y Wang, T.*
New Techniques for Best-Match Retrieval

ACM Transactions on Information Systems, **8**:2, 140–158, 1990.

[TG, 74] *Tou, J.T. y Gonzalez, R.C.*
Pattern Recognition Principles

Addison Wesley, 1974.

[Vid, 85] *Vidal, E.*
Diversas aportaciones al Reconocimiento Automático del Habla

Tesis Doctoral. Fac. Físicas. Universidad de Valencia, 1985.

[Vid, 86] *Vidal, E.*
An algorithm for finding nearest neighbours in (approximately) constant average time complexity

Pattern Recognition Letters, **4**, 145–157, 1986.

[VL, 88] *Vidal, E. y Lloret, M.J.*
Fast Speaker Independent DTW Recognition of Isolated Words Using a Metric-Space Search Algorithm (AESA)

Speech Communication, **7**:4, 417–422, 1988.

[VCR, 85] *Vidal, E., Casacuberta, F. y Rulot, H.*
Is the DTW "distance" really a metric? An algorithm reducing the number of DTW comparisons in isolated word recognition

Speech Communication, 4, 333–344, 1985.

[VRCB, 88] *Vidal, E., Rulot, H., Casacuberta, F. y Benedí, J.*
On the use of a metric-space search algorithm (AESA) for fast DTW-bassed recognition of isolated words

IEEE Transactions on Acoustics, Speech, and Signal Processing, **36**, 651–660, 1988.

[Vid, 94] *Vidal, E.*
New formulation and improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA)

Pattern Recognition Letters, **15**:1, 1–7, Enero 1994.

[WF, 74] *Wagner, R.A. y Fischer, M.J.*
The string-to-string correction problem

J. Assoc. Comput. Machinery, **21**:1, 168–173, Enero 1974.

[Yun, 76] *Yunk, T.P.*
A technique to identify nearest neighbors.

IEE Transactions On S.M.C., **SMC-6**:10, Octubre 1976.