Universitat d'Alacant
Universidad de Alicante

Aportaciones a la mejora de la eficiencia de la búsqueda del vecino más cercano

Eva Gómez Ballester

Tesis Doctorales

www.eltallerdigital.com

UNIVERSIDAD de ALICANTE

Universitat d'Alacant
Universidad de Alicante

— TESIS DOCTORAL —

Aportaciones a la mejora de la eficiencia de la búsqueda del vecino más cercano

Eva Gómez Ballester

Dirigida por:

María Luisa Micó Andrés
Jose Oncina Carratalá

DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS

2012

A Adriana
A mi familia

Agradecimientos

Han pasado muchos años desde que comencé la tesis y, por lo tanto, hay mucha gente a la que dar las gracias. Probablemente me olvidaré de nombrar a algunas personas, y espero que me perdonen.

Tengo que dar las gracias, por un lado, a muchas personas de mi ámbito laboral[1] y, por otro, a las del ámbito familiar. Aunque es cierto que pasamos tanto tiempo en el trabajo que en muchas ocasiones es difícil saber dónde termina uno y dónde comienza el otro. Para mí lo importante no es el orden de la lista de agradecimientos, sino el hecho de que les tengo que estar muy agradecida.

Mi más sincero agradecimiento a Luisa y Jose por todo lo que me han enseñado, por su disponibilidad para ayudarme en todo lo que he necesitado, más allá de lo que cabe esperar, y por su infinita paciencia conmigo. Soy consciente de que sin ellos, sin sus valiosos comentarios, sin sus consejos, esta tesis nunca se hubiera terminado. Nunca les estaré lo suficientemente agradecida.

Mi agradecimiento en general para mis compañeros del departamento por su interés y sus palabras de ánimo y, en particular, a mis compañeros de unidad docente encabezados por Armando Suárez como coordinador, siempre intentando facilitar las cosas. Mi agradecimiento también para mis compañeros del Grupo de Reconocimiento de Formas e Inteligencia Artificial. Mil gracias a Miguel Ángel Varó y a Miguel Ángel Baeza, nuestros técnicos del departamento, que siempre han estado cuando les he necesitado y además con buen humor. También agradecer a Paco Moreno su implementación del kd-tree y sus comentarios, así como los comentarios de Paloma Moreda.

Muchísimas gracias a mi gran familia. A mis padres, Reme y Paco, por su esfuerzo para permitirme tener una educación, por su apoyo incondicional siempre y por su respeto a la forma de vida que he elegido. A mi hija Adriana por su paciencia, ya muy pequeña la palabra tesis formaba parte de su re-

[1] Gracias al Ministerio de Ciencia e Innovación español por su apoyo a través del proyecto TIN2009-14205-C04-01, Red de Excelencia Pascal, y del programa CONSOLIDER INGENIO 2010 (CSD2007-00018).

ducido vocabulario, gracias también por su cariño y por su vitalidad que, por suerte, muchas veces es contagiosa. A mi hermano Ezequiel porque desde su situación tan especial me ha enseñado a valorar todas esas pequeñas cosas a las que les restamos importancia en nuestra acelerada vida. A mis hermanas Carol, Nuria y Paloma porque es muy tranquilizador y gratificante saber que estáis siempre ahí. A mis cuñados por su apoyo, sus bromas y sus risas. A mis tías y primas por sus gestos de interés y de cariño.

Quiero agradecerle a Emiliet que sea tan buen padre. La tranquilidad que me proporciona saber que el tiempo que no puedo estar con Adriana, ella está disfrutando con él, constituye para mí un apoyo fundamental. Gracias por tu tiempo y tu dedicación. Gracias también a Beverly, Emilio, Patricia y Cristina, por acogerme tan cariñosamente en su familia y por su comprensión en muchas situaciones.

Gracias a mis amigas Adrienn, Elena y Natalia por los ratos de lloros y de risas. Gracias a Chema por su apoyo y aliento en unos momentos difíciles para mí. Gracias a José Ramón que siempre ha tenido palabras de ánimo.

Para terminar quería volver a dar las gracias a mis padres por su cariño y dedicación durante toda mi vida; y a Luisa, además de por su papel como codirectora de la tesis, por algo que para mí es mucho más importante, por su amistad.

Muchas gracias a todos por el privilegio que para mí supone que forméis parte de mi vida.

Índice general

Índice de figuras

Índice de Tablas

Lista de algoritmos

Símbolos y acrónimos

x	muestra para la que se quiere encontrar el vecino más cercano
nn	vecino más cercano
d_{nn}	distancia al vecino más cercano
p, p_i	prototipos
S	conjunto de prototipos
t	nodo del árbol
S_{t}	conjunto de prototipos del nodo t
c_{t}	prototipo representante del nodo t
r_{t}	radio del nodo t
hi_{t}	hijo de la izquierda de t
hd_{t}	hijo de la derecha de t
SMM	(Sibling Median Median) tipo de árbol descrito en el capítulo 4
SFF	(Sibling Far Far) tipo de árbol descrito en el capítulo 4
SMF	(Sibling Median Far) tipo de árbol descrito en el capítulo 4
PRF	(Pather Random Far) tipo de árbol descrito en el capítulo 4
PMR	(Parent Median Far) tipo de árbol descrito en el capítulo 4
PMM	(Parent Median Median) tipo de árbol descrito en el capítulo 4
SAT	(Spatial Approximation Tree) tipo de árbol descrito en el capítulo 3
VPT	(Vantage Point Tree) tipo de árbol descrito en el capítulo 3

FNR (Fukunaga Narendra Rule) regla de eliminación descrita en el capítulo 5

SBR (Sibling Based Rule) regla de eliminación descrita en el capítulo 5

GR (Generalized Rule) regla de eliminación descrita en el capítulo 5

TR (Table Rule) regla de eliminación descrita en el capítulo 5

Parte I

Introducción y conceptos teóricos

Capítulo 1

Introducción

La información es la base de muchos momentos de nuestra vida cotidiana: la compra en el supermercado, la salud, las transacciones comerciales, el ocio, etc. Muchos de ellos conllevan consultar información, cuyo almacenamiento y tratamiento determinarán la utilidad y fiabilidad de las respuestas y, por lo tanto, lo provechosa o no que sea esta información. El objetivo de almacenar información es poder realizar posteriormente una búsqueda o consulta en la información almacenada para obtener una respuesta a una pregunta concreta. En términos informáticos, estaríamos hablando de *Recuperación de Información* (Salton and McGill, 1986), área de investigación que está recibiendo especial interés en los últimos años, entre otras cosas, por la aparición de los motores de búsqueda. Por ejemplo, si observamos el uso de buscadores como *Google*, un simple vistazo a las cifras referidas al número de búsquedas que reciben sirve para darse cuenta del éxito de este tipo de herramientas, 1 billón de búsquedas al día [1].

Por otro lado, en el día a día nos encontramos con sistemas que reconocen automáticamente la matrícula de nuestro coche al entrar en un parking, sistemas biométricos capaces de reconocer una huella dactilar o una cara en tiempo real como, por ejemplo, el sistema recientemente presentado en Japón y desarrollado por Hitachi Kokusai Electric, que es capaz de reconocer una cara entre treinta y seis millones en un segundo. Estos sistemas se emplean a diario tanto en aplicaciones lúdicas (sistemas de organización y tratamiento de fotografías, como Picasa) como en temas de seguridad [2]. Son casos en los que la información se utiliza para realizar *Reconocimiento de Formas* (Duda et al., 2000), área de investigación también ampliamente extendida debido a

[1] http://www.jeffbullas.com/2011/05/16/50-amazing-facts-and-figures-about-google/

[2] Por ejemplo, según un portavoz de Scotland Yard se utilizaron sistemas de reconocimiento facial en los recientemente concluidos Juegos Olímpicos de Londres 2012.

la diversidad de aplicaciones a las que se pueden aplicar sus técnicas.

Se podrían poner multitud de ejemplos más, la asistencia casi desapercibida que nos proporciona un teléfono móvil al escribir un SMS (Dunlop and Crossan, 2000), la asistencia que nos presta un vehículo a la hora de aparcarlo (Paromtchik, 2004), las aplicaciones que ayudan a los médicos a dar un diagnóstico (Begun y Devi, 2011), y se podrían seguir dando ejemplos de aplicaciones dentro del campo de la *Recuperación de Información*, del *Reconocimiento de Formas*, de la *Minería de Datos* (Zhu and Davidson, 2007), de la *Biomedicina* (Najarian and Splinter, 2012), o de la *Compresión de audio y video* (Waggoner, 2009). En muchas de estas aplicaciones nos encontramos un problema común a todas ellas y fundamental para lograr su objetivo de manera eficaz: el problema de la *búsqueda de una respuesta* entre gran cantidad de información.

A diferencia de las bases de datos tradicionales, donde la información almacenada estaba representada por características simples, con el desarrollo de la tecnología informática (CPU's más rápidas, memorias de gran capacidad, mejores comunicaciones), la información almacenada en bases de datos ha pasado a ser más compleja en naturaleza y forma. Hemos pasado de introducir en una aplicación un número de DNI esperando obtener como respuesta el nombre y fecha de ingreso de un empleado, a introducir en las aplicaciones actuales una fotografía y esperar que la aplicación reconozca al empleado del que se trata, nos devuelva su DNI y le permita el acceso a una sala. Se ha pasado de trabajar con aplicaciones cuyos objetos de interés eran cadenas de caracteres o valores numéricos, y en cantidades no muy elevadas, que podían ser representados por estructuras sencillas (arrays, listas, etc.), a emplear sistemas que trabajan con imágenes, videos, música, distintos formatos de documentos de texto, y en cantidades muchísimo más elevadas.

En este escenario, el problema que aparece entonces es que para poder tener almacenada de la mejor forma posible este nuevo tipo de información, normalmente va a ser necesaria una representación vectorial de muy alta dimensionalidad, o la definición de nuevos tipos de datos complejos de representación de los objetos (árboles, grafos, histogramas, etc), en los que, en muchos casos, podrán aplicarse técnicas en las que sólo pueda utilizarse una medida de distancias entre los objetos (Zezula et al., 2006).

Volviendo a la recuperación (o búsqueda) de información, otro problema con el que nos encontramos es que el concepto de "búsqueda exacta" no tendrá mucho sentido debido al tipo de información almacenada (por ejemplo, las imágenes o el sonido almacenado no coincidirán exactamente con la consulta). Por lo tanto, en este caso, tiene mayor sentido el concepto de "proximidad" entre los objetos. Hablamos entonces de la *búsqueda por similitud*

donde, utilizando los conceptos de proximidad o similitud, se desemboca en búsquedas más eficaces. En una *búsqueda por similitud* la respuesta a una consulta es el objeto (u objetos) de la base de datos que más se "parece" o "parecen" al objeto de la consulta.

Aunque en la literatura existen muchas y variadas formas de realizar la búsqueda, en esta tesis nos centraremos en la *búsqueda en espacios métricos*. Este tipo de búsqueda es muy interesante por su generalidad, ya que lo único que se necesita definir entre los objetos entre los que se va a realizar la búsqueda es una medida de *proximidad* que normalmente llamaremos *distancia*. A su vez, este tipo de búsqueda cumple la propiedad de extensibilidad porque, definida una técnica concreta, la misma podrá aplicarse a problemas muy diferentes, ya que lo único que vamos a tener que cambiar es la medida de *proximidad*.

Un ejemplo sencillo es la similitud entre palabras (strings), donde lo normal es medir su similitud en función del número de operaciones (inserciones, borrados y sustituciones de símbolos de la cadena) necesario para transformar una palabra en otra (Levenshtein, 1966). Esta similitud entre palabras se usa en aplicaciones que utilizamos cotidianamente como son los editores de texto. Otro ejemplo es la similitud entre árboles que representan el esqueleto de un objeto. En este caso las operaciones a realizar para medir la similitud serán la inserción, borrado y sustitución de nodos del árbol (Zhang and Shasha, 1989).

La necesidad, cada vez mayor, de trabajar con grandes volúmenes de datos en muchas de las áreas anteriormente comentadas, nos lleva a un nuevo problema a resolver, no menos importante que la representación más o menos compleja de los objetos o el tipo de búsqueda a realizar, y es la eficiencia a la hora de realizar búsquedas. No sólo necesitamos realizar la búsqueda, sino que queremos una respuesta lo antes posible.

Figura 1.1: En multitud de aplicaciones utilizadas en el día a día se emplean técnicas sobre las que se centra el estudio desarrollado en esta tesis.

Objetivos

En general, las técnicas propuestas para acelerar la búsqueda suelen almacenar en preproceso información relacionada con el problema, que normalmente será utilizada posteriormente. Durante la búsqueda, y utilizando esta información, se toman decisiones para reducir el tiempo de respuesta a la consulta realizada.

Los objetivos específicos de esta tesis son:

- proponer nuevas estructuras para construir el índice. En particular, todas las estructuras propuestas se basan en la construcción de un árbol en el que se analizarán diferentes propiedades como su profundidad, su equilibrio, el solapamiento entre sus nodos, etc;
- proponer nuevas estrategias de poda en el árbol que permitan acelerar la búsqueda. Dependiendo de la estrategia, se analizará el tipo de información almacenada y las ventajas de su uso;
- proponer una combinación de uso eficiente de las estrategias de poda propuestas que permitan acelerar la búsqueda y decidir en qué contextos son útiles cada uno de ellas o combinaciones concretas de las mismas.

Desarrollo de la tesis

Esta memoria se estructura en dos partes. En la primera de ellas se realiza una revisión de los conceptos relacionados con su contenido, básicamente con las técnicas de búsqueda por similitud en espacios métricos, y se hace un repaso de las técnicas ya existentes en el mismo ámbito de trabajo. En la segunda parte se presentan los nuevos métodos que se han propuesto en esta tesis, tanto para la construcción del árbol como para la aceleración de la búsqueda.

Los algoritmos y técnicas desarrollados en esta tesis están destinados principalmente, en el contexto de la *búsqueda por similitud*, a *acelerar la búsqueda* del *vecino más cercano* en *espacios métricos*. Para centrar el tema, en la primera parte se expondrán algunos conceptos teóricos de la *búsqueda por similitud* en espacios métricos (capítulo 2), dirigiendo la atención hacia el método de *búsqueda del vecino más cercano*, recordando en qué consiste y revisando algunos de los algoritmos más representativos que se han propuesto para realizar esta tarea de un modo eficiente (capítulo 3). De entre todos estos algoritmos, se presentan con mayor detalle los más representativos en la literatura que trabajan con estructuras de datos de características

similares a los propuestos. De este modo se podrán posteriormente realizar comparaciones entre resultados, y analizar los motivos que pueden justificar los resultados obtenidos.

En la segunda parte de esta memoria se presentan los nuevos métodos que se han desarrollado. Tal y como se ha indicado anteriormente, los métodos propuestos tienen como objetivo *acelerar la búsqueda* del *vecino más cercano* cuando se utilizan *estructuras arborescentes* para almacenar el índice. Ya que para conseguir este objetivo general en el apartado anterior nos hemos propuesto dos objetivos específicos, este estudio se desarrolla desde dos frentes que se corresponden con los capítulos 4 y 5 de esta tesis.

En el capítulo 4 se detallan las estrategias que se han estudiado para construir el árbol donde almacenar los objetos. Se detallan los pasos que se han seguido para construir cada estructura de árbol propuesta, examinando cómo queda dividido el espacio de los datos al emplear cada una de ellas.

En el capítulo 5 se definen y formalizan las reglas de eliminación que se proponen en esta tesis para acelerar el proceso de búsqueda del vecino más cercano. Los experimentos que comparan las nuevas propuestas con otros métodos conocidos se incluyen también en este capítulo.Para concluir con el estudio de las nuevas reglas se plantea además un algoritmo con la intención de combinar de modo óptimo las reglas propuestas. En el capítulo 6 se realiza un estudio experimental de las propuestas de construcción del árbol y de las reglas de eliminación utilizando datos reales.

Para finalizar, en el capítulo 7, se presentan las conclusiones sobre el trabajo realizado y se plantean algunas de las propuestas que quedan abiertas para continuar con su estudio en el futuro.

Capítulo 2

Búsqueda por similitud

La búsqueda, ya sea simple o compleja, es un problema fundamental para un gran número de aplicaciones informáticas (Chávez et al., 2001). Tradicionalmente la búsqueda se realizaba sobre datos estructurados, por ejemplo, datos numéricos o cadenas de caracteres, donde dada una *muestra* que es objeto de la *consulta*, se busca en una base de datos aquel dato que sea igual a la *muestra* dada. Esto nos lo encontramos en las bases de datos tradicionales, como las relacionales (Codd, 1970), que están orientadas a una búsqueda por *coincidencia exacta*.

Por ejemplo, en una base de datos relacional que almacene información sobre los empleados de una empresa, una consulta podría consistir en saber qué empleados tienen una determinada categoría, cobran más de una cierta cantidad o llevan en la empresa más de un número de años. Serían consultas fáciles de responder. Pero si en esta base de datos almacenamos una foto de cada empleado, no sería tan fácil responder a qué empleado corresponde una determinada imagen que se da como entrada, entre otros motivos porque la imagen puede no ser exactamente la misma (condiciones diferentes de iluminación, oclusión, diferencias cromáticas, etc). Lo más usual en este caso es responder a consultas por similitud a la muestra dada.

La utilidad y definición del concepto de *similitud* es un tema ampliamente estudiado y que juega un papel muy importante en diversas áreas. El estudio del término *similitud* tiene su origen en la filosofía y la psicología y se utilizó para determinar cómo y porqué los objetos son agrupados en categorías, y el motivo por el que algunas categorías son comparables y otras no. Hace más de 40 años Quine (Quine, 1969) indicaba que

> *"Similarity is fundamental for learning, knowledge and thought, for only our sense of similarity allows us to order things into kinds so that these can function as stimulus meanings. Reasonable expectation depends on the similarity of circumstances and on our tendency to expect that similar causes will have similar effects".*

Descubrir lo que es similar o no para las personas es fundamental, ya que los objetos similares tienden a tener comportamientos similares. Tendemos a basarnos en la *similitud* para generar inferencias y para establecer categorías de objetos (Goldstone and Son, 2005).

En este capítulo se introducirá el concepto de *similitud en espacios métricos*, se definirán diferentes tipos básicos de *búsqueda por similitud*, y se mostrará un abanico de aplicaciones informáticas en las que se utiliza la *búsqueda por similitud*. Por último, se hablará sobre las estructuras y métodos de acceso que permiten realizar estas búsquedas de manera eficiente.

2.1. Búsqueda por similitud en espacios métricos

Ya que los problemas en los que se aplica la *búsqueda por similitud* pertenecen a áreas muy diferentes, la naturaleza de los objetos sobre los que se va a aplicar la búsqueda es también muy diversa, y será importante el modo en que se represente el espacio de las muestras objeto de estudio. ¿Qué es lo que queremos representar?, ¿qué propiedades son importantes para distinguir estos objetos? y ¿cómo representamos esas propiedades?

Ya hace casi un siglo, Campbell escribía afirmaciones como éstas (Campbell, 1920):

> *"Meassurement is the assignment of numerals to represent properties. Why is the process important and why is it applicable to some properties (e.g. weight) and not to others (e.g. colour)?"*
> ...
> *"the object of measurement is to enable the powerful weapon of mathematical analysis to be applied to the subject matter of science"*
> ...

Sobre el proceso de medir propiedades de los objetos, varios autores han mostrado su acuerdo o desacuerdo con las propuestas de otros investigadores. Por

ejemplo, en el libro de C.W. Savage, titulado "The measurement of sensation: a critique of perceptual psychophysics" (Savage, 1970) se hace referencia a (Campbell, 1920) y a (Stevens, 1946):

> *"Suppose measurement is narrowly defined as the assignment of numerals to things by the physical addition of dimension units (Campbell's definition). It then follows that pitch is not measurable, since there is no physical operation of addition for pitch. Suppose, on the other hand, measurement is broadly defined as the assignment of numerals to things by some rule or other (Stevens' definition). It then follows that pitch is measurable, since numerals can be assigned to sounds according to the rule: to the higher pitch assign the higher numeral. Chapters 4 and 5 critize both unacceptably narrow and unacceptably broad definitions of measurement, and chapter 5 recommends that measurement be defined as the assignment of numerals to things by comparing the things (by addition or some other operation) with units."*

Representar los objetos a través de las mediciones de algunas de sus propiedades y encontrar la función distancia que nos permita, en base a estas medidas, establecer la similitud entre ellos, no es una tarea trivial. Y esta tarea se complica aún más cuando estas propiedades no se pueden medir de un modo numérico, sino que algunas de ellas toman valores categóricos y, por lo tanto, los objetos no son representables a través de un vector en el espacio euclídeo. Incluso en otras ocasiones a los objetos se les aplica una función distancia para medir su similitud sin que estén representados en un espacio vectorial. Por lo tanto, es necesario el desarrollo de técnicas en espacios métricos en general, donde lo único que se necesita es una distancia para poder realizar la búsqueda.

El concepto de *espacio métrico* fue introducido por Fréchet en 1906 (Fréchet, 1906). Formalmente (Kolmogorov and Fomin, 1954), un espacio métrico está formado por

- un universo de objetos M
- y una función distancia (también llamada métrica)

  $d : M \times M \to \mathcal{R}$ (donde $\mathcal{R}$ es el conjunto de los números reales)

Para todo $x, y, z \in M$, esta función debe satisfacer las siguientes propie-

dades:

1.	$d(x, y) \geq 0$	Positividad
2.	$d(x, x) = 0$	Reflexividad
3.	$d(x, y) > 0$ (if $x \neq y$)	Positividad estricta
4.	$d(x, y) = d(y, x)$	Simetría
5.	$d(x, z) \leq d(x, y) + d(y, z)$	Desigualdad triangular

Hay que tener en cuenta que si el espacio no es un espacio métrico debido a que la función distancia incumple alguna de las propiedades, pero mantiene las 2 primeras, se puede asociar una nueva función distancia de un modo sencillo, que sí sea una métrica (Clarkson, 2006) y, por lo tanto, ya se estaría trabajando en un espacio métrico.

Si la condición 3 falla, se dice que (M, d) es una *pseudométrica*. En un espacio pseudométrico podemos encontrar diferentes objetos a distancia 0 entre sí. Estos espacios se pueden adaptar fácilmente, suponiendo que todos los objetos que están a distancia 0 entre sí son el mismo objeto, es decir, particionando el espacio M en clases de equivalencia. Dos objetos, p_1 e p_2, pertenecen a la misma clase si $d(p_1, p_2) = 0$. Si se considera ahora la distancia entre dos clases de equivalencia como la distancia entre dos objetos que pertenecen a cada una de ellas, estamos trabajando en un espacio métrico.

Si la que falla es la condición 4, la simetría, el espacio se dice que es *cuasimétrico*. Podemos suponer como ejemplo el recorrido de un autobús de línea de una ciudad entre dos paradas consecutivas A y B, y que la distancia esté en función del recorrido por las calles, teniendo en cuenta que el sentido del tráfico por las calles puede hacer que no se recorran las mismas calles para ir de A a B que de B a A. En este caso, existen técnicas para derivar una nueva distancia simétrica desde una asimétrica, por ejemplo las dos propuestas siguientes:

$$d'(x, y) = d(x, y) + d(y, x) \quad \text{(Zezula et al., 2006)}$$
$$d'(x, y) = (d(x, y) + d(y, x))/2 \quad \text{(Clarkson, 2006)}$$

En el resto de esta tesis usaremos un conjunto finito de objetos $S \in M$ de tamaño $n = |S|$. Al conjunto de elementos de S lo llamaremos *conjunto de prototipos*, o *base de datos*. El término *distancia* lo utilizaremos para referirnos a una métrica.

Nomenclatura

Al hablar de la búsqueda por similitud tanto al repasar las técnicas propuestas por otros autores como al presentar nuestras propuestas, además del concepto de *espacio métrico*, se repiten una serie de términos cuya definición y simbología se indican a continuación:

- *prototipo*: es cada uno de los objetos de la base de datos donde se realizan las búsquedas para dar respuesta a una consulta. Normalmente lo denotaremos con la letra p para referirnos a un prototipo o p_1, p_2, ..., p_k para referirnos a la vez a varios;
- *muestra*: es el objeto sobre el que se realiza la consulta. Lo denotaremos con la letra x;
- vecino más cercano: será el prototipo en S más similar a la muestra objeto de la consulta. Nos referiremos a él como nn.

2.2. Tipos de búsqueda por similitud

La forma de dar la respuesta es distinta según queramos realizar *una búsqueda por coincidencia exacta* o *una búsqueda por similitud*.

La *búsqueda por coincidencia exacta* divide el conjunto de prototipos almacenados en dos subconjuntos, uno en el que están los prototipos que satisfacen la consulta y otro que contiene los que no satisfacen la consulta. La *búsqueda por similitud* ordena el conjunto de prototipos almacenados en relación a su similitud a la muestra dada en la consulta.

En el caso de la *búsqueda por similitud*, para reducir el tamaño del conjunto resultado se consideran generalmente dos técnicas:

- tener en cuenta aquellos prototipos cuya similitud a la muestra objeto de la consulta está por debajo de un determinado umbral especificado, o
- indicar el número máximo de prototipos que se espera como respuesta.

En base a esta consideración existen *varios tipos básicos de consulta por similitud* (Zezula et al., 2006; Hjaltason and Samet, 2003):

- Una consulta *por rango* consta de una muestra objeto de la consulta, y de un umbral de similitud. La respuesta a la consulta es el conjunto de todos los prototipos almacenados cuya similitud a la muestra objeto de la consulta es menor que el umbral (fig. 2.1(a)). Si el umbral fuese 0 estaríamos ante una búsqueda por coincidencia exacta.

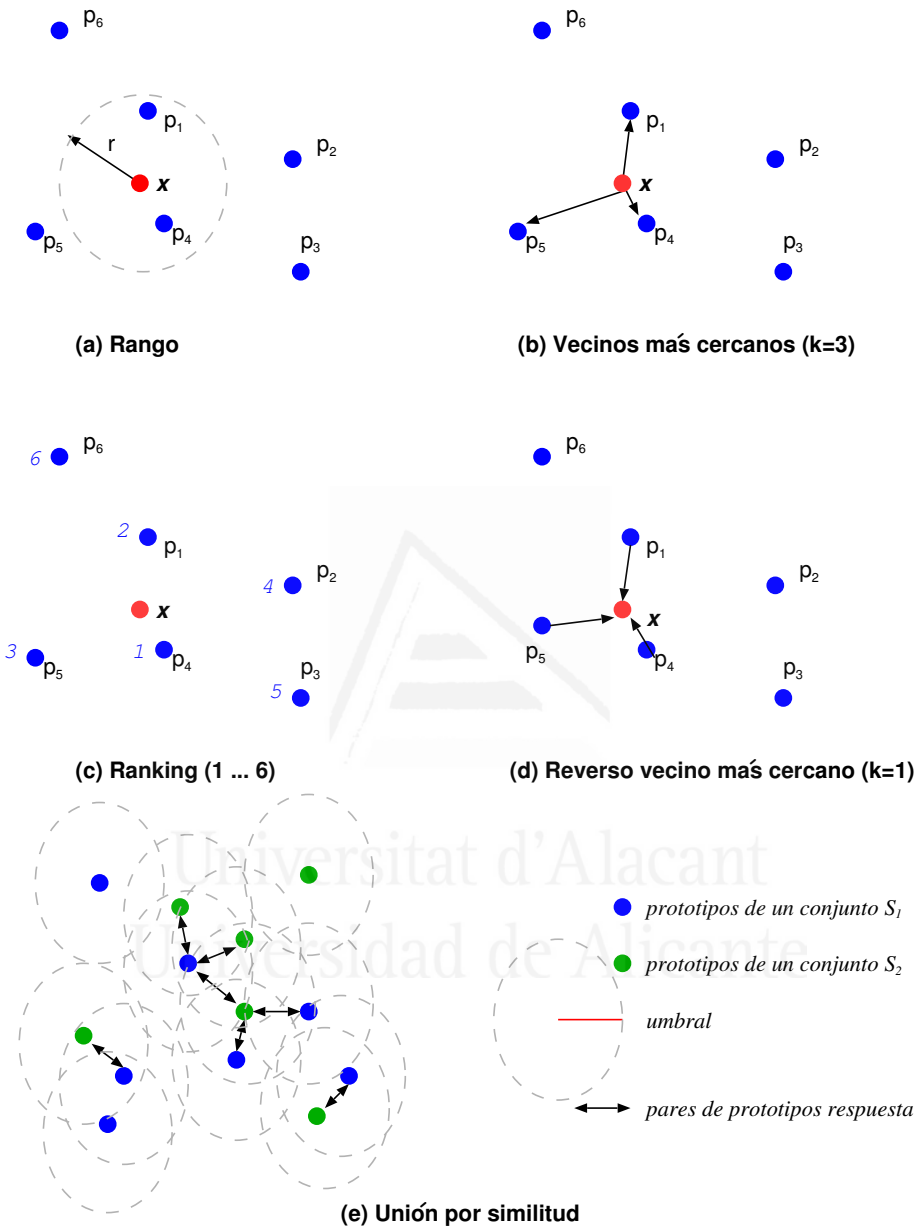


Figura 2.1: Dada una muestra x, la búsqueda por rango (a) de radio r devuelve como respuesta los prototipos p_1 y p_4. La búsqueda de los vecinos más cercanos (b) con k=3 da como respuesta p_1, p_4 y p_5. La búsqueda por ranking da la ordenación de los prototipos que se muestra en (c). La búsqueda del reverso del vecino más cercano con k=2 devuelve todos los prototipos excepto p_3. En el caso de la unión por similitud (e), en lugar de considerar un conjunto de prototipos y una muestra, se consideran dos conjuntos distintos de prototipos.

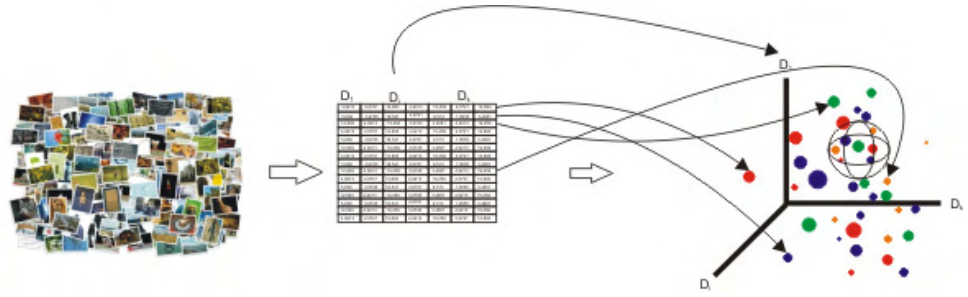


Figura 2.2: Imagen extraida de `http://gim.unex.es`

- Una consulta de los *k-vecinos más cercanos* consta de una muestra objeto de la consulta y de un valor k que indica el número máximo de prototipos que se deben obtener como respuesta. La respuesta a la consulta es el conjunto de los k prototipos más similares a la muestra objeto de la consulta (fig. 2.1(b)). La búsqueda del vecino más cercano es el caso particular para $k = 1$.

- Una consulta *por ranking* consta de una muestra objeto de la consulta. La respuesta a la consulta es el conjunto de todos los prototipos almacenados ordenados por su similitud a la muestra objeto de la consulta (fig. 2.1(c)), pudiendo utilizarse algún parámetro para delimitar el tamaño de la respuesta.

- Una consulta de *k-vecinos reversos más cercanos* consta de una muestra objeto de la consulta y de un valor k, y devuelve aquellos prototipos almacenados para los que la muestra objeto de la consulta es uno de sus *k-vecinos más cercanos* (fig. 2.1(d)).

- Una *unión por similitud (similarity join)* entre dos conjuntos devuelve todos los pares de prototipos, uno de cada conjunto, cuya distancia entre sí no excede de un umbral dado como entrada (fig. 2.1(e)).

Independientemente del tipo de consulta que se realice, la calidad de la respuesta en una búsqueda por similitud va a estar condicionada a la elección de una función de disimilitud[1] adecuada al dominio de aplicación. En algunas aplicaciones la distancia euclídea es una elección adecuada para medir la similitud entre objetos, sin embargo, en muchas otras aplicaciones, la elección de una función distancia o de disimilitud no es tan obvia.

[1]Para establecer los similares que son dos prototipos nos apoyamos en un afunción que mide la distancia entre ellos, su disimilitud.

Además de la calidad de la respuesta hay otro factor crucial en la búsqueda por similitud, y éste es la eficiencia del proceso, es decir, el *tiempo necesario* para ofrecer una respuesta adecuada a la consulta. Este tiempo de respuesta se ve condicionado por el tamaño del conjunto de prototipos con el que se trabaja que, en la mayoría de las aplicaciones, suele ser grande. Para intentar minimizar este tiempo de respuesta los algoritmos de búsqueda deben evitar la exploración de todo el espacio de trabajo, evitando recorrer zonas del espacio donde no es posible que se encuentre la solución a la consulta. Para evitar la exploración completa, se aplican reglas que se basan en las propiedades geométricas del espacio.

Uno de los objetivos de esta tesis es el estudio de métodos que dividen el conjunto de prototipos que constituyen el espacio de trabajo en una serie de subconjuntos de forma recursiva atendiendo a unas propiedades que permiten, posteriormente, realizar la *búsqueda del vecino más cercano* evitando la exploración completa del espacio de trabajo. Ya que se va a emplear la técnica del vecino más cercano, a continuación se definirá en qué consiste esta técnica.

2.2.1. Búsqueda del vecino más cercano

Su simplicidad y eficacia hace que haya sido objeto de numerosos estudios (Chávez et al., 2001; Clarkson, 2006; Shakhnarovich et al., 2006).

La definición y uso del vecino más cercano (*1-NN*) en tareas de clasificación fue propuesta por Fix y Hodges en 1951 (Fix and Hodges, 1951). Esto se debió a la necesidad de poder realizar análisis discriminantes cuando la estimación paramétrica de funciones de densidad de probabilidad era imposible o difícil de obtener. Por ello propusieron una técnica no paramétrica que fue denominada la regla del vecino más cercano *1-NN*.

Más tarde, en 1967, se estudiaron algunas propiedades de esta regla. De hecho se demostró que para $n \to \infty$, siendo n el tamaño de la base de datos, el error de clasificación de la *regla del vecino más cercano* está acotado superiormente dos veces por el error de Bayes (Covert and Hart, 1967).

Al ser inicialmente propuesto como un método de clasificación dentro del campo del *Reconocimiento de Formas*, la formulación de la *regla del vecino más cercano* en su versión original se asocia al concepto de clase en la que se clasifica una muestra.

Dado un espacio métrico (M, d) donde se ha definido una función distancia d:

- Regla de clasificación del vecino más cercano *1-NN*.

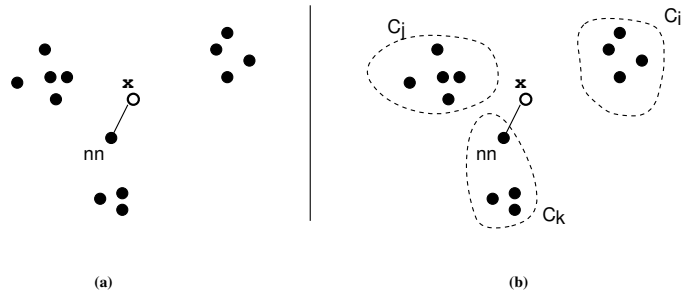


Figura 2.3: (a) Dada una muestra x, su vecino más cercano será aquel prototipo, nn, que se encuentra a menor distancia de x. (b) En una tarea de clasificación, a la muestra x se le asigna la clase C_k.

Se basa en la suposición de que la mejor clase para la muestra a clasificar, x, es la del prototipo más cercano en el conjunto de búsqueda (Duda et al., 2000).

Sea $S = \{p_1, p_2, ..., p_n\}$ un subconjunto de prototipos de M, $S \subset M$, sea $C = \{C_1, C_2, \ldots, C_m\}$ las clases en las que están clasificados estos prototipos, sea $\{S, \theta\} = \{(p_1, \theta_1), (p_2, \theta_2), \ldots, (p_n, \theta_n)\}$ donde $p_i \in S$ y $\theta_i \in C$, el conjunto de pares formados por un prototipo y la etiqueta de su clase, sea $\delta_{1\text{-}NN}$ una función que, dada una muestra, devuelve la mejor clase para la misma, y sea x la muestra a clasificar.

Entonces la *regla del vecino más cercano* se define como

$$\delta_{1\text{-}NN}(x) = \theta_i \leftrightarrow d(x, p_i) = \min_{\forall p \in S} d(x, p).$$

Es decir, la muestra x se clasificará en la misma clase en la que se encuentra clasificado el prototipo más cercano a x.

Si lo aplicamos de un modo general a la *búsqueda por similitud*, el prototipo más cercano, nn, a la muestra x quedaría definido como

$$nn = \arg\min_{\forall p \in S} d(x, p).$$

En la figura 2.3 mostramos cómo se clasifica la muestra x con la regla *1-NN* para un problema de clasificación de tres clases. En este caso el prototipo más cercano a la muestra pertenece a la clase C_k, por lo que la muestra x se clasifica en esta clase. En términos de similitud, nn sería el prototipo *más similar* a la muestra x.

- Regla de clasificación de los k vecinos más cercanos (k-*NN*)

  La regla de clasificación de los k vecinos más cercanos es una generalización de la regla anterior. En este caso la muestra se clasifica en la clase a la que pertenecen más vecinos de entre los k más cercanos.

  Sea $S = \{p_1, p_2, ..., p_n\}$ un subconjunto de prototipos de M, $S \subset M$, sea $C = \{C_1, C_2, \ldots, C_m\}$ las clases en las que están clasificados estos prototipos, sea $\{S, \theta\} = \{(p_1, \theta_1), (p_2, \theta_2), \ldots, (p_n, \theta_n)\}$ donde $p_i \in S$ y $\theta_i \in C$, el conjunto de pares formados por un prototipo y la etiqueta de su clase, sea $\delta_{k\text{-}NN}$ una función en la que dada una muestra devuelve la mejor clase para la misma, y sea x la muestra a clasificar. Si llamamos K_i, $(i = 1, \ldots, m)$, a la cantidad de prototipos, de entre los k más cercanos que pertenecen a la clase i, la muestra x se clasificará como perteneciente a la clase a la que le corresponde el máximo valor de K_i.

  $$\delta_{k\text{-}NN}(x) = \theta_i \leftrightarrow i = \arg\max_{j=1,\ldots,m} K_j.$$

  Si lo aplicamos a la *búsqueda por similitud* de los k prototipos más cercanos, nn_1, nn_2, ..., nn_k a la muestra x, éstos quedan definidos como

  $$nn_1 = \arg\min_{\forall p \in S} d(x, p)$$
  $$nn_2 = \arg\min_{\forall p \in S - \{nn_1\}} d(x, p)$$
  $$\ldots$$
  $$nn_k = \arg\min_{\forall p \in S - \{nn_i\}_{\{i=1,\ldots,k-1\}}} d(x, p).$$

En la figura 2.4 se muestra cómo se realiza la clasificación *3-NN* de la misma muestra que se utilizó como ejemplo de clasificación *1-NN* en la figura 2.3. En este caso, de entre los 3-vecinos más cercanos, 2 pertenecen a la clase C_j, 1 a la clase C_k y ninguno a la clase C_i, por lo que x se clasifica en la clase C_j.

2.2.2. Distancias

Tal y como se ha indicado en el apartado anterior, para medir la similitud entre objetos es necesario la definición de una función distancia. El concepto de distancia entre objetos nos permite interpretar geométricamente el conjunto de prototipos con el que trabajamos al posibilitar la representación de estos prototipos como puntos de un espacio métrico.

En esta tesis hemos trabajado con dos distancias: la distancia euclídea y la distancia de edición.

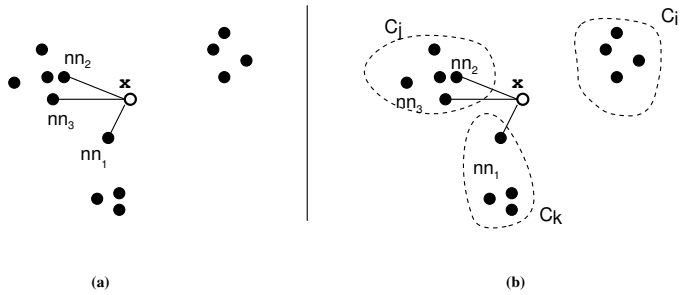


Figura 2.4: (a) Los 3-vecinos más cercanos a la muestra x serían nn_1, nn_2, nn_3. (b) Si estuviésemos clasificando atendiendo a la regla *3-NN*, la muestra x se clasificaría en la clase C_j.

Distancia euclídea

Siendo $p_1 = (p_{11}, p_{12}, \ldots, p_{1n})$ y $p_2 = (p_{21}, p_{22}, \ldots, p_{2n})$ dos prototipos de $\mathcal{R}^n$, la distancia euclídea entre ellos se define como:

$$d(p_1, p_2) = \sqrt{\sum_{i=1}^{n} (p_{1i} - p_{2i})^2}$$

Distancia de edición

La distancia de edición o distancia de Levenshtein, toma este nombre del científico ruso Vladimir Levenshtein, quien propuso en 1965 el algoritmo para calcularla (Levenshtein, 1966; Wagner and Fischer, 1974). Levenshtein definió la distancia entre dos palabras como el menor número de operaciones de edición necesarias para transformar una en otra, entendiendo por operación la inserción, borrado o sustitución de un carácter.

La distancia de edición entre cadenas es una medida de similitud que se utiliza para solucionar diversos problemas. Por ejemplo, en correctores ortográficos, donde si se encuentra en el texto una palabra que no está en el diccionario se sugieren palabras semejantes o cercanas, es decir, con una distancia de edición pequeña (Rimrott and Heift, 2011). También es interesante en estudios de ADN, donde las secuencias de ADN están representadas por cadenas sobre el alfabeto A,C,G,T, y la distancia de edición nos da una idea de lo cercanas que están dos cadenas (Somboonsak and Munlin, 2011).

2.3. Aplicaciones de la búsqueda por similitud

La *búsqueda por similitud* se utiliza actualmente con buenos resultados en una gran variedad de aplicaciones que han surgido con el avance de la tecnología (Patella and Ciaccia, 2008). No hay un modo global, independiente de la aplicación, para medir la similitud. Esto hace que sea complicado la selección de una medida de similitud para un área concreta de aplicación, así como la comparación de diferentes medidas de similitud (Janowicz and Wilkes, 2009).

Las decisiones de las personas en cuanto a la similitud están influenciadas por la edad, el lenguaje y el bagaje cultural, lo que puede jugar un papel muy importante en la interacción persona-máquina cuando se intenta definir una medida de similitud que no tenga en cuenta estos factores.

A continuación se presentan algunas de las áreas en las que se utiliza la *búsqueda por similitud.*

Bases de datos multimedia

Si es fundamental en cualquier aplicación informática dar una solución eficiente a los problemas que se derivan de la búsqueda de información, este hecho adquiere una importancia especial dentro del campo de las bases de datos, ya que precisamente la forma de almacenar y recuperar la información constituye el pilar de este área.

Las búsquedas más tradicionales suelen llevarse a cabo sobre datos estructurados en bases de datos relacionales (Codd, 1970), donde la información está organizada en tablas, y en las que cada columna representa un concepto. Los datos que tradicionalmente se almacenaban eran de tipo texto o de tipo numérico. Sin embargo, con la evolución de la tecnología, aparecen nuevos tipos de datos. En una base de datos multimedia podemos encontrar los siguientes tipos de datos [2]:

- Texto: estructurado o no estructurado.
- Gráficos: dibujos e ilustraciones codificados según un estándar de descripción de archivos (CGM, PICT, PostScript, etc).
- Imágenes: elementos gráficos codificados según algún formato estándar (mapa de bits, jpeg, png, tiff, etc).
- Animaciones: secuencias temporales de imágenes o datos (gif, swf, etc).

[2] http://es.scribd.com/doc/14870298/Bases-de-Datos-Multimedia

Figura 2.5: Encontrar las características adecuadas y establecer una función de disimilitud para comparar imágenes es una tarea complicada (imagen obtenida de (Bustos, 2009)).

- Video: secuencia temporal de imágenes a una determinada velocidad (fps).
- Audio estructurado: secuencia de sonidos identificados según nota, tono, duración, etc.
- Audio digital: secuencia de sonidos digitales.
- Datos multimedia compuestos: datos multimedia (imagen, sonido, ...) junto con información textual sobre los datos (metadatos).

Si consideramos, por ejemplo, las bases de datos donde se encuentran almacenadas grandes cantidades de imágenes, es probable que nuestro interés no resida en encontrar una imagen *exacta* a una imagen determinada dada como entrada, sino que lo que queremos es encontrar todas aquellas imágenes que tengan una *gran similitud* con una imagen determinada. Además del desafío que supone encontrar la imagen más similar, existe un componente semántico en las imágenes fácilmente comprensible para un humano pero no para un ordenador (fig. 2.5).

Dada la naturaleza tan heterogénea de los datos almacenados en una base de datos multimedia, es necesario realizar un análisis de estos datos para, además de almacenarlos, conseguir abstraer una estructura que permita aplicar una búsqueda por similitud (fig. 2.6). Las características de esa búsqueda por similitud, por ejemplo, la función de disimilitud, dependerán de la aplicación concreta (Bustos, 2006).

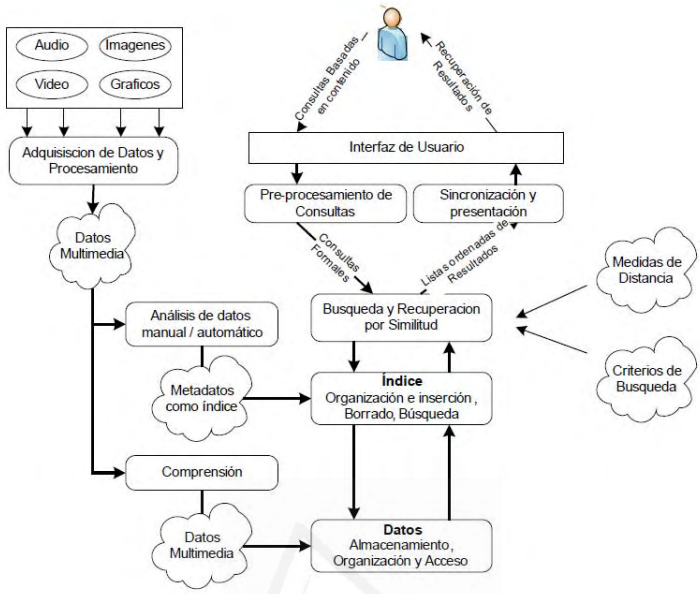


Figura 2.6: Base de datos multimedia

Biología computacional o Bioinformática

Aparte de las definiciones formales de organismos o instituciones de referencia [3], los manuales y libros sobre esta materia aportan sus propias definiciones. Por ejemplo, David W. Mount, en su difundido texto sobre bioinformática (Mount, 2004), precisa que:

> *"... la bioinformática se centra más en el desarrollo de herramientas prácticas para la gestión de datos y el análisis (por ejemplo, la presentación de información genómica y análisis secuencial), pero con menor énfasis en la eficiencia y en la precisión."*

Por otra parte, y según el mismo autor:

[3]Los términos bioinformática y biología computacional son utilizados a menudo como sinónimos, ya que, aunque existen áreas de aplicación propias de cada término, no hay definiciones que eviten completamente el solapamiento entre las actividades de estas técnicas (http://es.wikipedia.org/wiki/Bioinformática).

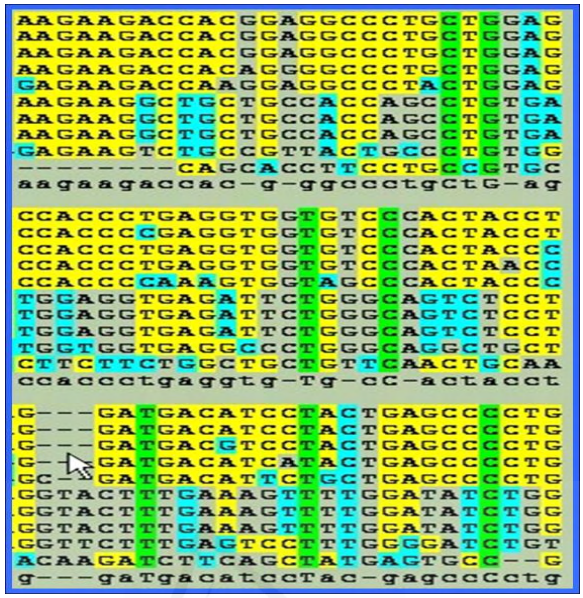


Figura 2.7: La forma más simple de comparar dos secuencias de ADN es alinearlas, insertando caracteres de hueco para hacer que estén en concordancia vertical.

> *"...la biología computacional generalmente se relaciona con el desarrollo de algoritmos nuevos y eficientes, que se puede demostrar funcionan sobre un problema difícil, tales como el alineamiento múltiple de secuencias o el montaje (o ensamblado) de fragmentos de genoma. Por último, se encuentra en ocasiones una categorización explícita de estos conceptos según la cual la bioinformática es una subcategoría de la biología computacional."*

La conexión entre la Informática y la Biología (Guigó, 2003)(Guigó and Gusfield, 2002) se remonta a finales de los años 60 cuando los ordenadores empezaron a estar al alcance de la investigación en universidades y centros especializados. Con la popularización de los lenguajes de programación de alto nivel y la mayor disponibilidad de ordenadores, la computación empezó a convertirse en parte habitual de la práctica científica y, en concreto en el caso de la Biología, el efecto fue mayor en campos como la Genética, la Ecología o la Fisiología, en los que el análisis estadístico o la formalización matemática juegan un papel importante. Los ordenadores permitían realizar análisis mucho más exhaustivos, por ejemplo, de las secuencias de aminoácidos, y se pudo comprobar que proteínas de función similar mostraban también una

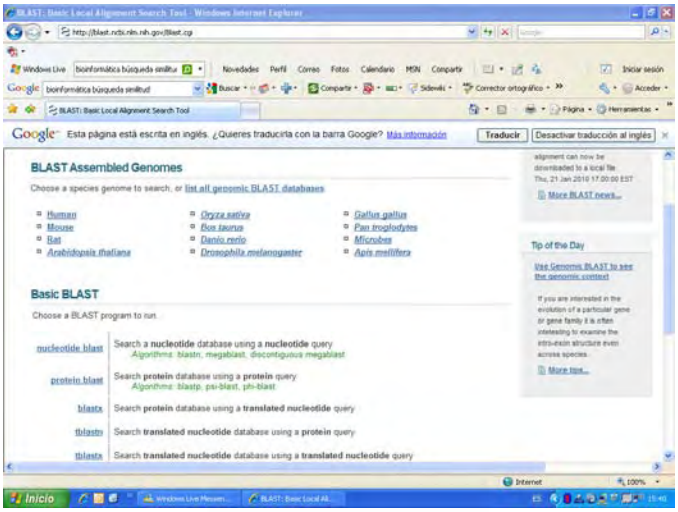


Figura 2.8: El algoritmo BLAST encuentra las secuencias de la base de datos que tienen *mayor parecido* a la secuencia problema. El artículo que describe el programa BLAST (Altschul et al., 1990) fue el más citado en Biología durante la década de los noventa, lo que refleja la enorme importancia de este algoritmo.

secuencia similar, y que similaridad de secuencia implicaba también proximidad filogenética.

A principios de los años setenta el número de proteínas para las que se había obtenido la secuencia de aminoácidos no dejaba de aumentar, no así la de ácidos nucleicos de los que sólo se habían determinado una veintena. Fue a mediados de los años setenta cuando se ponen a punto finalmente métodos practicables de secuenciación de ácidos nucleicos.

A principios de los años ochenta, el número de secuencias de ácidos nucleicos había crecido espectacularmente. La distribución de las colecciones de secuencias ya no se podían manejar en libros en formato impreso y, en 1982, se creaba en Los Álamos National Laboratory (New Mexico) la base de datos americana de secuencias de ácidos nucleicos en formato electrónico, GenBank. Prácticamente al mismo tiempo, el European Laboratory for Molecular Biology creaba su propia base de datos de secuencias de ADN en Heidelberg, y Japón crearía más tarde el DNA Data Bank of Japan. Las tres bases de datos comparten las mismas secuencias y una estructura similar.

La existencia de estas bases de datos electrónicas de secuencias facilitó extraordinariamente su análisis computacional, y el bioquímico Russell F. Doolittle dejó sorprendidos en 1983 a muchos biólogos que estudiaban el cáncer cuando, mientras realizaba comparaciones entre las secuencias alma-

cenadas en estas bases de datos, descubrió la similitud entre la secuencia de un oncogen y la secuencia del factor de crecimiento derivado de plaquetas, una relación que había pasado desapercibida anteriormente y que contribuía substancialmente a la comprensión de los mecanismos moleculares involucrados en el cáncer. Éste y otros resultados en los que la función de un gen era, al menos en parte, inferida a partir de la *similitud* de su secuencia con secuencias de función conocida, demostraron la importancia de la Biología Computacional.

Sin embargo, a medida que aumentaba el tamaño de las bases de datos de secuencias, los algoritmos de programación dinámica desarrollados por Needleman y Wunsch (Needleman and Wunsch, 1970), y Smith y Waterman (T. and M., 1981) eran demasiado lentos para llevar a cabo *búsquedas de similitud* entre una nueva secuencia y las secuencias previamente almacenadas en las bases de datos. Programas como FASTA (Pearson y Lipman, 1988) y BLAST (Altschul et al. 1990) resolvieron este problema mediante la utilización de algoritmos que proporcionaban alineamientos generalmente muy aproximados al alineamiento óptimo y que eran mucho más rápidos. La clave fue la construcción de un índice de la base de datos de secuencias basado en las mismas secuencias.

BLAST (Basic Local Alignment Search Tool) es un programa informático (fig. 2.8) de alineamiento de secuencias de tipo local, ya sea de ADN o de proteínas. El programa es capaz de comparar la muestra objeto de una consulta contra una gran cantidad de secuencias que se encuentran en una base de datos. El algoritmo encuentra las secuencias de la base de datos que tienen *mayor parecido* a la secuencia problema.

Sistemas de información geográfica

La organización y búsqueda de información geográfica se ha convertido en un campo de estudio muy activo (Janowicz et al., 2008) gracias a la gran cantidad de información disponible en Internet. Ya que esta información surge desde y para el uso de las personas, las metodologías aplicadas para recuperar y relacionar esta información deben corresponderse con los criterios de similitud de las personas.

Muchas aplicaciones geoespaciales ofrecen la posibilidad de integrar técnicas de recuperación de información basadas en el concepto de similitud. Interfaces web, tales como la interfaz basada en similitud para la guía geográfica digital Alexandria (Janowicz et al., 2007) (fig. 2.9 y 2.10), pueden proponer características o tipos de características geográficas similares ante un requerimiento de un usuario.

En el área de la información geográfica hay varios campos de aplicación

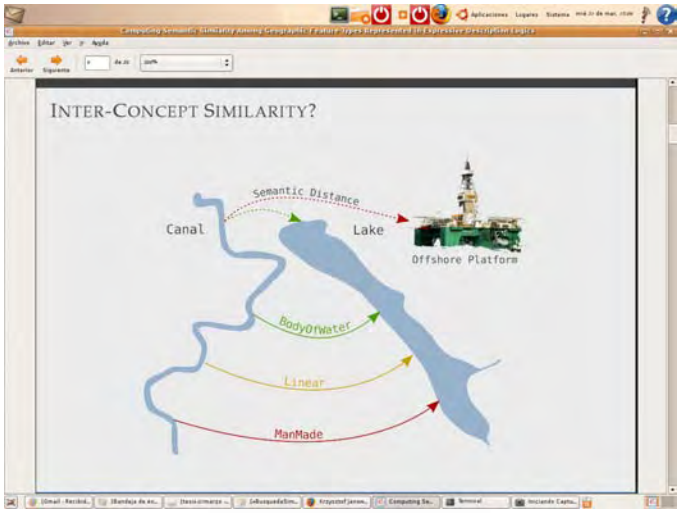


Figura 2.9: ¿Es muy similar el canal al río? (Imagen extraída de (Janowicz, 2008)).

Figura 2.10: Imagen extraída de (Janowicz, 2008) sobre búsqueda en un mapa.

del concepto de similitud. Podemos encontrar referencias a los mismos en (Janowicz et al., 2008) y en (Schwering, 2008).

La *Matching Distance Similarity Measure* propuesta en 2004 (Rodríguez and Egenhofer, 2004) fue una de las primeras medidas de similitud que se han desarrollado específicamente para el dominio geoespacial. Otros autores

también han propuesto más medidas de similitud geométrica (Raubal, 2004; Scwering and Raubal, 2005).

Predicción con series temporales

Una serie temporal es una secuencia de valores medidos en incrementos constantes de tiempo y, por lo tanto, ordenados cronológicamente. Hoy en día, en muchos campos dispares entre sí como pueden ser la medicina y el mundo de los negocios, analizar colecciones de datos ordenados en el tiempo puede ser fundamental a la hora de tomar decisiones. Estudiar cómo se ha comportado una variable hasta el momento puede ser muy importante a la hora de predecir su comportamiento futuro y, en esta misma línea, determinar qué otros valores han tenido un comportamiento similar puede ayudar a decidir las acciones que se deben llevar a cabo, bien para conservar dicha evolución o bien para modificarla (Box et al., 2008).

Como en las otras áreas citadas, la cantidad de información con la que se suele trabajar es muy elevada, por lo que no es eficaz realizar una búsqueda secuencial de dos series temporales para saber si son o no similares, y se hacen necesarias técnicas que aceleren este proceso. Tanto si lo que se pretende es buscar secuencias temporales que se parezcan entre sí, como si se va a buscar la secuencia que más se parece a otra dada como entrada, o si se pretende buscar un patrón de comportamiento dentro de una secuencia temporal (fig. 2.11), se hace necesario, como en el resto de áreas descritas, una medida de distancia y una técnica que basada en la misma permita acelerar la búsqueda. Una de las medidas respecto a la que parece haber consenso en cuanto a su efectividad es la medida *Dynamic Time Warping (DTW)* propuesta por Sakoe y Chiba en 1978 (Sakoe and Chiba, 1978).

Como ejemplos típicos de series temporales nos podemos referir a precios de stock, cambio actual de moneda, el volumen de ventas de un producto, medidas biomédicas, datos meteorológicos, tasa de desempleo, etc, recogidos en el tiempo en base a incrementos de tiempo constantes (Gunopulos and Das, 2000; Negi and Bansal, 2005; Lin et al., 2009). Teniendo en cuenta estas series, a continuación se enuncian algunos ejemplos de *búsquedas por similitud*:

- identificar compañías con un patrón de crecimiento similar;
- determinar qué productos tienen un patrón de ventas similar;
- encontrar stocks con movimientos similares de precios;
- averiguar si un tema musical es similar a otro.

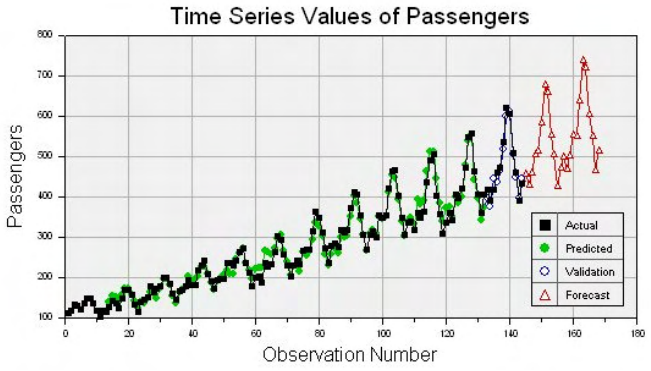


Figura 2.11: Serie temporal donde se observa el número real de pasajeros de una línea aérea en distintos momentos en el tiempo, el número estimado y la previsión para el futuro hecha en base a la similitud entre series temporales (imagen extraída de `http://dtreg.com`).

Reconocimiento de formas

En Reconocimiento de Formas se estudia la construcción de sistemas automáticos que intentan emular los aspectos perceptivos de los seres humanos (Duda et al., 2000; Theodoridis and Koutroumbas, 2009; Paulus and Hornegger, 1998). No se sabe muy bien cómo se desarrolla en las personas el proceso de reconocimiento de "entes" (ya que no son sólo objetos, puede tratarse, por ejemplo, de música, olor, etc.) partiendo de las señales que le llegan por sus sentidos, y no es fácil conseguir ese mismo proceso a través de un ordenador. Al igual que las personas reciben la información por sus sentidos, en términos informáticos se dispone de sensores que permiten captar parte de esta información (figura 2.12). La información obtenida se somete a unos procesos de segmentación y extracción de características en los que cada "objeto" queda representado por una colección de descriptores. Uno de los aspectos importantes del Reconocimiento de Formas es la clasificación donde dada una muestra de entrada la salida será la etiqueta de la clase a la que pertenece dicha muestra, escogida esta etiqueta entre un conjunto de etiquetas previamente definido.

El ámbito de aplicación del área de Reconocimiento de Formas es muy amplio [4]. Algunos ejemplos son:

- reconocimiento de caracteres escritos a mano o a máquina: es una de las utilidades más populares de los sistemas de reconocimiento de patrones

[4] http://es.wikipedia.org/wiki/Reconocimiento_de_patrones

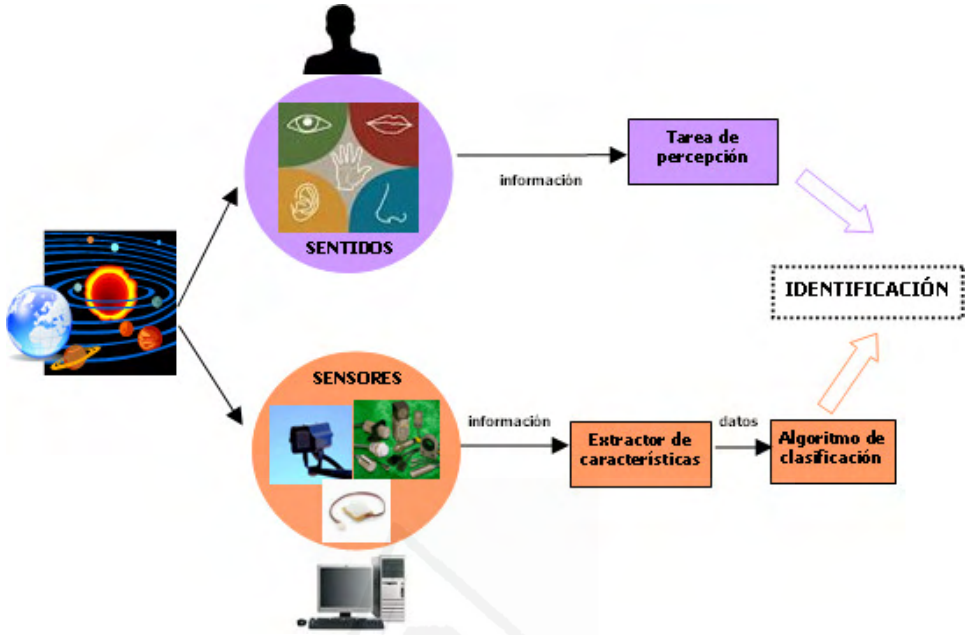


Figura 2.12: Para el reconocimiento automático de formas es necesario realizar un proceso de extracción de características de la información proporcionada por diversos sensores y, posteriormente, utilizar un algoritmo que intente simular el proceso de percepción de una persona.

ya que los símbolos de escritura son fácilmente identificables (figura 2.13);

- reconocimiento de voz: el análisis de la señal de voz se utiliza actualmente en muchas aplicaciones, un ejemplo claro son los teleoperadores informáticos;

- aplicaciones en medicina: análisis de biorritmos, detección de irregularidades en imágenes de rayos-x que permiten identificar enfermedades, detección de células infectadas, marcas en la piel, etc.

- biometría: reconocimientos tanto de huellas dactilares como de caras o del habla;

- teledetección: interpretación de fotografías aéreas y de satélite con gran utilidad para propuestas militares o civiles, como la agricultura, la geología, la geografía o la planificación urbana;

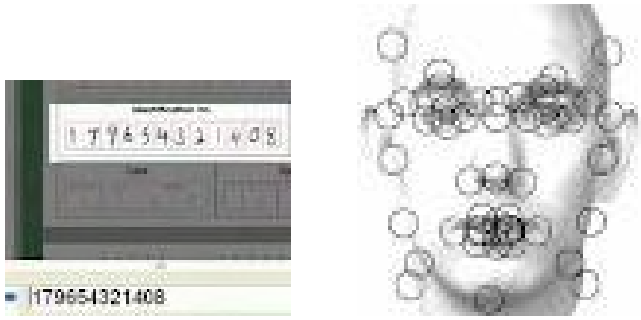


Figura 2.13: Ejemplo de reconocimientos de caracteres manuscritos (imagen extraída de http://www.scantrom.com) y de reconocimiento de caras (imagen extraída de http://www.ebanking.cl).

- reconocimiento de objetos con aplicaciones para personas con discapacidad visual;
- reconocimiento de música: identificar el estilo de música o la canción concreta que suena.

En Reconocimiento de Formas existen principalmente dos aproximaciones: la geométrica o estadística basada en la teoría estadística de la decisión (Webb, 1999), y la sintáctica o estructual basada en la teoría de lenguajes formales (Fu, 1982).

2.4. Métodos de acceso para la búsqueda por similitud

La búsqueda de información en grandes bases de datos no es una tarea fácil y, con el objetivo de intentar realizar esta tarea de un modo eficiente, se han desarrollado gran número de estructuras de almacenamiento y métodos de acceso (Chávez et al., 2001; Hjaltason and Samet, 2003; Gaede and Gunther, 1998; Böhm et al., 2001).

Según (Amato, 2002) podemos considerar tres categorías:

- **Acceso secuencial**.

  La técnica básica para procesar requerimientos de búsqueda por similitud es realizar una exploración secuencial de la base de datos completa. Para ejecutarlo eficientemente, normalmente los datos se almacenan en bloques contiguos de almacenamiento secundario. Las técnicas de exploración acceden secuencialmente a toda la base de datos conforme a

la ordenación utilizada para situar los datos en almacenamiento secundario. Esta técnica es más rápida que acceder aleatoriamente a bloques pequeños almacenados en memoria secundaria.

Los algoritmos de exploración secuencial transfieren a memoria principal grandes bloques de datos, dependiendo de la cantidad disponible de memoria principal. Después de que un bloque se transfiere, los datos contenidos en él se procesan, y un nuevo bloque es transferido hasta que todos los datos son procesados.

La ejecución de técnicas de exploración secuencial es linealmente proporcional al tamaño de la base de datos.

- **Acceso directo o de dispersión (hashing)**.

  Según Knuth (Knuth, 1987), fue un empleado de IBM el que en el año 1953 utilizó en un informe este término por primera vez. Diez años más tarde el término se comenzó a emplear de forma extendida.

  La dispersión es una técnica que utiliza una *función hash* para realizar inserciones, borrados y búsquedas en un tiempo promedio constante. Una buena función hash es aquella que experimenta pocas colisiones en el conjunto esperado de entrada, identificándolas unívocamente.

  Las técnicas de *dispersión* para la *búsqueda por coincidencia exacta* usan funciones *hash* arbitrarias: cuando el requerimiento se procesa, la función *hash* se aplica a la muestra de entrada. Entonces se busca, en el bloque que se ha obtenido al aplicar la función, la muestra almacenada que coincide exactamente con la de la entrada.

  Las técnicas de *dispersión* para la *búsqueda por similitud* usan funciones *hash* que mantienen la proximidad de las muestras de forma que las muestras cercanas son almacenadas en el mismo bloque.

  Existen muchos métodos de dispersión para la indexación aplicados con éxito en la búsqueda por similitud como *Locality Sensitive Hashing* (Gionis et al., 1999) o *Distance-Based Hashing* (Athitsos et al., 2008).

- **Métodos de acceso basados en estructuras de árbol**.

  Se basan principalmente en la descomposición jerárquica del espacio de representación. Es en este grupo de métodos donde se desarrollan las técnicas propuestas en esta tesis, por lo que a continuación se dará una visión general de estos métodos y en capítulos posteriores se formalizarán en concreto los métodos que se han propuesto.

  Todos los métodos de acceso basados en estructuras de árbol tienen caracterísicas comunes en cuanto a su estructura, todos ellos organizan

el espacio de los objetos usando regiones compactas. Sin embargo, difieren en la forma en que se definen estas regiones y el modo en que los árboles son construídos y almacenados.

Los métodos de acceso basados en estructuras de árbol dividen el conjunto de los objetos y almacenan los subconjuntos de objetos resultantes de una partición en distintos *nodos* (lo que en los métodos anteriores denominábamos *bloques*). Sin embargo, mientras que en los métodos de *dispersión* se accede a un nodo *de forma directa* como resultado de aplicar una función *hash*, en estos métodos se accede a los nodos recorriendo la estructura jerárquica de árbol.

La estructura de árbol se compone de nodos que contienen referencias a otros nodos. Uno de los nodos se distingue como raíz y existe una relación de *paternidad* que impone una estructura jerárquica sobre los nodos. Un árbol se puede definir de manera recursiva como sigue (Aho et al., 1988):

- un solo nodo es, por sí mismo, un árbol. Este nodo es también la **raíz** de dicho árbol;
- si n es un nodo y $A_1, A_2, \ldots, A_k$ son árboles con raíces $n_1, n_2, \ldots, n_k$ respectivamente, se puede construir un nuevo árbol haciendo que n sea el padre de los nodos $n_1, n_2, \ldots, n_k$. En dicho árbol, n es la raíz y $A_1, A_2, \ldots, A_k$ son los subárboles de la raíz. Los nodos $n_1, n_2, \ldots, n_k$ reciben el nombre de *hijos* del nodo n (figura 2.14).

Otros términos que también se emplearán en esta tesis son:

- los nodos sin hijos se denominan *hojas*;
- *radio* del nodo es la distancia máxima entre el representante del nodo y los prototipos contenidos en el mismo;
- si $n_1, n_2, \ldots, n_s$ es una sucesión de nodos de un árbol tal que n_i es el padre de n_{i+1} para $1 \leq i \leq s$, entonces la secuencia se denomina *camino* del nodo n_1 al nodo n_s. La *longitud* de un camino es el número de nodos del camino menos 1;
- *profundidad* del árbol: longitud máxima de todos los caminos del árbol.

El nodo raíz representará el espacio completo de búsqueda, el espacio de prototipos y cada nodo del árbol estará asociado con una región que representa un subespacio dentro de ese espacio completo de búsqueda. El subconjunto de prototipos asociado a un nodo estará contenido en el

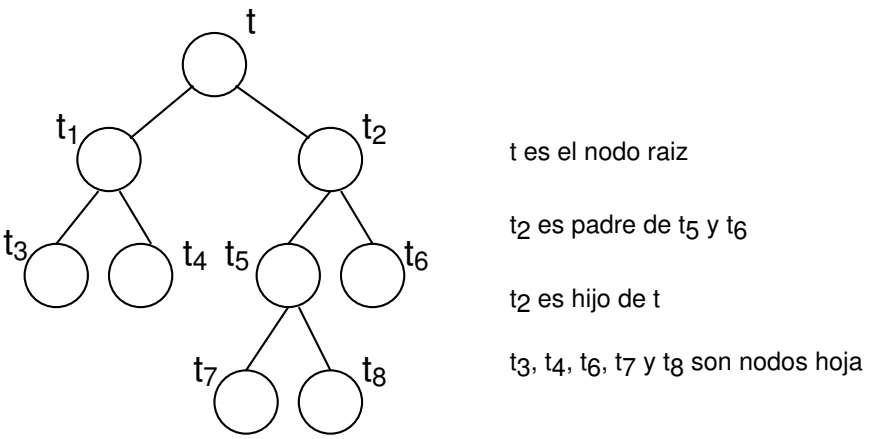


Figura 2.14: Estructura de un árbol.

subconjunto de prototipos asociado a su *nodo padre*, y contiene todos los subconjuntos de prototipos asociados a sus nodos hijo.

Las estrategias utilizadas para dividir el conjunto de prototipos y el número máximo de prototipos que puede contener un nodo, son características específicas de los distintos métodos de acceso definidos dentro de esta categoría de métodos de acceso.

Algunos métodos son diseñados para ser mantenidos en memoria principal (Brin, 1995), por lo que no suelen ser apropiados para bases de datos de gran tamaño. Otros métodos de acceso son diseñados para ser mantenidos en memoria secundaria (Ciaccia et al., 1997). En este caso, para minimizar el coste de acceso a disco, normalmente se hace coincidir el tamaño de los nodos del árbol con el tamaño de una página en disco para que, de este modo, el acceso a un nodo del árbol corresponda con un acceso a disco.

Muchos de los métodos de acceso basados en estructuras de árbol se construyen utilizando estrategias que los mantienen equilibrados en altura. Por lo tanto, las longitudes de los caminos desde el *nodo raíz* a todos los *nodos hoja* son similares.

La estructura jerárquica de estos métodos de acceso permite a los algoritmos de búsqueda por similitud ignorar subconjuntos de prototipos en los que una serie de condiciones garantizan que no se va a encontrar el resultado esperado de la búsqueda. Los algoritmos de búsqueda en este tipo de estructuras recorren el árbol desde la raíz hasta las hojas, des-

cartando en este recorrido los caminos que se corresponden con ramas del árbol donde, teniendo en cuenta la definición de los subconjuntos asociados a los nodos de estas ramas, es imposible que se localicen los prototipos que pueden dar solución a la búsqueda propuesta. Cuando en la búsqueda se alcanza un *nodo hoja*, se recorre exhaustivamente dentro de él (si hay más de un prototipo). Si aún así queda espacio por explorar, se continua la búsqueda en el resto del árbol.

En esta tesis, además de las nuevas propuestas que realizamos para construir el árbol, se estudiarán algunos árboles desarrollados por otros autores.

Capítulo 3

Algoritmos rápidos de búsqueda

Cuando se trabaja con grandes volúmenes de datos, la búsqueda por similitud requiere una atención especial a la eficiencia para poder obtener una respuesta en el menor tiempo posible. Como ya se apuntó en el capítulo anterior, en esta tesis para realizar la búsqueda se emplea la *técnica del vecino más cercano*.

La técnica trivial para realizar la búsqueda del *vecino más cercano* requiere explorar todos los prototipos de la base de datos, S. Esto significa que el coste de la búsqueda depende linealmente de la talla del conjunto S cuando se realiza una *búsqueda exhaustiva*. Por lo tanto si consideramos, como ocurre en la práctica, escenarios de trabajo con un gran número de prototipos, la *búsqueda exhaustiva* o *fuerza bruta* (algoritmo 1) resulta inapropiada. Hay que tener en cuenta que se calculan tantas distancias como prototipos hay en la base de datos, y además que el cálculo de estas distancias puede suponer un coste relativamente elevado.

Desde hace ya años se han ido proponiendo un gran número de algoritmos encaminados a reducir el coste computacional de la búsqueda de vecino más cercano, ya sea su objetivo reducir el número de distancias calculadas, el número de accesos a disco, el coste temporal del algoritmo (*overhead*), etc. Se pueden encontrar referencias y el detalle de alguno métodos en (Böhm et al., 2001; Chávez et al., 2001; Hjaltason and Samet, 2003).

No obstante, hay que considerar que las mejoras que proporcionan todos estos métodos en la búsqueda, son obtenidas gracias a un *preproceso* que conlleva a su vez un coste. El preproceso consiste en la selección de un conjunto de prototipos, o en la construcción de un árbol de búsqueda, o en el cálculo y almacenamiento de una serie de distancias, etc. Esta información procesada será utilizada para la búsqueda posterior. En consecuencia, para determinadas aplicaciones, se ha de considerar que el coste de preproceso puede ser alto

y que la elección de estos métodos como alternativa a la fuerza bruta debe considerar este coste y llegar a un compromiso entre los costes de preproceso y búsqueda (Chávez et al., 2001).

Algoritmo 1 `Búsqueda exhaustiva`(S, x)

Entrada: S (conjunto de prototipos no vacío);
x (muestra);
Salida: $nn \in S$;
$d_{nn} \in \mathcal{R}$;

```
1: d_nn = ∞;
2: nn = elemento arbitrario de S;
3: para todo p ∈ S hacer
4:    d_aux = d(x, p);
5:    si d_aux < d_nn entonces
6:       nn = p;
7:       d_nn = d_aux;
8:    fin si
9: fin para
```

Por otro lado hay que considerar lo que se llama *la maldición de la dimensionalidad*, término introducido por Bellman (Bellman and Corporation, 1957) (Bellman, 1961) que se refiere al crecimiento exponencial del volumen asociado a un espacio matemático debido al incremento de dimensiones, y que influye de manera significativa en la búsqueda del vecino más cercano (Beyer et al., 1999). Entendiendo por *dimensión* el número de características que definen los prototipos, ha quedado constatado experimentalmente que si la dimensión de los prototipos es muy alta, el resultado de aplicar estrategias alternativas no mejora la eficiencia obtenida con la *fuerza bruta*. No obstante, no se puede asumir que la dimensión de un conjunto de prototipos determine su comportamiento (Filho et al., 2001). Un conjunto de prototipos puede estar ocupando un subespacio dentro del espacio definido por su dimensión. Además, los conjuntos de prototipos en espacios métricos generales (cadenas, grafos, etc) no tienen asociada ninguna dimensión, por lo tanto para valorar de un modo más adecuado la incidencia que tiene el tamaño del espacio que ocupan los prototipos sobre la eficiencia habría que hablar de la *dimensionalidad intrínseca*. La *dimensionalidad intrínseca* de un conjunto de prototipos hace referencia al número mínimo de parámetros necesarios para la "generación" de dichos prototipos (Baydal et al., 1987), y es un factor que influye significativamente en la eficiencia de casi todos los métodos propuestos para dar solución a la búsqueda del vecino más cercano

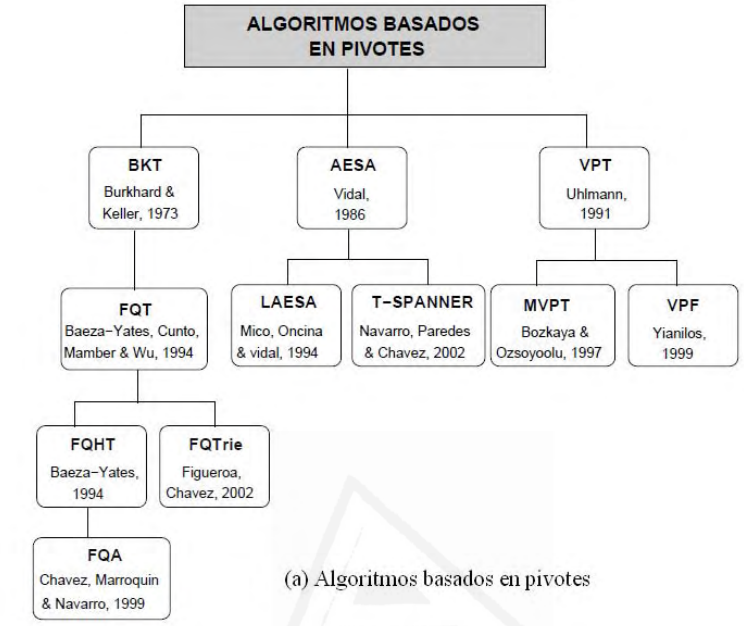
Exact NN	*Based on Space Partitioning*	
	1. KDTrees	6. X-Trees
	2. Metrics Trees	7. M-Trees
	3. vp-Trees	8. SR-Trees
	4. Voronoi Diagrams	9. TV-Trees
	5. R-Trees	10. VA-Files
	Based on Scalar Projection	
	1. Orchard's Method	4. LAESA
	2. Annulus Method	5. LSH
	3. AESA	
	Based on Intrinsic Dimensionality	
	1. Cover Trees	
Approx NN	*Based on Space Partitioning*	
	1. BBF-Trees (KDTrees with Priority Queue)	
	2. BBD-Trees	
	3. Hybrid Sp-Trees	
	Based on Scalar Projection	
	1. LSH	
	Based on Intrinsic Dimensionality	
	1. Navigating Nets	

Figura 3.1: Clasificación de métodos de búsqueda del vecino más cercano propuesta por Ashraf Masood Kibriya en su tesis (Kibriya, 2007).

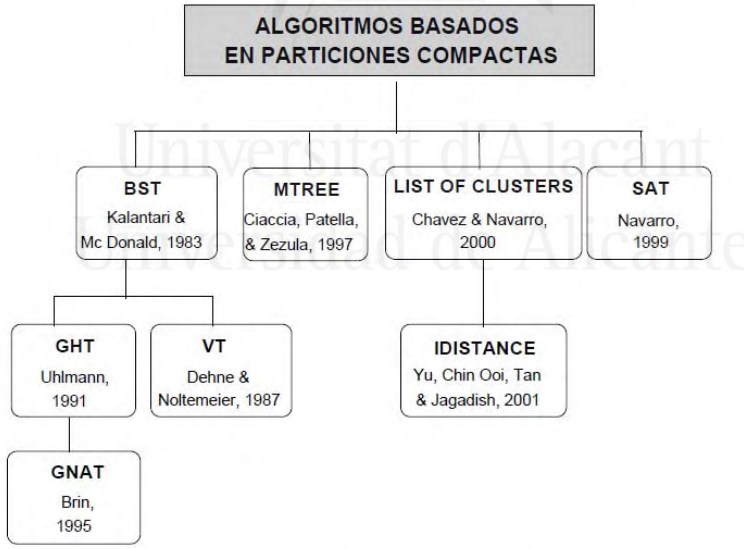
(Houle et al., 2010; Camastra, 2003; Pettis et al., 1979).

3.1. Estado de la cuestión en búsqueda rápida del vecino más cercano

Tal y como ya se ha comentado, principalmente son dos los factores que determinan el coste computacional de la *búsqueda del vecino más cercano*: la *dimensionalidad intrínseca de los datos* y *el tamaño del conjunto de datos*. Sin embargo, tal y como algunos autores indican (Kibriya, 2007), la mayoría de los métodos propuestos se centran en el tamaño del conjunto de datos intentando reducir el conjunto de datos a examinar, mientras que hay pocas técnicas conocidas que se centran en la dimensionalidad de los mismos: *Partial Distance Search* (Bei and Gray, 1985) y *TV-Trees* (Lin et al., 1994) (que también se ocupa del tamaño del conjunto de datos).



(a) Algoritmos basados en pivotes



(b) Algoritmos basados en particiones compactas

Figura 3.2: Clasificación de métodos de búsqueda del vecino más cercano propuesta por Karina Figueroa en su tesis (Figueroa, 2007).

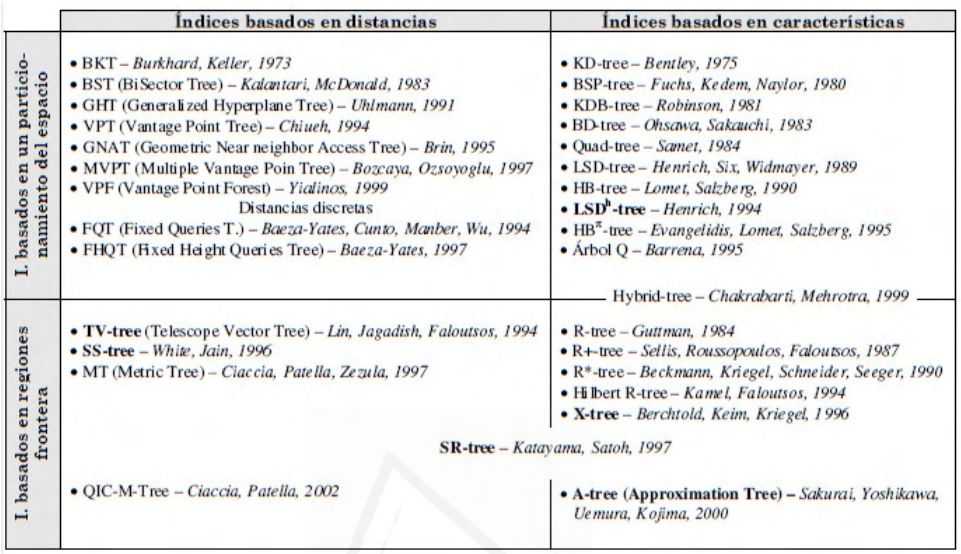


Figura 3.3: Clasificación de métodos de búsqueda del vecino más cercano propuesta por Elena Jurado en su tesis (Jurado, 2003).

Son muchos los métodos propuestos para reducir el coste asociado a encontrar los *k-vecinos más cercanos* en general, y se podrían clasificar atendiendo a diversos criterios.

Un criterio de clasificación puede venir dado por el hecho de que sean métodos de búsqueda *exacta* o *aproximada*, tal y como empieza clasificándolos Ashraf Masood en su tesis (Kibriya, 2007), y podemos ver en la figura 3.1. Otro criterio, que se puede combinar con el anterior, podría ser si están basados en *pivotes* [1] o en *particiones compactas*, tal y como se muestra en la figura 3.2 extraída de la tesis de Karina Figueroa (Figueroa, 2007). Otros autores distinguen entre métodos aplicables en cualquier *espacio métrico* o sólo en *espacios vectoriales* (Moënne-Loccoz, 2005). Basándose a la vez en varios criterios de los que se han comentado anteriormente, Elena Jurado propone en su tesis (Jurado, 2003) la clasificación que se muestra en la figura 3.3.

Cuando los métodos son válidos sólo para espacios vectoriales se suele utilizar la idea de particionar el espacio en subconjuntos, agrupando en el mismo subconjunto aquellos prototipos que tienen alguna propiedad en común, por

[1] *pivote* se define normalmente como un elemento representativo del conjunto para construir el índice

ejemplo, alguna condición referida al valor de una coordenada, como ocurre con los *kd-trees* (Bentley, 1975) o los *quad-tress* (Finkel and Bentley, 1974).

En el caso de los espacios métricos generales también se pueden utilizar técnicas de particionamiento aunque, en este caso, el particionamiento de estos espacios es más complicado al no existir coordenadas en las que apoyarse para realizar la división de los prototipos en subconjuntos. En el caso de los espacios métricos generales, los métodos propuestos hacen uso de los valores de la distancia entre prototipos, valores que, bien se almacenan, o bien sirven para estructurar o particionar el espacio (*índices métricos o basados en distancias*). Estos métodos hacen uso posteriormente de algunas propiedades de la distancia, fundamentalmente la *desigualdad triangular* (Heath and Heath, 1956), para dejar de considerar en la búsqueda algunos prototipos o incluso regiones enteras del espacio.

Son varios los autores que al hablar del estado de la cuestión en búsquedas por similitud hacen referencia a (Zezula et al., 2006), por lo que se comenta a continuación la visión que se da en esta obra. Se consideran los siguientes tipos de métodos:

- métodos que se basan en una *partición del espacio en hiperesferas.*

  Entre estos métodos podemos destacar Burkhard-Keller Tree (*bk-tree*) (Burkhand and Keller, 1973) o Vantage Point Tree (*vp-tree*) (Yianilos, 1993). Estos métodos dividen el espacio de los prototipos S en dos subconjuntos S_1 y S_2 realizando un "corte esférico" obtenido en base a un prototipo c y una distancia d, de manera que todos los prototipos p de S se distribuyen en dos subconjuntos, S_1 o S_2, de acuerdo a las siguientes reglas:

  - $S_1 = \{p \in S | d(p, c) \leq d\}$
  - $S_2 = \{p \in S | d(p, c) > d\}$.

- Métodos que se basan en *particiones del espacio mediante hiperplanos*, como *Bisector Tree* (Kalantari and McDonald, 1983), una mejora de éste, *Voronoi Tree* (Dehne and Noltemeier, 1987), o el método *Generalized Hyperplane Tree* (Uhlmann, 1991). Estos métodos también particionan un conjunto de prototipos S en dos subconjuntos S_1 y S_2, pero esta vez lo hacen en base a dos prototipos, p_1 y p_2 del conjunto S. Los prototipos de S se asignan a S_1 o S_2 dependiendo de las distancias a estos 2 prototipos:

  - $S_1 = \{p \in S | d(p, p_1) \leq d(p, p_2)\}$
  - $S_2 = \{p \in S | d(p, p_1) > d(p, p_2)\}$.

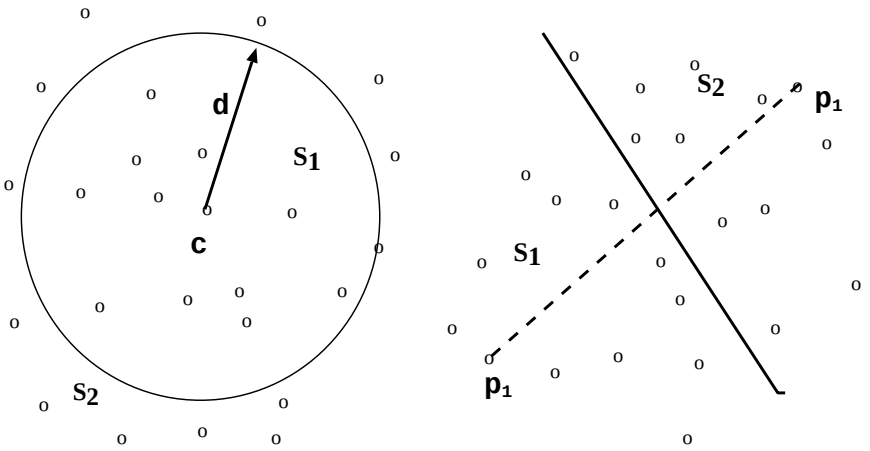


Figura 3.4: Partición del espacio de los prototipos mediante hiperesferas (izquierda) e hiperplanos (derecha)

En la figura 3.4 podemos ver un ejemplo de una partición basada en hiperesferas y otra mediante hiperplanos con un reparto equilibrado de los prototipos.

- Métodos *basados en distancias precalculadas*, tales como Approximating and Eliminating Search Algorithm (*AESA*) (Vidal, 1986) (Vidal, 1994) y su variante lineal (Micó et al., 1992) (Micó et al., 1994). Estos métodos almacenan distancias entre prototipos del espacio de búsqueda, bien entre todos o entre prototipos seleccionados de este espacio y, posteriormente, usan estas distancias para poder estimar otras, lo que permite ahorrar en el cálculo de distancias.

- Métodos *híbridos* que combinan la partición de espacio con el uso de distancias precalculadas, como *Multi Vantage Point Tree* (Bozkaya and Özsoyoglu, 1997) (Bozkaya and Özsoyoglu, 1999), los que se basan en diagramas de Voronoi como *Geometric Near-neighbor Access Tree* (*GNAT*) (Brin, 1995) y *Spatial Approximation Tree* (*SAT*) (Navarro, 1999) (Navarro, 2002), o la estructura dinámica *M-Tree* (Ciaccia et al., 1997).

A continuación, y por orden cronológico de aparición, se revisan con mayor profundidad algunos de estos métodos, en particular los que posteriormente se utilizarán para establecer comparaciones con los resultados obtenidos en los nuevos métodos propuestos en esta tesis.

3.1.1. k-dimensional tree (kd-tree)

Entre los árboles clásicos para la búsqueda del vecino más cercano se encuentra el *kd-tree*, un árbol donde se organizan y almacenan los prototipos de un espacio de k dimensiones (*k-dimensional tree*) y que fue propuesto por Bentley en 1975 (Bentley, 1975). Es un árbol binario en el que, en cada nivel del árbol, se han distribuido los prototipos en dos subárboles La división se realiza recursivamente en cada subárbol. mínimo establecido, que es usado para su representación en el árbol.

En la raíz del árbol la división se hace en base a un *valor de corte* de la primera coordenada, en el siguiente nivel se utilizará la siguiente coordenada, y así sucesivamente hasta utilizar todas las coordenadas. Una vez utilizadas todas las coordenadas de nuevo se volverá a utilizar la primera, tal y como Bentley definía originalmente en su artículo al indicar que cada nivel del árbol tenía asociada una *coordenada discriminante* (para el nivel i la coordenada discriminante sería $(i+1) \mod k$). [2]

Durante la fase de búsqueda se recorre el árbol siguiendo un esquema de ramificación y poda para encontrar el vecino más cercano a una muestra dada x. El proceso de búsqueda en el *kd-tree* es recursivo. Dada una muestra x y un nodo cualquiera del árbol (que no sea una hoja), se compara la coordenada de x que es discriminante para ese nodo con el valor de corte que sirvió para construir sus subárboles, y se continua buscando por el subárbol correspondiente. Cuando se llega a una hoja se calcula la distancia y se continúa explorando el árbol mientras queden ramas por visitar.

Como se ha indicado, tanto en la fase de construcción del árbol como en la de búsqueda, se utilizan las coordenadas de los prototipos, por lo que este árbol sólo puede representar espacios vectoriales. Sobre este algoritmo se han realizado muchas optimizaciones (Bentley, 1990; Duch et al., 1998; Samet, 2006; Silpa-Anan and Hartley, 2008). Sin embargo, sigue siendo su versión original un algoritmo de referencia cuando se utilizan distancias euclídeas. No obstante, este algoritmo presenta un problema importante, *degenera con la dimensionalidad*, es decir, cuando la dimensionalidad de los datos es alta, el algoritmo de búsqueda se comporta prácticamente igual que el algoritmo exhaustivo (visita casi todos los nodos del árbol).

Construcción del kd-tree

Para construir un *kd-tree* a partir de un conjunto de prototipos S la cuestión principal es elegir un hiperplano que divida el conjunto S en dos

[2] En *http://donar.umiacs.umd.edu/quadtree/points/kdtree.html* hay disponible una herramienta para visualizar gráficamente la construcción del árbol.

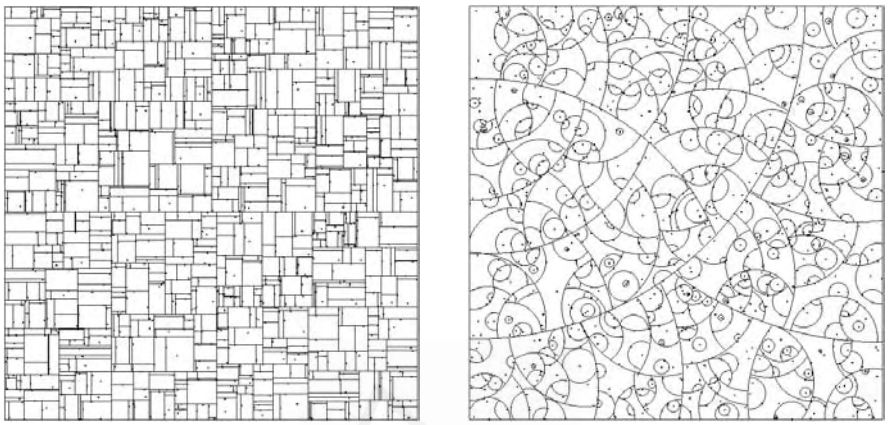


Figura 3.5: Ejemplo mostrado en (Yianilos, 1993) para la descomposición de un espacio de dimensión 2 utilizando el algoritmo *kd-tree* y utilizando el *vp-tree*.

subconjuntos y proceder de manera recursiva con los subconjuntos. Otro aspecto a resolver es la elección de la coordenada (*coordenada discriminante*) que guía la construcción.

Para intentar que el árbol resulte lo más equilibrado posible, se suele seleccionar el hiperplano de manera que se sitúe en la mediana de los valores de la coordenada discriminante. Además, también se suele elegir como coordenada discriminante aquella que tenga la máxima varianza.

El árbol se construye del siguiente modo:

- en cada nodo, comenzando por el nodo raíz que contiene el conjunto completo de prototipos, se elige la coordenada discriminante y se almacena como información del nodo;

- se obtiene la mediana de los valores de dicha coordenada (v) para los prototipos del nodo (también se almacena como información del nodo);

- se divide el conjunto de prototipos del nodo en dos subconjuntos que dan lugar a sus dos *nodos hijo*, situando en un subconjunto los prototipos para los que el valor de la coordenada discriminante es menor o igual al valor de la mediana y, en el otro subconjunto, los prototipos cuya coordenada discriminante es mayor;

- se crean recursivamente los árboles asociados a cada subconjunto, terminando el proceso cuando el tamaño del conjunto de prototipos que constituyen el nodo sea menor o igual a un tamaño previamente fijado, caso en el que nos encontraríamos en un *nodo hoja*.

La parte izquierda de la figura 3.5 muestra la división de un espacio de dimensión 2 utilizando el *kd-tree*.

Búsqueda en el kd-tree

Dada una muestra $x = (x_1 \dots x_d)$, la distancia al vecino más cercano hasta el momento d_{nn} (inicialmente $d_{nn} = \infty$) y un nodo del árbol, para buscar el vecino más cercano a x, se procede recursivamente del siguiente modo (Friedman et al., 1975):

- Si el nodo no es un *nodo hoja*
  - se compara el valor en x de la coordenada discriminante c para ese nodo con el valor de corte v;
  - si $x_c + d_{nn} \leq v$, el hijo derecho de ese nodo no puede contener al vecino más cercano y, por tanto, no es necesario realizar la búsqueda por el hijo de la derecha (de igual modo, si $x_c - d_{nn} \geq v$, no es necesario buscar en el hijo izquierdo). En caso contrario se procede a realizar la búsqueda en el hijo derecho (de igual modo en el izquierdo);
- si el nodo es un *nodo hoja*, se compara la muestra x con todos sus prototipos (es decir, se calcula la distancia de la muestra x a todos los prototipos).

3.1.2. Algoritmo de Fukunaga y Narendra (FN)

Este algoritmo (Fukunaga and Narendra, 1975) fue de los primeros que se desarrolló para espacios métricos generales y se basa en la construcción de un árbol a partir de un conjunto inicial de prototipos. Este conjunto constituye el *nodo raíz* del árbol, y se divide en l subconjuntos, *nodos*, que a su vez se dividen de nuevo, y así sucesivamente hasta llegar a un cierto nivel de profundidad del árbol en el que los subconjuntos de prototipos constituyen las *hojas* del árbol.

Cada nodo del árbol está representado por $S_{i,j}$ (donde i se refiere a la profundidad del árbol y j al número de nodo). Además, si un nodo no es una hoja:

- tiene l hijos;
- representa a un subconjunto de prototipos $S_{i,j}$;
- tiene un representante $c_{i,j}$;
- y tiene un radio $r_{i,j}$ que es la distancia máxima del representante del nodo, $c_{i,j}$, a los prototipos del nodo $S_{i,j}$.

Para obtener los distintos subconjuntos en que se dividen los nodos, los autores propusieron la utilización de un algoritmo de agrupamiento como el *k-medias*. A raíz de esta sugerencia se podría pensar que el algoritmo es sólo válido para espacios vectoriales, sin embargo, ya que el algoritmo no hace uso de las coordenadas, el algoritmo de *Fukunaga y Narendra* es aplicable a espacios métricos utilizando un algoritmo de agrupamiento válido para estos espacios.[3]

Construcción del árbol

Dado un conjunto de prototipos S y una función distancia d, para construir un árbol de Fukunaga y Narendra con l hijos por nodo y una profundidad n, el proceso sería el siguiente:

1. todos los prototipos están en el nodo raíz y se inicializa $i = 0$, $j = 0$, es decir, $S_{0,0} = S$;
2. dado un nodo asociado a $S_{i,j}$:
   - se seleccionan l prototipos del nodo como resultado de la aplicación de un algoritmo de agrupamiento seleccionado (*k-medias*) y se reparten los prototipos del nodo en l subconjuntos según el representante al que su distancia es menor;
   - se incrementa en 1 el nivel, $i = i + 1$;
   - se crean l nodos a los que se les asigna como representantes los l prototipos seleccionados y como conjunto de prototipos los subconjuntos asociados a cada uno de estos representantes, a la vez que se va actualizando el número de nodo, $S_{i,j+1}$, $S_{i,j+2}$, ..., $S_{i,j+l}$;
   - si $i < n$, para los nodos obtenidos que contengan más de l prototipos se repite el mismo proceso desde el paso 2.

[3]En concreto, en la implementación que se ha empleado en esta tesis se utiliza el algoritmo de las $k-medianas$, en el que en vez de utilizar la media se obtiene la mediana, que además se usa como representante del nodo.

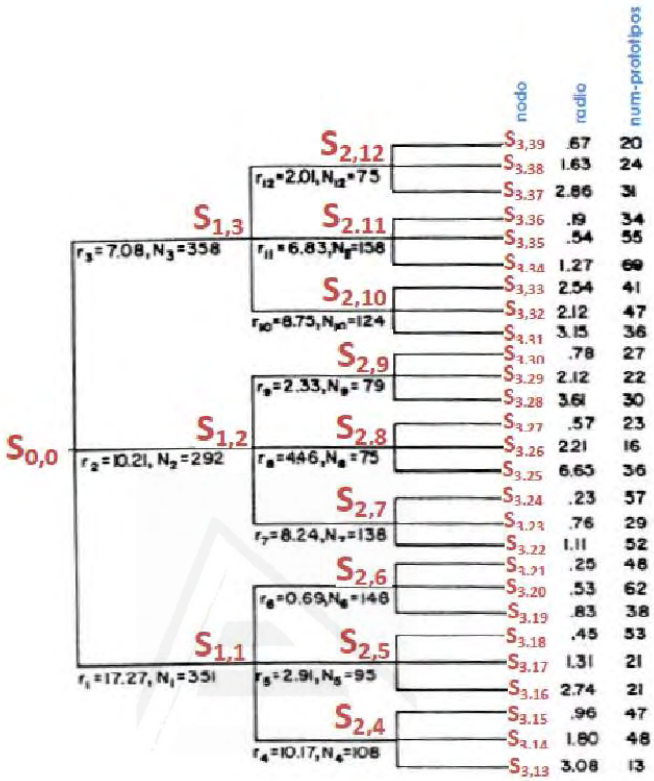


Figura 3.6: Construcción de un árbol con 3 hijos por nodo siguiendo la propuesta de Fukunaga y Narendra, para un conjunto de 1000 prototipos de una distribución gaussiana.

En la figura 3.6 se puede ver la construcción del árbol correspondiente al experimento realizado por Fukunaga y Narendra con 1000 prototipos pertenecientes a una distribución gaussiana.

Búsqueda en el árbol de Fukunaga y Narendra

Dada una muestra x para la que se quiere encontrar el vecino más cercano, el algoritmo propuesto para la búsqueda utiliza dos reglas de eliminación, una para los nodos hoja y otra para los que no lo son. Estas reglas se definen en base a la información que se almacena en cada nodo.

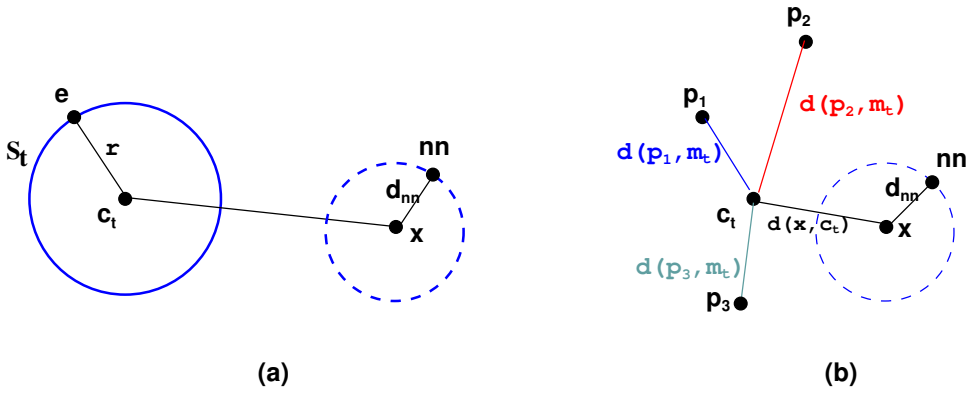


Figura 3.7: Reglas de eliminación propuestas por Fukunaga y Narendra para un nodo que no es hoja (a) y para un nodo hoja con 3 prototipos (b)

- Un nodo t que no es hoja puede ser eliminado si se cumple que

$$d_{nn} + r < d(x, c_t)$$

, donde d_{nn} es la distancia de la muestra, x, al prototipo más cercano hasta el momento y c_t es el representante del nodo (fig. 3.7(a)).

- Un nodo hoja t puede ser eliminado si se pueden eliminar todos sus prototipos. Es decir, se debe ir aplicando la misma regla a cada uno de los prototipos. La regla dice que un prototipo puede ser eliminado si se cumple que

$$d_{nn} + d(p_i, c_t) < d(x, c_t)$$

, donde p_i representa a cada prototipo del nodo hoja t (fig. 3.7(b)).

En el artículo original de Fukunaga y Narendra se explica el algoritmo de búsqueda utilizando una lista por cada nivel del árbol para realizar su recorrido, sin embargo, resulta mucho más sencilla la formulación recursiva (Moreno-Seco, 2004) que realiza la búsqueda con los siguientes pasos:

1. dado un nodo t del árbol, que no es hoja, se calcula la distancia de la muestra x al representante de cada hijo de t, y se actualiza el vecino más cercano hasta el momento (inicialmente $d_{nn} = \infty$);

2. de todos los nodos hijos de t que no se han eliminado ni visitado anteriormente, se toma aquel nodo, t_j, cuya distancia a la muestra sea la menor;

3. dado un nodo t_j:
   - si t_j no es una hoja se comprueba si se puede eliminar aplicando la regla para los nodos que no son hoja enunciada anteriormente, figura 3.7(a). Si no se puede eliminar se debe buscar recursivamente en t_j;
   - si t_j es una hoja, se eliminan aquellos prototipos p_i contenidos en t_j que no pueden estar más cercanos a la muestra que el vecino más cercano hasta el momento, es decir, aquellos que cumplen la regla de eliminación de los nodos hoja, figura 3.7(b). Para los prototipos de t_j no eliminados se debe calcular su distancia a la muestra y actualizar el vecino más cercano;
4. se repiten los pasos 2 y 3 hasta que no queden hijos de t sin eliminar o visitar.

3.1.3. Vantage Point tree (*vp-tree*)

El *vp-tree* es un árbol binario completamente equilibrado que se construye realizando cortes esféricos en el conjunto de prototipos para distribuirlos en las ramas del árbol. El *vp-tree* fue propuesto por Yianilos en 1993 (Yianilos, 1993), junto con algunas mejoras del mismo *vp^s-tree* y *vp^{s}b-tree*, para la búsqueda del vecino más cercano en espacios métricos generales. Fue diseñado para funciones distancia continuas aunque también se puede utilizar con funciones discretas sin casi modificaciones.

Como en otros árboles, cada nodo del árbol representa un subconjunto de prototipos del conjunto global, y se utiliza un elemento destacado del conjunto, llamado pivote (*vantage point*), para dividir el conjunto de prototipos de un nodo en dos subconjuntos, uno por cada nodo hijo.

Construcción del árbol

Llamemos t a un nodo del árbol, S_t al conjunto de prototipos de ese nodo, hi_t al nodo hijo de la izquierda de t y hd_t al nodo hijo de la derecha de t.

Inicialmente todos los prototipos se encuentran en el nodo raíz y a partir de él se construye el *vp-tree* básico con los pasos siguientes:

- se selecciona uno de los prototipos del nodo al que se llama pivote p

$$c_t = p;$$

- recursivamente se repite el siguiente proceso hasta que en cada nodo sólo queda un prototipo, los que constituyen los *nodos hoja*:

- se calcula la *distancia de partición* del nodo, d_p, que será la mediana de las distancias entre el prototipo p y el resto de prototipos del nodo [4];
- se crea un nodo hijo de la izquierda, hi_t, al que se asignan los prototipos cuya distancia a p es menor o igual que la distancia de partición d_p y se almacenan la distancia máxima y mínima de p a los prototipos del nodo hi_t:

$$S_{hi_t} = \{s | s \in S_t - \{p\} \wedge d(s,p) < d_p\},$$
$$d_{\max}(hi_t) = \max_{s \in S_{hi_t}} d(s,p),$$
$$d_{\min}(hi_t) = \min_{s \in S_{hi_t}} d(s,p);$$

- se crea un nodo hijo de la derecha, hd_t, al que se asignan los prototipos cuya distancia a p es mayor que la distancia de partición d_p y se almacenan la distancia máxima y mínima de p a los prototipos del nodo hd_t:

$$S_{hd_t} = \{s | s \in S_t - \{p\} \wedge d(s,p) \geq d_p\},$$
$$d_{\max}(hd_t) = \max_{s \in S_{hd_t}} d(s,p),$$
$$d_{\min}(hd_t) = \min_{s \in S_{hd_t}} d(s,p).$$

La elección de la mediana como distancia de partición hace que sea posible que los prototipos de un nodo se distribuyan de forma equitativa entre sus dos nodos hijos, dando lugar a un árbol binario equilibrado. La profundidad del árbol sería en este caso $O(\log_2(n))$ donde n es el número de prototipos en el nodo raíz. La construcción del árbol requiere el cálculo de $O(n \log_2(n))$ distancias.

En la versión original del *vp-tree* el prototipo pivote p se elige de forma aleatoria. En (Yianilos, 1993) se propone una mejora que consiste en elegir el pivote basándose en una sencilla característica estadística: se selecciona como pivote el prototipo que maximiza la varianza de las distancias entre prototipos del nodo, ya que la elección de este pivote permite podar más nodos durante la fase de búsqueda.

En cada nodo sólo se almacenan, además de los prototipos pertenecientes al nodo, su pivote p y la distancia de partición d_p.

La parte derecha de la figura 3.5 muestra la partición de un espacio de dimensión 2 utilizando el *vp-tree*.

[4] En una propuesta posterior se propone como distancia de partición el uso de la distancia media en lugar de la mediana (Chávez et al., 2001)

Búsqueda en el árbol

La búsqueda del vecino más cercano a una muestra dada x es un proceso recursivo que se inicia en el nodo raíz, suponiendo que inicialmente la distancia al vecino más cercano es infinita, $d_{nn} = \infty$.

La búsqueda en un nodo concreto t consiste en los siguientes pasos:

1. se calcula la distancia de la muestra al pivote del nodo y se actualiza el vecino más cercano hasta el momento
   - $d = d(x, c_t)$,
   - si $d < d_{nn}$ entonces
     $$d_{nn} = d,$$
     $$nn = c_t;$$
2. **si** t no es un nodo hoja **entonces**
   - $c = \frac{d_{\max}(hi_t)+d_{\min}(hd_t)}{2}$;
   - **si** $d < c$ **entonces**
     si $d_{\min}(hi_t) - d_{nn} < d < d_{\max}(hi_t) + d_{nn}$ **entonces**
     se inicia el paso 1 para el nodo hi_t*;*
     si $d_{\min}(hd_t) - d_{nn} < d < d_{\max}(hd_t) + d_{nn}$ **entonces**
     se inicia el paso 1 para el nodo hd_t;
   - **si no entonces**
     si $d_{\min}(hd_t) - d_{nn} < d < d_{\max}(hd_t) + d_{nn}$ **entonces**
     se inicia el paso 1 para el nodo hd_t;
     si $d_{\min}(hi_t) - d_{nn} < d < d_{\max}(hi_t) + d_{nn}$ **entonces**
     se inicia el paso 1 para el nodo hi_t;

Posteriormente, Bozkaya y Ozsoyoglu (Bozkaya and Özsoyoglu, 1999) propusieron el *mvp-tree* que, como el *vp-tree*, divide el espacio de los prototipos en cortes esféricos alrededor de los pivotes. A diferencia del *vp-tree*, el *mvp-tree* crea particiones respecto a más de un pivote en cada nivel y mantiene información adicional en los nodos hoja para utilizarla al realizar posteriormente la búsqueda por el árbol, lo que mejora los resultados obtenidos con el *vp-tree* (Bozkaya and Özsoyoglu, 1997).

3.1.4. La familia *AESA*

El *AESA* (Approximating Eliminating Search Algorithm) es un algoritmo propuesto por Vidal en 1985 (Vidal, 1986)(Vidal, 1994) que utiliza como

base una matriz donde almacena las distancias entre todos los prototipos del espacio de trabajo. Esta matriz se utiliza durante la búsqueda del vecino más cercano para ir descartando candidatos, empleando para ello las distancias almacenadas y la propiedad de la desigualdad triangular.

Este algoritmo, tal y como su nombre indica, consta de dos fases: *aproximación* y *eliminación*. Dado un conjunto de prototipos S entre los que se quiere realizar la búsqueda, para cada uno de los prototipos de este conjunto se obtiene una cota inferior de la distancia real entre el prototipo y la muestra x. Cuando dicha cota es mayor que la distancia del vecino más cercano hasta el momento, el prototipo ya no puede ser el vecino más cercano y por tanto se puede eliminar (eliminación). El siguiente candidato a vecino más cercano (aproximación) será áquel cuya cota sea la menor de entre los prototipos que quedan tras la eliminación.

Búsqueda con el *AESA*

Dado un conjunto de prototipos S y una función distancia d, para calcular el vecino más cercano utilizando este algoritmo necesitamos los siguientes nuevos elementos:

- M, la matriz precalculada de las distancias entre todos los prototipos de S;
- una función cota inferior de la distancia, g, definida del siguiente modo:

$$g(p) = \max_{s \in S} |d(x,s) - d(s,p)| \quad \forall s \in S^5$$

con $g(p) = 0$ si $S = \emptyset$.

AESA es uno de los algoritmos que calcula menor número de distancias para encontrar el vecino más cercano, sin embargo, al necesitar hacer uso de una matriz donde están almacenadas las distancias entre todos los prototipos del conjunto con el que se trabaja, su coste espacial es $O(n^2)$ siendo n el número de prototipos. Esto conlleva, además de un coste espacial elevado, un tiempo de preproceso también elevado.

Dado el buen comportamiento del algoritmo de búsqueda, han sido muchas las propuestas que se han realizado para intentar mejorar tanto su tiempo de preproceso como su coste espacial, como por ejemplo el algoritmo *LAESA* (Micó et al., 1992) (Micó et al., 1994) que elige k prototipos de S como pivotes, almacenando sólo las distancias de estos prototipos al resto y reduciendo el coste espacial a $O(kn)$, o una versión mejorada del *LAESA*,

[5] La elección de esta función se detalla en (Vidal, 1986)

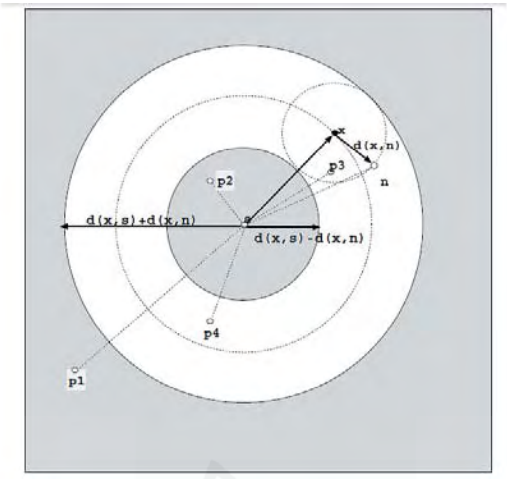


Figura 3.8: Fase de eliminación empleando *AESA*, extraída de la tesis de Luisa Micó.

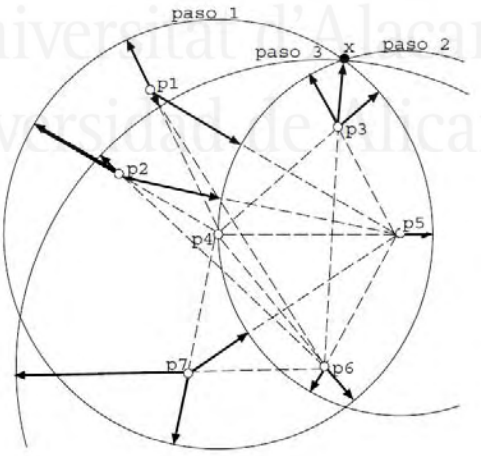


Figura 3.9: Fase de aproximación empleando *AESA*, extraída de la tesis de Luisa Micó.

Algoritmo 2 `AESA`(S, x, M)

Entrada: S (conjunto de prototipos no vacío);
x (muestra);
M (matriz con las distancias precalculadas entre prototipos);
Salida: nn; *// vecino más cercano*
$d_{nn} \in \mathcal{R}$;

1: **para todo** $p \in S$ **hacer**
2: $g(p)$=0;
3: **fin para**
4: $d_{nn} = \infty$;
5: s = elemento arbitrario de S;
6: **mientras** $S \neq \emptyset$ **hacer**
7: $d_s = d(x, s)$;
8: $S = S - \{s\}$;
9: **si** $d_s < d_{nn}$ **entonces**
10: $nn = s$; *// actualización del vecino más cercano*
11: $d_{nn} = d_s$;
12: **fin si**
13: s_1 = elemento arbitrario de S;
14: $g_{min} = \infty$;
15: **para todo** $p \in S$ **hacer**
16: $g(p) = \text{máx}(g(p), |M[p, s] - d_s|)$; *// actualización de la cota*
17: **si** $g(p) > d_{nn}$ **entonces**
18: $S = S - \{p\}$; *// eliminación*
19: **si no**
20: **si** $g(p) < g_{min}$ **entonces**
21: $g_{min} = g(p)$;
22: $s_1 = p$; *// aproximación*
23: **fin si**
24: **fin si**
25: **fin para**
26: $s = s_1$;
27: **fin mientras**
28: **devolver** nn;

TLAESA (Micó et al., 1996) que logra un coste sublineal para el número de distancias calculadas, o el algoritmo Spaghettis (Chávez et al., 1999) que propone reducir el tiempo de CPU extra, necesario al realizar una consulta, utilizando una estructura de datos en donde las distancias a los pivotes están ordenadas, lo que permite utilizar una búsqueda binaria.

3.1.5. Spatial approximation tree (*SAT*)

El algoritmo *SAT*, *Árbol de Aproximación espacial*, (Chávez et al., 2001; Navarro, 2002) parte de una idea diferente a los algoritmos comentados hasta ahora, y es acercarse espacialmente a la consulta. Para conseguirlo, para cada prototipo se define su conjunto de vecinos durante la construcción de la estructura. En la búsqueda, se parte del mismo prototipo con el que se empezó a construir esta estructura, y se va realizando una *aproximación espacial* a través de sus vecinos, de forma que poco a poco se está cada vez más cerca del objetivo de la consulta. Cuando no se pueden realizar más movimientos es porque ya se ha encontrado el prototipo más cercano.

La estructura que de un modo natural representa esta restricción es un grafo dirigido donde los nodos son los prototipos de S y de cada nodo parten arcos hacia los prototipos que constituyen su conjunto de vecinos. El grafo de Delaunay sería la respuesta ideal en términos de complejidad espacial y permitiría una búsqueda rápida. Sin embargo, no es posible calcular el grafo de Delaunay en espacios métricos generales y como solución alternativa se opta por construir un árbol, el *SAT* (Spatial Approximation Tree).

Construcción del árbol

Dado un espacio métrico (M, d) y un subconjunto finito de M, $S \subset M$, la construcción del árbol de aproximación espacial se realiza del siguiente modo:

- se selecciona aleatoriamente un prototipo $p \in S$ como raíz del árbol;
- se obtiene el conjunto de vecinos de p, $N(p)$, inicialmente vacío, en el que se incluyen los prototipos que cumplen la siguiente propiedad:

$$\forall s \in S,\ s \in N(p) \iff \forall y \in N(p) - \{s\},\ d(s,y) > d(s,p)$$

  es decir, $N(p)$, constituye el conjunto de prototipos que se encuentran más cerca de p que de cualquier otro prototipo;
- los prototipos que no están en $\{p\} \cup N(p)$ pasan al conjunto de vecinos de su vecino más cercano en $N(p)$;

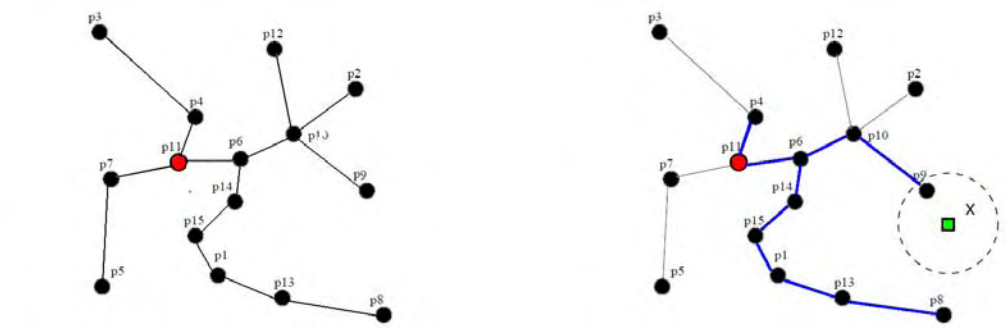


Figura 3.10: Ejemplo de construcción del árbol (izquierda), y de la búsqueda posterior en él (derecha), utilizando el árbol de aproximación espacial (SAT).

- el proceso realizado con p se repite recursivamente para todos sus vecinos en $N(p)$.

Por la forma en la que se define $N(p)$ puede haber más de una posibilidad que satisfaga la definición. Encontrar el conjunto mínimo para $N(p)$ no es un problema trivial, sin embargo, no es necesario trabajar con el conjunto mínimo para que el algoritmo funcione bien. Escogiendo p como raíz, una forma de construir $N(p)$ es ordenando todos los $p_i \in S$ por su distancia a p. Inicialmente $N(p)$ está vacío y se va considerando de manera ordenada cada p_i. Dado un p_i, se comprueba si es más cercano a algún p_i en $N(p)$ que al propio p y, si no es el caso, se añade a $N(p)$.

La estructura que resulta es un árbol, en el que se puede buscar por cualquier $s \in S$ usando la *aproximación espacial*. En la figura 3.10 (a) se muestra para un posible conjunto de prototipos S el árbol SAT resultante tomando como raíz el prototipo p_{11}.

Búsqueda en el árbol

Dada una muestra x para la que se quiere encontrar el vecino más cercano en un conjunto de prototipos S almacenados en un SAT con raíz p (fig. 3.10 (b)):

- la búsqueda comienza comparando x con $\{p\} \cup N(p)$;

- si p está más cercano a x, entonces p es la respuesta;

- si no es así se continúa la búsqueda por el *subárbol* del prototipo más cercano a x en $N(p)$.

Se pueden ahorrar algunas comparaciones durante la búsqueda almacenando en cada nodo t su radio, r_t, es decir, la distancia máxima entre t y cualquier prototipo en el subárbol de raíz t.

Algoritmo 3 `Buscar_NN_SAT`(a, x, k)

Entrada: a; *// a será el nodo raíz*
x *// muestra*
k *// número de vecinos más cercanos*

Salida: A; *// cola de prioridad formada por pares (nodo, d(nodo,x)) ordenada por la distancia*

1: $Q = \{(a, \text{máx}(0, d(q,a) - R(a)), d(q,a))\}$; *// Q es una cola de prioridad de trios donde R(a) es el radio del nodo a*
2: $A = \emptyset$;
3: $r = \infty$;
4: **mientras** $Q \neq \emptyset$ **hacer**
5: $(b, t, m) = elemento\ \ de\ \ Q\ \ con\ \ el\ \ menor\ \ valor\ \ en\ \ t$;
6: $Q = Q - \{(b, t, m)\}$;
7: **si** $t > r$ **entonces**
8: **devolver** A;
9: **fin si**
10: $A = A \cup \{(b, d(x, b))\}$;
11: **si** $|A| = k + 1$ **entonces**
12: $(c, maxd) = elemento\ \ de\ \ A\ \ con\ \ mayor\ \ valor\ \ en\ \ maxd$;
13: $A = A - \{(c, maxd)\}$;
14: **fin si**
15: **si** $|A| = k$ **entonces**
16: $(c, maxd) = elemento\ \ de\ \ A\ \ con\ \ mayor\ \ valor\ \ en\ \ maxd$;
17: $r = maxd$;
18: **fin si**
19: $m = \text{mín}(\{m\} \cup \{d(c, x), c \in N(b)\})$
20: **para todo** $v \in N(b)$ **hacer**
21: $Q = Q \cup (v, \text{máx}(t, m/2, d(x, v) - R(v)), m)$;
22: **fin para**
23: **fin mientras**
24: **devolver** A;

En el algoritmo 3 se muestra el algoritmo de búsqueda de los k vecinos más cercano a una muestra x. Para resolver la búsqueda del vecino más cercano, se comenzaría del mismo modo con $r = \infty$ y se reduciría r cada vez

que una comparación da un valor menor que r.

3.1.6. Comparación entre algoritmos

Para finalizar este repaso, en la figura 3.11, se muestra un ejemplo en el que se refleja gráficamente el modo en el que quedan relacionados los prototipos al almacenarlos siguiendo cada uno de los algoritmos que se acaban de describir. Para el algoritmo propuesto por *Fukunaga y Narendra*, para el *kd-tree* y para el *vp-tree* se muestran los árboles que se obtienen, para el *AESA* la matriz de distancias y para el *SAT* la vinculación de los prototipos según su cercanía espacial.

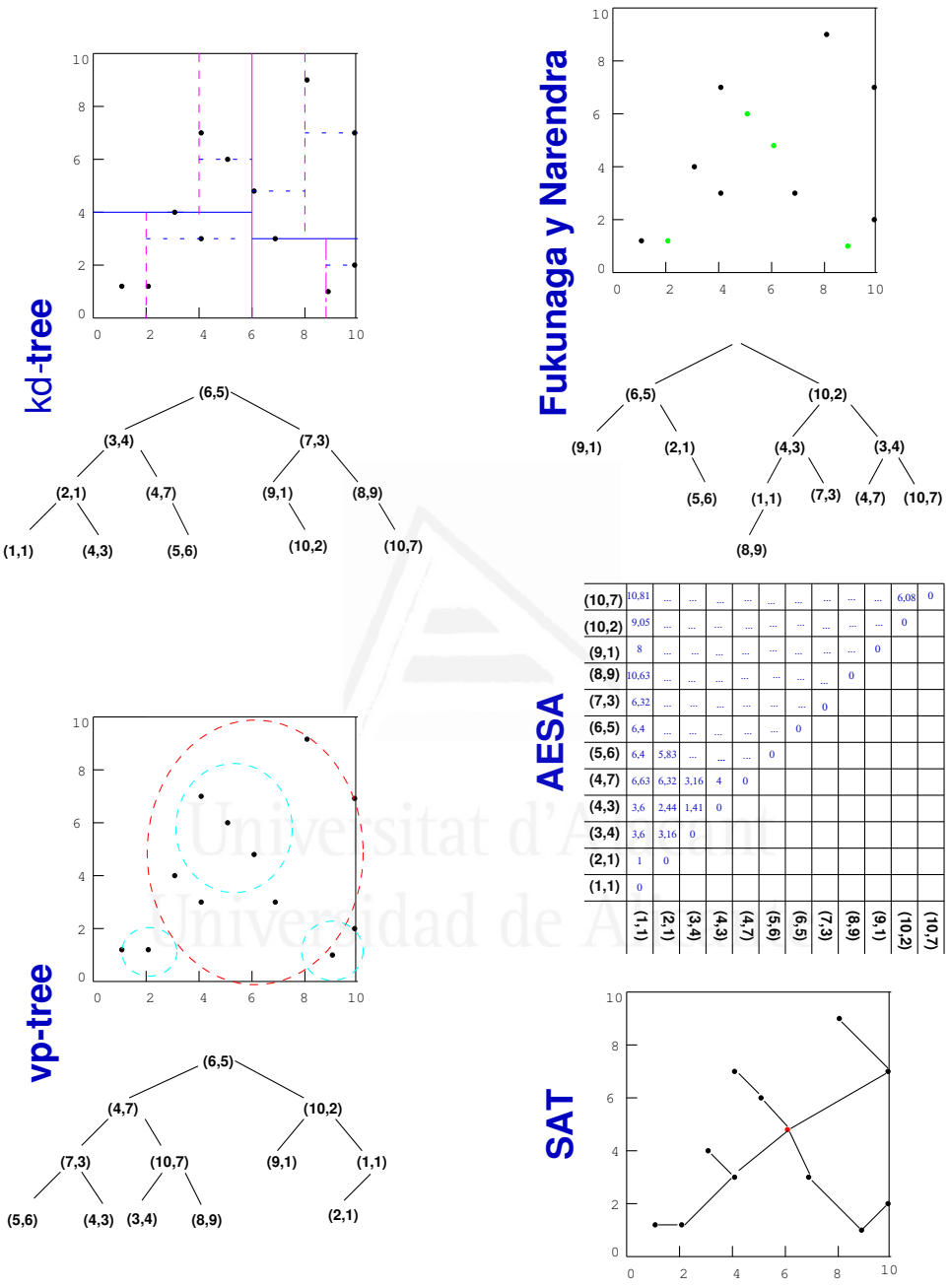


Figura 3.11: Ejemplo del tratamiento que hacen algunos algoritmos rápidos de búsqueda del vecino más cercano de un conjunto de prototipos de dimensión 2, tomando como raíz o pivote de estos algoritmos el prototipo con coordenadas (6,5)).

Parte II

Aportaciones

Capítulo 4

Construcción del árbol

Como se comentó en el capítulo anterior, obtener la respuesta a una consulta implica una búsqueda entre una gran cantidad de datos. Una forma de facilitar esta búsqueda consiste en encontrar una estructura adecuada para almacenar los datos que lleve asociada un método de acceso eficiente a los mismos. En el capítulo 2 se comentaron varios tipos de técnicas: secuenciales, basadas en técnicas hash y basadas en árboles.

En esta tesis utilizamos árboles binarios para representar particiones de los datos a varios niveles. Cada nodo t del árbol representa un conjunto de prototipos S_t. En cada uno de los nodos se almacena

- un representante del nodo (c_t),
- el radio del nodo (r_t), que es la distancia entre el representante del nodo y el prototipo del nodo más alejado al representante,
- y dos referencias a sus dos nodos hijo (fig. 4.1).

Dependiendo del criterio utilizado a la hora de distribuir los prototipos en los nodos del árbol, se obtienen árboles más o menos equilibrados y, lo

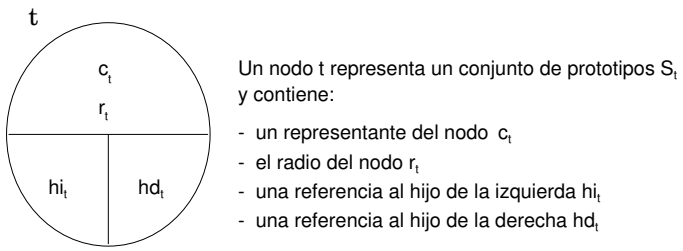


Figura 4.1: Estructura de un nodo del árbol

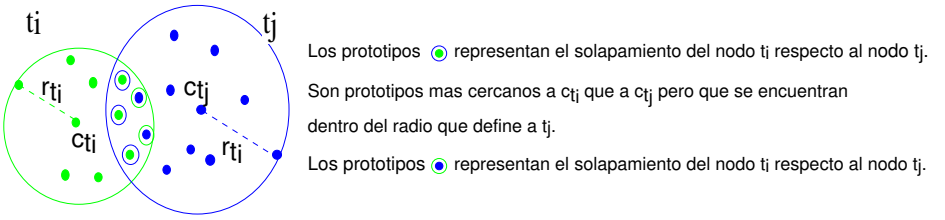


Figura 4.2: Solapamiento de un nodo respecto a su hermano.

que puede ser más relevante para la búsqueda, con nodos con diferente grado de solapamiento.

El solapamiento de un nodo respecto a su nodo hermano es una de las características que se estudia en este capítulo para cada uno de los árboles que proponemos. A continuación definimos de forma más precisa este concepto.

Solapamiento entre nodos

Dados dos nodos t y t', de representantes c_t y $c_{t'}$, y radios r_t y $r_{t'}$ [1], el *solapamiento* de un nodo respecto a su hermano (fig. 4.2) se define como

$$SP(t,t') = \frac{|N(t,t')|}{|N(t)|}$$

donde

$$N(t,t') = \{x | x \in S_t \wedge d(x,c_{t'}) < r_{t'}\} \quad y \quad N(t) = \{x | x \in S_t\}.$$

Es decir, con el solapamiento nos estamos refiriendo a todos los prototipos que se encuentran a la vez dentro de las dos hiperesferas definidas por los representantes de cada nodo y sus radios. El solapamiento del nodo t respecto a su nodo hermano t' se refiere a los prototipos pertenecientes a t que se encuentran dentro de la hiperesfera definida por $c_{t'}$ y $r_{t'}$ y, del mismo modo, el solapamiento de t' respecto a su nodo hermano t se refiere a los prototipos pertenecientes a t' que se encuentran dentro de la hiperesfera definida por c_t y r_t que, como se puede observar en la figura 4.2, no tienen porqué coincidir.

A continuación se definen los métodos propuestos en esta tesis para construir los índices. Estas propuestas se comparan con la técnica utilizada en el algoritmo de Fukunaga y Narendra, y con los métodos *kd-tree*, *SAT* y

[1] Si t es un nodo que aloja a un conjunto de prototipos S_t y tiene como representante c_t, el radio del nodo es $r_t = \max_{x \in S_t} d(x, c_t)$.

vp-tree descritos en el capítulo anterior. En todas las propuestas se ha analizado el solapamiento entre nodos y la profundidad del árbol resultante. Las gráficas y tablas de resultados son las medias obtenidas de 10 experimentos independientes[2]. Para realizar estos experimentos se utilizaron distribuciones uniformes en el hipercubo unidad con distinto número de dimensiones, la función distancia utilizada fue la distancia euclídea.

4.1. El árbol basado en las medianas (*SMM*)

Fukunaga y Narendra (Fukunaga and Narendra, 1975) propusieron la construcción de un árbol mediante la aplicación recursiva del c-means (MacQueen, 1967), tal y como se explicó en el capítulo anterior. En este trabajo nosotros utilizamos el algoritmo c-medoids propuesto por Kaufman y Rousseeuw (Kaufman and Rousseeuw, 1990) donde, a diferencia del c-means, el representante de cada subconjunto se elige entre los prototipos del mismo, en concreto con la mediana del subconjunto de prototipos.

Construcción del árbol

La construción de este árbol, *SMM*[3], se detalla en el algoritmo 4. En este algoritmo, al igual que se considera en los algoritmos propuestos para el resto de árboles de este capítulo, suponemos que partimos de un conjunto S que contiene más de un prototipo.

Inicialmente se construye un nodo raíz donde están todos los prototipos. En este conjunto se aplica el algoritmo c-medoids, que permite obtener las dos medianas $(m_{\mathrm{i}}, m_{\mathrm{d}})$ y una partición del conjunto inicial en dos subconjuntos, S_{i} y S_{d} (paso 5 del algoritmo 4). Con esta información se construyen dos nodos del árbol a partir de los cuales, y de manera recursiva, se continúa creando el árbol *SMM*, tal y como se recoge en el algoritmo 5.

Dado un nodo t que contiene un conjunto de prototipos, S_t, del que se conoce su mediana, la construcción del árbol que parte del nodo t se realiza del siguiente modo:

- se asigna la mediana del nodo, pasada como parámetro, como representante del nodo;

$$c_t = rep$$

[2]Se ha establecido un intervalo de confianza de dos desviaciones típicas en todos los experimentos de esta tesis, casi imperceptible en la mayoría de los casos ya que en general los resultados presentan una desviación típica muy pequeña.

[3]de las siglas en inglés de Sibling Median Median, es decir, los dos nodos hermanos tienen como representantes las dos medianas

- se calcula el radio del nodo t (línea 4 del algoritmo 5);
- se emplea el algoritmo 2_medoids (línea 5 del algoritmo 5) para dividir el conjunto de prototipos del nodo t en dos subconjuntos;
- se crean dos nodos[4], hi_t y hd_t, con los dos conjuntos obtenidos;
- para cada nodo se repiten recursivamente los pasos anteriores, hasta que en cada nodo sólo quede un prototipo, el representante.

Algoritmo 4 `CreaRaiz_SMM(`S`)`

Entrada: S (conjunto de prototipos no vacío);
Salida: t (árbol que representa a S);

```
1: t = CreaNodo();
2: 2-medoids(S, m_i, m_d, S_i, S_d); // de S se obtienen 2 medianas y 2 subconjuntos
3: hi_t=CreaArbol_SMM(S_i, m_i);
4: hd_t=CreaArbol_SMM(S_d, m_d);
5: devolver t;
```

Algoritmo 5 `CreaArbol_SMM(`S, rep`)`

Entrada: S (conjunto de prototipos no vacío);
$rep \in S$;
Salida: t (árbol que representa a S);

```
1: t = CreaNodo();
2: c_t = rep; // el representante es la mediana de S
3: si |S| > 1 entonces
4:    r_t = máx_{p∈S} d(c_t, p);
5:    2-medoids(S, m_i, m_d, S_i, S_d); // de S se obtienen 2 medianas y dos subconjuntos
6:    hi_t=CreaArbol_SMM(S_i, m_i);
7:    hd_t=CreaArbol_SMM(S_d, m_d);
8: fin si
9: devolver t;
```

Profundidad del árbol

La figura 4.3 muestra la partición que se produce en un conjunto de 1000 prototipos a partir de una distribución uniforme de dimensión 2 en los primeros pasos del algoritmo. Como se puede observar, el árbol de las

[4]la estructura de cada nodo es la que se ha indicado al inicio de este capítulo.

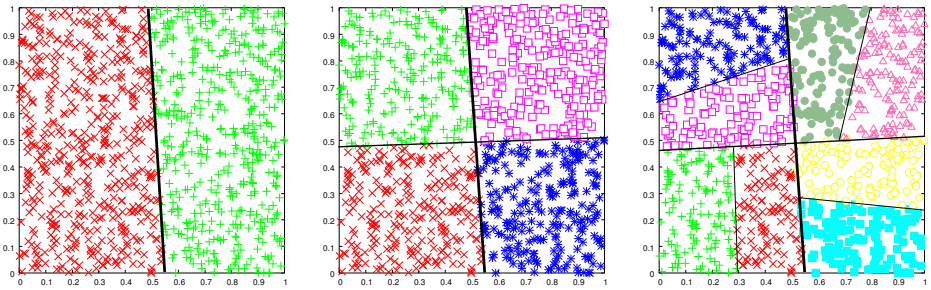


Figura 4.3: División de 1000 puntos de una distribución uniforme en dimensión 2 en las primeras etapas de construcción del árbol de las 2 medianas (*SMM*).

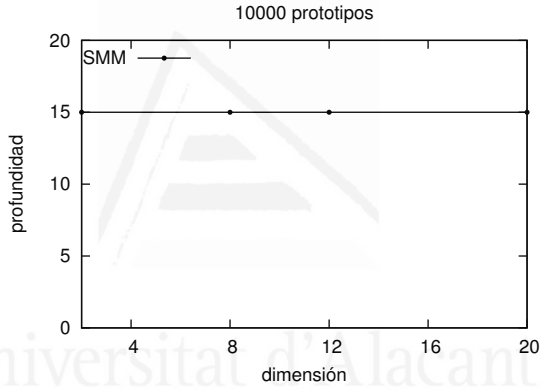


Figura 4.4: Profundidad de un árbol de 10000 prototipos en función de la dimensión en un árbol *SMM*.

medianas es un árbol bastante equilibrado. Esto es debido a que los prototipos están distribuidos homogéneamente en el espacio de trabajo. Al escoger como representantes de los nodos hijo las medianas, los nodos resultantes en cada división tienen un tamaño muy similar, y los representantes de cada nodo se encuentran centrados respecto a los prototipos que constituyen el nodo.

La profundidad de este árbol no varía conforme aumenta la dimensión, como se puede ver en la figura 4.4, y esto es debido al uso de las dos medianas para la distribución de prototipos que, independientemente de la dimensión, por su definición están centradas respecto al conjunto que representan, produciendo un reparto equitativo de los prototipos, figuras 4.3 y 4.5.

Solapamiento entre nodos

En la figura 4.5 se muestran las primeras etapas en la construcción de un árbol *SMM* que almacena 10000 prototipos. Para cada nodo se muestra el número de prototipos que contiene, el radio del nodo y el solapamiento (en tanto por ciento) con su nodo hermano. Las etiquetas de los nodos indican la cantidad de prototipos que le corresponden a cada nodo, por ejemplo, el primer hijo de la izquierda contiene 5028 prototipos $_{5028}$. La circunferencia roja punteada que acompaña a cada nodo indica el solapamiento con su hermano (para el primer hijo de la izquierda $_{45.29}$). El número que se muestra en azul en la parte superior del nodo es el radio del nodo (para el primer hijo de la izquierda es 0.57).

En la figura 4.5 se puede observar que los nodos hermanos tienen radios y tamaños (número de prototipos) bastante similares. En cuanto al solapamiento, uno de los nodos está prácticamente cubierto por el radio del otro. Esto se debe a que, por su posición centrada, los representantes de ambos nodos pertenecen a la hiperesfera definida por el representante y el radio de su nodo hermano.

4.2. Construcción del árbol en base a los prototipos más distantes (*SFF*)

El uso de la media o la mediana como representantes de un conjunto de prototipos es una elección muy usual en las técnicas de clustering (Bock, 2007) (Jain, 2010). Ya que cada nodo t puede ser considerado como una hiperesfera centrada en el representante c_t de radio r_t, el hecho de trabajar con las dos medianas no evita que se presente un solapamiento entre nodos hermanos.

El solapamiento es un problema que afecta a la etapa de búsqueda ya que, como se puede comprobar en numerosos trabajos (Traina et al., 2000), la búsqueda empeora a medida que aumenta el solapamiento entre nodos del árbol. Dicho de otra manera, a medida que aumenta el solapamiento disminuye la posibilidad de realizar podas en el árbol.

Para conseguir reducir el solapamiento, una estrategia sencilla es elegir los representantes de los nodos de forma que estén lo más alejados entre sí. Esta idea da lugar a los árboles *SFF*[5].

[5]de las siglas en inglés de Sibling Far Far, es decir, los dos nodos hermanos tienen como representantes los dos prototipos más alejados.

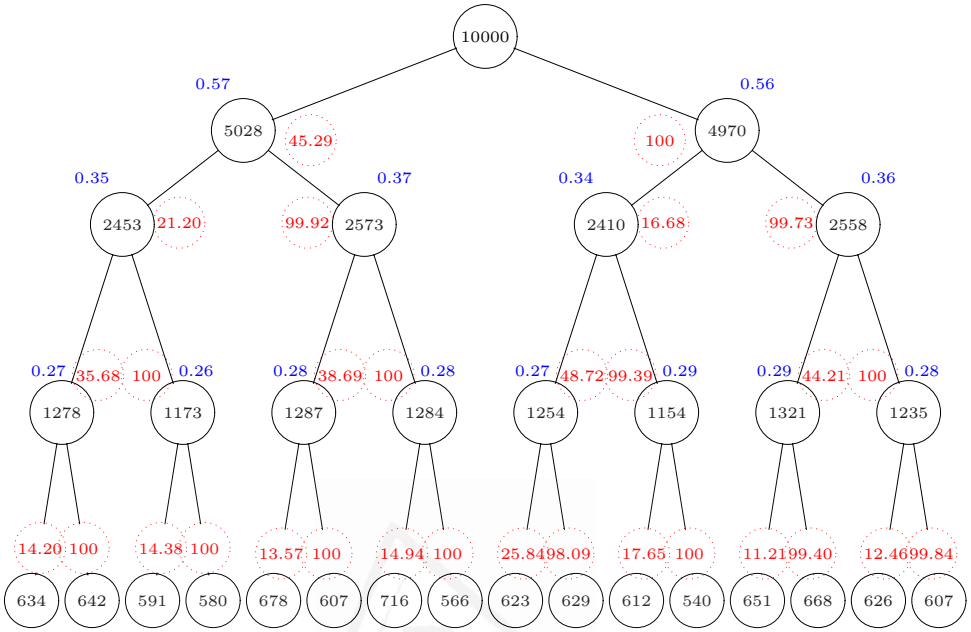


Figura 4.5: Parte superior de un árbol *SMM* para 10000 prototipos extraídos de una distribución uniforme en el hipercubo unidad en dimensión 2 utilizando la distancia euclídea.

Construcción del árbol

La construcción de este árbol se detalla en el algoritmo 6. Inicialmente se construye un nodo raíz donde están todos los prototipos. En este conjunto se eligen los dos prototipos más alejados como representantes (línea 2), a partir de los cuales se continúa creando el árbol *SFF* tal y como se recoge en el algoritmo 7.

Dado un nodo t que contiene un conjunto de prototipos, S_t, del que se conoce su representante c_t, la construcción del árbol que parte del nodo t se realiza del siguiente modo:

- se asigna el representante, pasado como parámetro

$$c_t = rep;$$

- se calcula el radio del nodo t (línea 4);
- se obtienen los dos prototipos p y $q \in S_t$, más alejados en S (línea 5):

$$d(p,q) \geq d(x,y) \quad \forall x, y \in S_t;$$

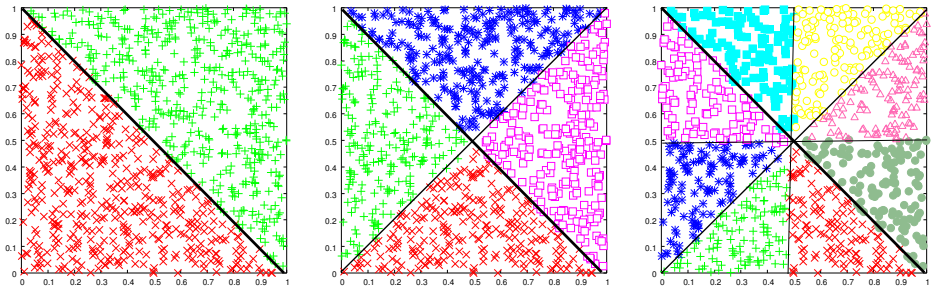


Figura 4.6: División de 1000 puntos de una distribución uniforme en dimensión 2, en las primeras etapas de construcción del árbol *SFF*.

- se crean dos nodos hi_t y hd_t que tienen como representantes los dos prototipos elegidos y donde se distribuyen los prototipos según su distancia a cada representante (líneas 8 y 9). Al hacer la distribución de prototipos no se consideran los prototipos que son representantes (líneas 6 y 7);
- para cada nodo se repiten recursivamente los pasos anteriores, hasta que en cada nodo sólo quede un prototipo, el representante.

En el peor caso, cuando el árbol degenera a una lista, la complejidad temporal de construcción del árbol es $O(n^3)$, donde n es el número de prototipos. Sin embargo, en la mejor situación, el árbol es equilibrado (fig. 4.6), y el coste es $O(n^2)$.

Algoritmo 6 `CreaRaiz_SFF(`S`)`

Entrada: S (conjunto de prototipos no vacío);
Salida: t (árbol que representa a S);
1: $t =$ `CreaNodo()`;
2: $(c_\mathrm{i}, c_\mathrm{d}) = \arg\max_{(p,q\in S^2)} d(p,q)$;
3: $S_\mathrm{i} = \{p \in S | d(p,c_\mathrm{i}) < d(p,c_\mathrm{d})\} - \{c_\mathrm{i}\}$;
4: $S_\mathrm{d} = S - S_\mathrm{i} - \{c_\mathrm{d}\}$;
5: $hi_t =$`CreaArbol_SFF(`$S_\mathrm{i}, c_\mathrm{i}$`)`;
6: $hd_t =$`CreaArbol_SFF(`$S_\mathrm{d}, c_\mathrm{d}$`)`;
7: **devolver** t;

Profundidad del árbol

La figura 4.6 muestra la partición que se produce en un conjunto de 1000 prototipos de dimensión 2 en los primeros pasos del algoritmo. Como

Algoritmo 7 `CreaArbol_SFF`(S, rep)

Entrada: S (conjunto de prototipos no vacío);
$rep \in S$;
Salida: t (nodo de árbol);

```
1: t = CreaNodo();
2: c_t = rep;
3: si |S| > 1 entonces
4:    r_t = máx_{p∈S} d(c_t, p); // radio del nodo
5:    (c_i, c_d) = arg máx_{(p,q)∈S²} d(p, q);
6:    S_i = {p ∈ S | d(p, c_i) < d(p, c_d)} − {c_i};
7:    S_d = S − S_i − {c_d};
8:    hi_t=CreaArbol_SFF(S_i, c_i);
9:    hd_t=CreaArbol_SFF(S_d, c_d);
10: fin si
11: devolver t;
```

se puede observar, el reparto de prototipos entre los nuevos nodos que se van formando es bastante equitativo, como también se puede observar de modo cuantitativo por el número de prototipos contenidos en los nodos hermanos en la figura 4.8.

Por otro lado, la profundidad del árbol no varía prácticamente con el aumento de dimensión de los prototipos (figura 4.7) que muestra la media de 10 experimentos para una conjunto de 100000 prototipos en una distribución uniforme en el hipercubo unidad. Al igual que en el caso del árbol *SMM* donde se usaba el c-medoids, en este caso la distribución de prototipos va a ser bastante similar entre dos nodos hermanos independientemente de la dimensión.

Solapamiento entre nodos

En la figura 4.8 se muestran las primeras etapas en la construcción de un árbol SFF que almacena 100000 prototipos. Para cada nodo se muestra el número de prototipos que contiene, el radio del nodo y el solapamiento (en tanto por ciento) con su nodo hermano.

Como ya se ha comentado y se puede observar en la figura 4.8, el reparto de prototipos entre nodos hermanos es bastante equilibrado; se observa también que los radios de dos nodos hermanos, así como el solapamiento de uno respecto a otro presentan valores similares. En esta estructura de árbol el solapamiento de los nodos hermanos en los primeros niveles de construcción del árbol es bastante elevado, ya en dimensión 2 es de alrededor de un 50 %,

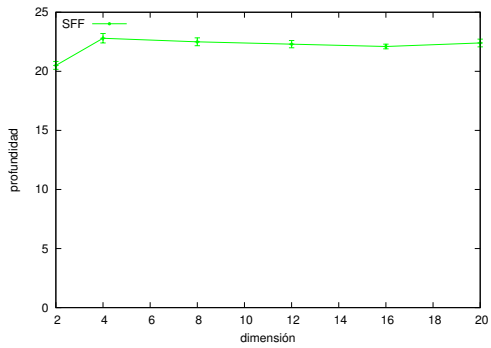


Figura 4.7: Profundidad en función de la dimensión para un árbol *SFF* con 100000 prototipos.

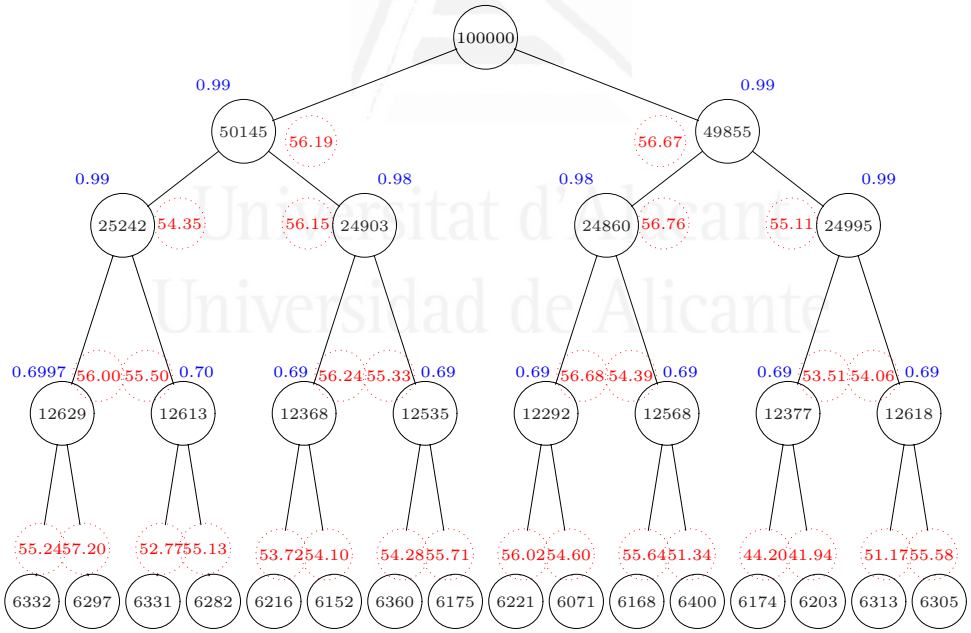


Figura 4.8: Parte superior de un árbol *SFF* para 100000 prototipos extraídos de una distribución uniforme en el hipercubo unidad en dimensión 2, usando la distancia euclídea.

		prototipos por nodo					prototipos por nodo				
		$3-10$	$11-100$	$101-1000$	$1001-10000$	$10001-100000$	$3-10$	$11-100$	$101-1000$	$1001-10000$	$10001-100000$
	% nodos respecto al total	69,93	26,84	2,95	0,26	0,03	69,49	27,07	3,11	0,3	0,03
solapamiento con el hermano(%)	*nodos sin solapamiento*	80,3	28,59	0	0	0	56,76	8	0,01	0	0
	$0-10$	0	12,83	0,61	0	0	0	6,82	0,28	0	0
	$10-20$	3,9	15,77	5,37	0	0	4,9	11,05	1,45	0	0
	$20-30$	4,56	14,93	13,44	0	0	7,79	12,86	3,24	0,23	0
	$30-40$	3,22	12,11	26,21	0,18	0	6,56	12,48	6,7	0,99	0
	$40-50$	1,31	8,22	34,78	25,09	0	2,94	11,35	10,63	4,85	0
	$50-60$	3,6	4,83	18,91	74,73	100	9,22	11,95	14,73	18,35	0,77
	$60-70$	1,36	1,82	0,67	0	0	4,4	9,46	17,67	29,28	28,98
	$70-80$	0,5	0,56	0	0	0	2,07	7,21	18,68	28,16	52,19
	$80-90$	0,32	0,23	0	0	0	1,53	5,38	16,55	16,09	18,06
	$90-100$	0,93	0,1	0	0	0	3,85	3,42	10,07	2,06	0
		dimensión 2					dimensión 8				

Tabla 4.1: Porcentaje de nodos, respecto al total, que pertenecen a cada uno de los intervalos de población de nodo considerados en un árbol *SFF* de 100000 prototipos. Para cada intervalo de población se muestra el porcentaje de nodos de ese intervalo que se encuentran dentro de un determinado rango de solapamiento con su nodo hermano.

como se puede observar. Esto se puede deber a que al elegir como representantes de los nodos los dos prototipos más alejados entre sí, el radio de los nodos es muy grande y envuelve alrededor de la mitad de los prototipos del nodo hermano.

También se analizó el solapamiento entre nodos hermanos en varias dimensiones. En la tabla 4.1 se muestran los valores medios obtenidos[6] para 10 conjuntos diferentes de 100000 prototipos en dimensión 2 y en dimensión 8. En primer lugar se muestra el porcentaje de nodos, respecto al total de nodos que constituyen el árbol, para diferentes intervalos de tamaño[7] de un

[6]El intervalo de error de estos datos no supera el 2 % de la medida excepto en los porcentajes referidos a nodos con una cantidad de prototipos entre 1001 y 10000 y entre 10001 y 100000 debido a que existen muy pocos nodos de estos tamaños.

[7]Por tamaño de un nodo entendemos el número total de prototipos que contiene.

nodo. Se han considerado 5 intervalos distintos, mostrándose el porcentaje de nodos con un tamaño entre 3 y 10 prototipos, entre 11 y 100, entre 101 y 1000, entre 1001 y 10000 y, por último, los que contienen entre 10001 y 100000 prototipos.

Como cabía de esperar, y se observa en la tabla, la mayoría de los nodos del árbol son nodos con pocos prototipos, los que tienen entre 3 y 10 prototipos constituyen casi el 70 % de los nodos del árbol y los que tienen entre 11 y 100 más del 25 % de los nodos, no llegando a un 5 % los nodos para los otros intervalos de población. El solapamiento con el nodo hermano aumenta conforme aumenta la dimensión para los nodos de mayor población. En dimensión 2, este porcentaje está entre el 50 y el 60 %, como también se observaba en la figura 4.8, mientras que para dimensión 8 la mayoría de los nodos de mayor población presentan un solapamiento con el nodo hermano en torno al 70-80 %. Si observamos el solapamiento en los nodos de menor población, vemos que mientras que en dimensión 2 un 80 % de los nodos no presentan solapamiento con su nodo hermano, en dimensión 8 este porcentaje ha disminuido, acercándose al 50 % los nodos que no presentan solapamiento.

4.3. Construcción del árbol en base a la mediana y al prototipo más alejado a ésta (*SMF*)

Los resultados de solapamiento obtenidos para los árboles *SFF* hacen pensar que al abordar posteriormente la búsqueda habrá que recorrer más nodos conforme va aumentando la dimensión de los datos, ya que aumenta el solapamiento entre nodos hermanos y, por tanto, será más difícil realizar podas en la búsqueda.

En lugar de mantener los nodos hermanos con una cantidad similar de prototipos, como ocurre con los árboles *SMM* y *SFF*, nos planteamos cambiar la estrategia y estudiar lo que ocurriría si el árbol no fuese equilibrado. Se ha optado entonces por otra construcción en la que, para dividir un nodo, se utilizan como representantes de sus nodos hijo la mediana del nodo y el prototipo más alejado a la mediana, dando lugar a los árboles *SMF*[8]. Lo que ocurre en este caso es que un nodo contendrá la mayoría de los prototipos mientras que el otro tendrá un número mucho más reducido. Esta nueva situación provoca que uno de los nodos tenga un radio mucho menor que el otro, y se trata de estudiar cómo afecta al solapamiento entre nodos hermanos.

[8]de las siglas en inglés de Sibling Median Far, es decir, los dos nodos hermanos tienen como representantes la mediana y el prototipo más alejado a la misma.

Construcción del árbol

La construcción de este árbol se detalla en el algoritmo 8. Inicialmente se construye un nodo raíz donde están todos los prototipos. En este conjunto se elige su mediana y el prototipo más alejado a la misma (líneas 2 y 3), que serán los representantes de los nodos hijo y, a partir de ellos y de forma recursiva, se continúa creando el árbol *SMF* tal y como se recoge en el algoritmo 9.

Dado un nodo t que contiene un conjunto de prototipos, S_t, del que se conoce su representante c_t, la construcción del árbol que parte del nodo t se realiza del siguiente modo:

- se asigna el representante, pasado como parámetro (línea 2)

$$c_t = rep;$$

- se calcula el radio del nodo t (línea 4);
- se obtienen la mediana de t, c_{i}, y el prototipo más alejado a la ésta, c_{d}

$$c_{\mathrm{i}} = \arg\min_{p \in S_t} \sum_{x \in S_t} d(p,x) \quad y \quad c_{\mathrm{d}} = \arg\max_{s \in S_t} d(p,s);$$

- se crean dos nodos hi_t y hd_t que tienen como representantes los dos prototipos elegidos y donde se distribuyen los prototipos según su distancia a cada representante (líneas 7-10);
- recursivamente se repiten los pasos anteriores, hasta que en cada nodo sólo quede un prototipo, el representante.

Al igual que se hacía al construir los árboles *SFF*, en la construcción de este árbol los prototipos que se van seleccionando como representantes de algún nodo se excluyen del conjunto de prototipos a asignar a los nodos hijo (líneas 4 y 5 del algoritmo 8 y líneas 7 y 8 del algoritmo 9).

Profundidad del árbol

Esta forma de construir el árbol hace que la distribución de los prototipos en los nodos no sea tan equitativa como en los casos anteriores, como se puede observar en la figura 4.9, dando lugar a árboles que ya no están equilibrados. Además de esta falta de equilibrio, se observa un aumento de la profundidad del árbol comparada con la de los árboles SFF, como se puede observar en la figura 4.10. Por otro lado, el aumento de la dimensión de los datos no supone

Algoritmo 8 `CreaRaiz_SMF`(S)

Entrada: S (conjunto de prototipos no vacío);
Salida: t (árbol que representa a S);

1: t = `CreaNodo`();
2: c_{i} = `Mediana`(S);
3: $c_{\mathrm{d}} = \arg\max_{(p \in S)} d(p, c_i)$;
4: $S_{\mathrm{i}} = \{p \in S | d(p, c_{\mathrm{i}}) < d(p, c_{\mathrm{d}})\} - \{c_{\mathrm{i}}\}$;
5: $S_{\mathrm{d}} = S - S_{\mathrm{i}} - \{c_{\mathrm{d}}\}$;
6: hi_t=`CreaArbol_SMF`$(S_{\mathrm{i}}, c_{\mathrm{i}})$;
7: hd_t=`CreaArbol_SMF`$(S_{\mathrm{d}}, c_{\mathrm{d}})$;
8: **devolver** t;

Algoritmo 9 `CreaArbol_SMF`(t, S, rep)

Entrada: S (conjunto de prototipos no vacío);
$rep \in S$;
Salida: t (árbol que representa a S)

1: t = `CreaNodo`();
2: $c_t = rep$;
3: **si** $|S| > 1$ **entonces**
4: $r_t = \max_{p \in S} d(c_t, p)$; *// radio del nodo*
5: c_{i} = `Mediana`(S);
6: $c_{\mathrm{d}} = \arg\max_{(p \in S)} d(p, c_i)$;
7: $S_{\mathrm{i}} = \{p \in S | d(p, c_{\mathrm{i}}) < d(p, c_{\mathrm{d}})\} - \{c_{\mathrm{i}}\}$;
8: $S_{\mathrm{d}} = S - S_{\mathrm{i}} - \{c_{\mathrm{d}}\}$;
9: hi_t=`CreaArbol_SMF`$(S_{\mathrm{i}}, c_{\mathrm{i}})$;
10: hd_t=`CreaArbol_SMF`$(S_{\mathrm{d}}, c_{\mathrm{d}})$;
11: **fin si**
12: **devolver** t;

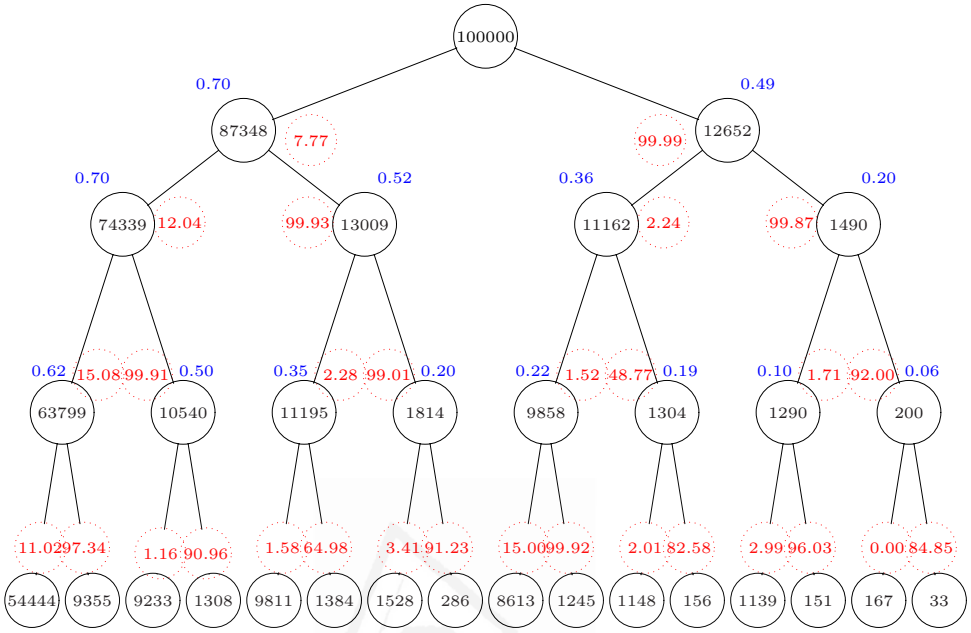


Figura 4.9: Parte superior de un árbol *SMF* para 100000 prototipos extraídos de una distribución uniforme en el hipercubo unidad en dimensión 2 usando la distancia euclídea.

un aumento en la profundidad del árbol que los representa Esto es debido a que en cada división de un nodo se calcula siempre la mediana del conjunto y el prototipo más alejado a la misma y, al calcular la mediana, el número de prototipos que se agrupan en torno a ella es similar independientemente de la dimensión.

Solapamiento entre nodos

En la tabla 4.2 se observa cómo el porcentaje de nodos que corresponden a las poblaciones más pequeñas, entre 3 y 10 prototipos y entre 11 y 100, siguen representando casi la totalidad de los nodos del árbol, sin embargo a diferencia de los árboles *SFF*, en los árboles *SMF* el aumento de la dimensionalidad de los datos no afecta tanto al porcentaje de nodos que no presentan solapamiento, característica que sí se observa en los árboles *SFF* (tabla 4.1). Por otro lado, dejando a un lado los nodos con muy pocos prototipos, que presentan unos porcentajes muy similares en cuanto a nodos sin solapamiento o con un solapamiento pequeño, para el resto de intervalos de población los nodos de los árboles *SFF* presentan un mayor solapamiento con el nodo

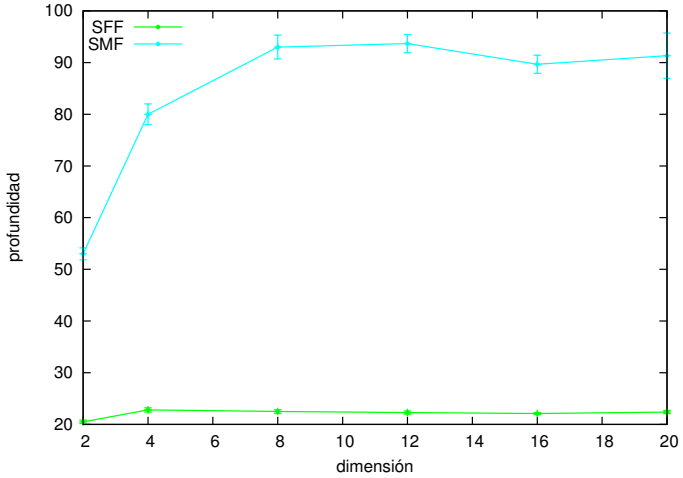


Figura 4.10: Los árboles *SMF* son mucho más profundos que los árboles *SFF*, como se puede observar en esta gráfica para una población del nodo raíz de 100000 prototipos.

hermano que los árboles *SMF*.

Para el segundo intervalo de población, el de los nodos que tienen entre 11 y 100 prototipos, podemos observar cómo para dimensión 2, en los árboles *SMF* el porcentaje de nodos sin solapamiento junto con los que tienen un solapamiento que no supera el 10 % representa más del 60 % de los nodos de este intervalo, mientras que en los árboles *SFF* son sólo el 40 %. Esta diferencia se hace más notoria en dimensión 8, donde alrededor del 45 % de los nodos no presentan solapamiento o tienen un solapamiento inferior al 10 % frente al 15 % que se encuentran en esta situación en los árboles *SFF*. Por otro lado, con independencia de la dimensión de los datos, se puede observar comparando los resultados de la tabla 4.2 con la tabla 4.1, cómo para las poblaciones más grandes, de 1001 a 10000 y de 10001 a 100000, los árboles *SMF* no suelen tener porcentaje de solapamiento alto mientras que en los árboles *SFF* ocurre justo al revés y lo que no encontramos son nodos sin solapamiento o con solapamiento reducido.

4.4. Construcción del árbol en base al prototipo más distante del padre (*PRF*)

Esta aproximación se basa en una técnica empleada en el algoritmo TLAESA (Micó et al., 1996). Aunque ya hablamos más adelante de la bús-

		prototipos por nodo					prototipos por nodo				
		$3-10$	$11-100$	$101-1000$	$1001-10000$	$10001-100000$	$3-10$	$11-100$	$101-1000$	$1001-10000$	$10001-100000$
	% nodos respecto al total	65,66	29,61	4,24	0,45	0,04	65,54	29,18	4,22	0,96	0,09
solapamiento con el hermano (%)	*nodos sin solapamiento*	68,1	46,23	8,15	0,05	0	60,88	23,59	1,44	0	0
	$0-10$	0	17,1	49,31	47,82	31,4	0	19,6	21,6	8,35	0
	$10-20$	1,86	8,08	21,36	33,55	43,1	2,91	16,56	28,14	39,37	15,79
	$20-30$	2,55	2,5	3,42	3,8	9,53	3,45	9,07	19,78	30,83	63,96
	$30-40$	2,54	1,03	0,04	0	0	3,12	4,16	7,97	11,14	19,75
	$40-50$	1,03	1,06	0,11	0,05	0	1,15	1,99	2,39	1,56	0,5
	$50-60$	5,58	1,91	0,24	0,5	0	6,45	1,72	0,31	0,14	0
	$60-70$	3,36	2,44	0,61	0,61	0	3,73	2,03	0,03	0	0
	$70-80$	2,09	3,15	1,19	0,56	0	2,34	2,72	0,1	0	0
	$80-90$	2,45	4,74	3,13	1,52	0	3,07	4,67	0,91	0	0
	$90-100$	10,43	11,76	12,44	11,54	15,96	12,89	13,88	17,33	8,61	0
		dimensión 2					dimensión 8				

Tabla 4.2: Porcentaje de nodos, respecto al total, que pertenecen a cada uno de los intervalos de población de nodo considerados en un árbol *SMF* de 100000 prototipos. Para cada intervalo de población se muestra el porcentaje de nodos de ese intervalo que se encuentran dentro de un determinado rango de solapamiento con su nodo hermano.

queda, la técnica consiste en repetir un representante (de nodo padre a hijo) para ahorrar distancias a calcular en la búsqueda. Normalmente, durante la búsqueda en un árbol, se calcula la distancia de la muestra a los representantes de los dos hijos del nodo en que nos encontramos para decidir en base a esta información por donde continuar la búsqueda.

En esta técnica, *PRF*[9], se evita uno de los cálculos de distancias en la búsqueda, usando como representante del hijo de la izquierda el mismo que el de su nodo padre. En este árbol, el representante del hijo de la derecha es el prototipo más alejado del representante del hijo de la izquierda, o lo

[9]de las siglas en inglés de Parent Random Far, es decir, se mantiene el representante del padre en uno de los hijos (elegido el representante del raíz aleatoriamente) y el prototipo más alejado como representante del hermano.

que es lo mismo, del representante de su padre[10]. Este árbol se puede ver como una particularización del mb-tree en el que el representante del nodo hermano puede ser elegido aleatoriamente (Noltemeier et al., 1992).

En esta propuesta el árbol no está tan equilibrado como cuando se emplea la técnica *SFF*, ya que desde las primeras divisiones una de las ramas del árbol contiene muchos más prototipos que la otra (fig 4.11), al ir asignando a uno de los nodos hijo los prototipos más alejados del representante del nodo a dividir, mientras que en el otro nodo hijo permanecerán la gran mayoría de los prototipos.

En el peor caso, cuando el árbol degenera a una lista, la complejidad temporal de construcción del árbol es $O(n^2)$, donde n es el número de prototipos. Sin embargo, cuando el árbol está equilibrado, la complejidad es $O(n\log(n))$.

En la técnica *PRF* se han estudiado diferentes inicializaciones de la raíz del árbol. El objetivo de este estudio era determinar si la elección del representante del nodo raíz tiene incidencia o no en la estructura del árbol y, por lo tanto, si tiene repercusión en la posterior búsqueda. A continuación se analizan dos inicializaciones.

4.4.1. Un prototipo al azar como representante del nodo raíz

La estrategia más sencilla es elegir aleatoriamente el representante del nodo raíz. A la construcción del árbol siguiendo esta estrategia la vamos a seguir denominando *PRF*. La construción de este árbol se detalla en el algoritmo 10.

Inicialmente se construye un nodo raíz, donde están todos los prototipos. En este conjunto se selecciona aletoriamente un prototipo y el prototipo más alejado a éste (líneas 2 y 3 del algoritmo 10). Con esta información se construyen dos nodos del árbol a partir de los cuales, y de manera recursiva, se continúa creando el árbol *PRF*, tal y como se recoge en el algoritmo 11.

Dado un nodo t que contiene un conjunto de prototipos, S_t, del que se conoce su representante, c_t, la construcción del árbol que parte del nodo t se realiza del siguiente modo (algoritmo 11):

- se asigna el representante, pasado como parámetro (línea 2)

$$c_t = rep;$$

- se calcula el radio del nodo t (línea 4 del algoritmo 11);

[10]Publicaciones derivadas de este estudio (Gómez-Ballester et al., 2006) y (Gómez-Ballester et al., 2003). La notación empleada en los artículos se ha modificado en la tesis para facilitar la lectura. El árbol PRF se nombraba en el artículo como MDF y el árbol SFF como MSP.

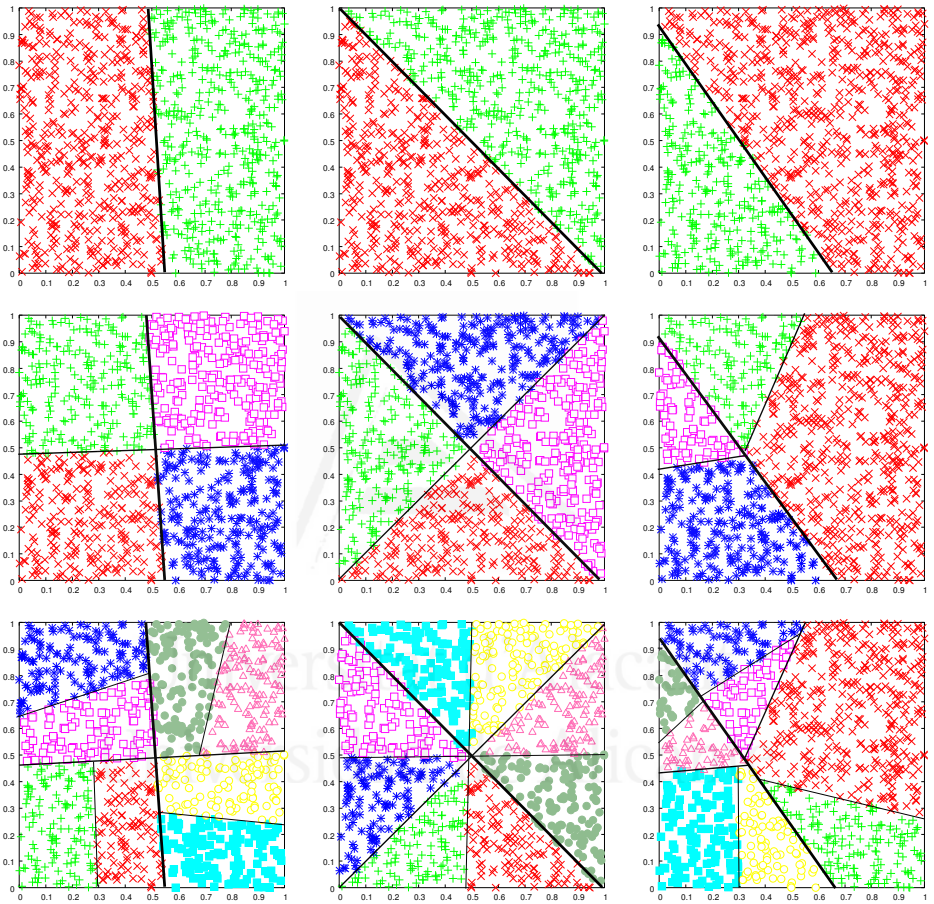


Figura 4.11: La columna de la izquierda, muestra la forma en que se dividen 1000 puntos de una distribución uniforme en dimensión 2, en las primeras etapas de construcción del árbol siguiendo la estrategia de las medianas, *SMM*. La columna de en medio muestra el comportamiento cuando se usa la estrategia *SFF* y la columna de la derecha cuando se usa la estrategia *PRF*.

- se seleccionan los representantes de los dos hijos, el de la izquierda es el mismo que el del padre, y el de la derecha el más alejado de éste (líneas 5 y 6)
$$c_{\mathrm{d}} = \arg\max_{s \in S_t} d(c_{\mathrm{t}}, s);$$
- se crean dos nodos hi_t y hd_t que tienen como representantes c_{i} y c_{d} y donde se distribuyen los prototipos según su distancia a cada representante;
- para cada nodo se repiten recursivamente los pasos anteriores hasta que en cada nodo sólo quede un prototipo, su representante.

Algoritmo 10 `CreaRaiz_PRF(`S`)`

Entrada: S (conjunto de prototipos no vacío);
Salida: t (árbol que representa a S);
1: t = `CreaNodo()`;
2: c_{i} = elemento arbitrario de S;
3: $c_{\mathrm{d}} = \arg\max_{(p \in S)} d(p, c_i)$;
4: $S_{\mathrm{i}} = \{p \in S | d(p, c_{\mathrm{i}}) < d(p, c_{\mathrm{d}})\}$;
5: $S_{\mathrm{d}} = S - S_{\mathrm{i}}$;
6: hi_t=`CreaArbol_PRF(`$S_{\mathrm{i}}, m_{\mathrm{i}}$`)`;
7: hd_t=`CreaArbol_PRF(`$S_{\mathrm{d}}, m_{\mathrm{d}}$`)`;
8: **devolver** t;

Algoritmo 11 `CreaArbol_PRF(`S, rep`)`

Entrada: S (conjunto de prototipos no vacío);
$rep \in S$;
1: t = `CreaNodo()`;
2: $c_t = rep$;
3: **si** $|S| > 1$ **entonces**
4: $r_t = \max_{p \in S} d(c_t, p)$; *// radio del nodo*
5: $c_{\mathrm{i}} = rep$; *// mismo representante que el padre*
6: $c_{\mathrm{d}} = \arg\max_{(p \in S)} d(p, c_i)$;
7: $S_{\mathrm{i}} = \{p \in S | d(p, c_{\mathrm{i}}) < d(p, c_{\mathrm{d}})\}$;
8: $S_{\mathrm{d}} = S - S_{\mathrm{i}}$;
9: hi_t=`CreaArbol_PRF(`$S_{\mathrm{i}}, c_{\mathrm{i}}$`)`;
10: hd_t=`CreaArbol_PRF(`$S_{\mathrm{d}}, c_{\mathrm{d}}$`)`;
11: **fin si**
12: **devolver** t;

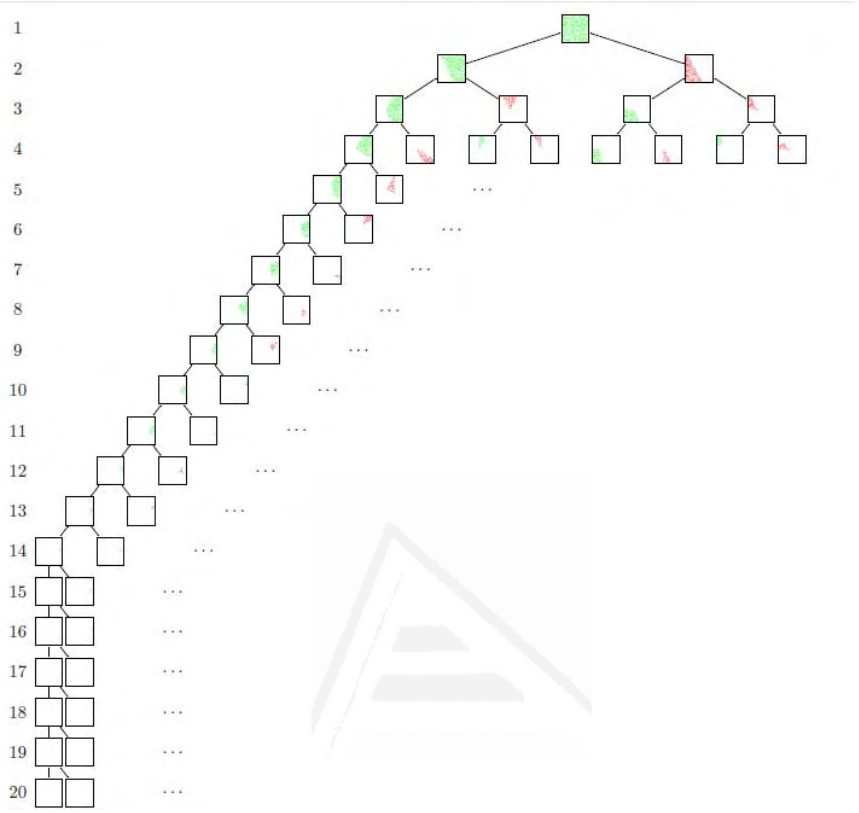


Figura 4.12: Rama más profunda de la construcción del árbol *PRF* para 1000 prototipos de una distribución uniforme en dimensión 2.

Profundidad del árbol

Esta técnica genera árboles en los que una de las ramas contiene más prototipos que la otra, como ocurre con los árboles *SMF*. En la figura 4.12 se puede apreciar gráficamente este comportamiento, como consecuencia una de las ramas es mucho más profunda que la otra. En la figura 4.14 se muestran los primeros niveles en el proceso de construcción de un árbol a partir de 100000 prototipos de dimensión 2 en el hipercubo unidad donde se puede apreciar con valores concretos esta diferencia de población entre nodos hermanos.

La profundidad de los árboles *PRF*, como se puede observar en la figura 4.13, presenta una mayor variabilidad para cada dimensión que los otros árboles debido a que la elección del representante del nodo raíz al azar determina que podamos partir, desde el primer nivel de construcción del árbol, con nodos de población muy diferente.

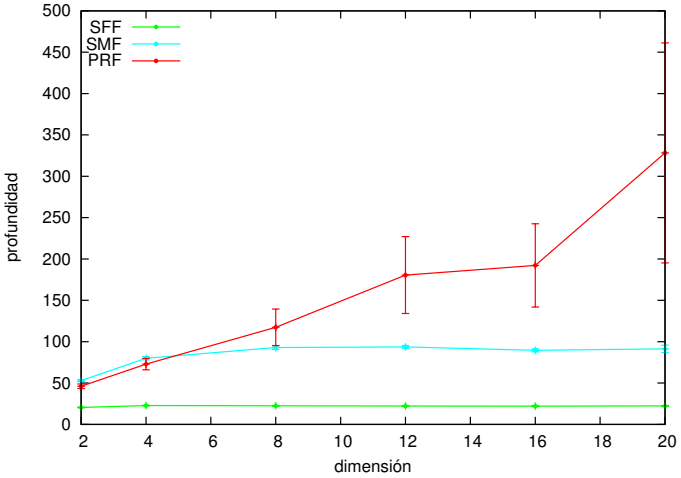


Figura 4.13: Profundidad de los árboles PRF, *SFF* y *SMF* para una población del nodo raíz de 100000 prototipos[11].

A pesar de esta variabilidad, se observa con claridad que los árboles *PRF* son más profundos que los dos anteriores. Con respecto a los árboles *SFF* esta diferencia se debe al desequilibrio que existe en los árboles *PRF* entre el número de prototipos que se asignan entre los nodos hermanos, frente al equilibrio de los árboles *SFF*. Sin embargo, en los árboles *SMF* el desequilibrio entre los prototipos asignados a cada nodo es mayor que en los árboles *PRF* por lo que se podría esperar que la profundidad de los árboles *SMF* fuese mayor que la de los árboles *PRF*. Esto sería así si no se fuesen excluyendo prototipos durante la construcción de los árboles *SMF*. Tal y como se comentó en el apartado anterior, al construir los árboles *SMF*, los prototipos que van siendo considerados representantes de algún nodo ya no son considerados para formar parte de los nodos hijo. Esto no ocurre en los árboles *PRF*, ya que al mantener el prototipo representante de un nodo como representante del hijo de la izquierda no se van excluyendo prototipos.

También se observa que la profundidad del árbol crece con la dimensión. Este aumento se debe a que, al aumentar la dimensión, la distancia del prototipo más alejado del representante del nodo raíz es cada vez mayor provocando un aumento del desequilibrio en el reparto de prototipos del nodo raíz, y con ello un aumento de la profundidad. Los árboles *SFF* y *SMF* no se ven afectados por el aumento de dimensión ya que la presencia de outliers no afecta sólo al representante de uno de los hijos de la raíz sino a los dos.

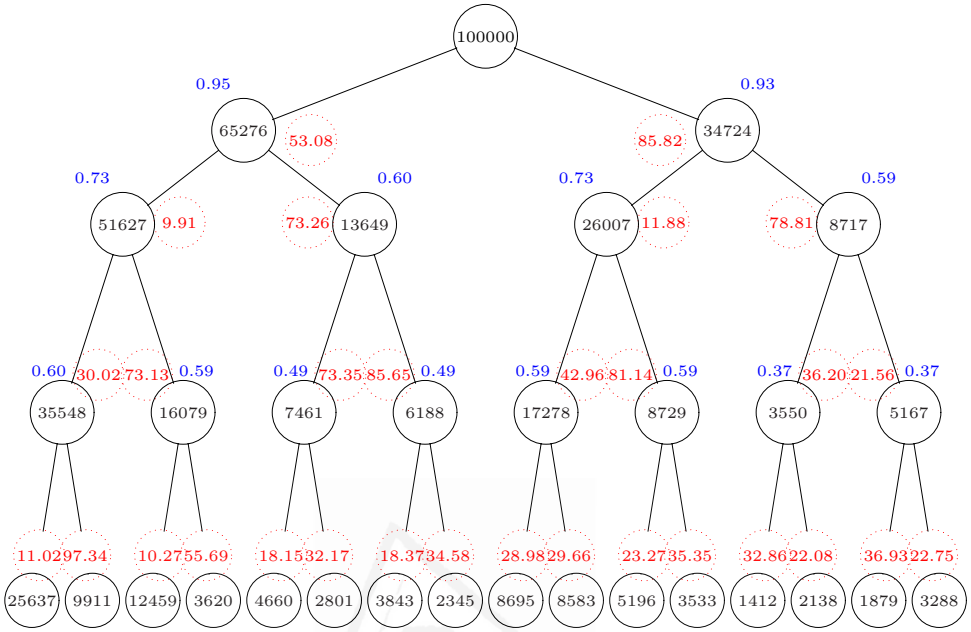


Figura 4.14: Parte superior de un árbol *PRF* para 100000 prototipos extraídos de una distribución uniforme en el hipercubo unidad en dimensión 2 usando la distancia euclídea.

Solapamiento entre nodos

Al estudiar los datos sobre el solapamiento que se muestran en la tabla 4.3 se deduce que los árboles *PRF* presentan un solapamiento inferior al de los árboles *SFF*, pero superior al de los árboles *SMF*.

Al comparar con los árboles *SFF*, tabla 4.1, encontramos que, en los árboles *SFF* los nodos dentro de los intervalos de mayor tamaño se encuentran mayoritariamente situados en porcentajes superiores al 40 % de solapamiento mientras que en caso de los árboles *PRF* la mayoría de los nodos de esas tallas no superan el 40 % de solapamiento; para los nodos con una población muy pequeña no existe una gran diferencia entre estos dos tipos de árbol y, para los nodos con una población entre 11 y 100 prototipos, la diferencia empieza a observarse cuando aumenta la dimensión, siendo ligeramente superior el porcentaje de nodos de los árboles *PRF* que no presentan solapamiento con el nodo hermano o el solapamiento es inferior al 10 %.

Al comparar con los datos sobre solapamiento de los árboles *SMF*, tabla 4.2, observamos que salvo para los nodos más pequeños, para los que presentan ambos tipos de árbol unos valores similares, para los nodos del

		prototipos por nodo					prototipos por nodo				
		$3-10$	$11-100$	$101-1000$	$1001-10000$	$10001-100000$	$3-10$	$11-100$	$101-1000$	$1001-10000$	$10001-100000$
	% nodos respecto al total	74,83	22,84	2,1	0,21	0,02	73,35	23,92	2,39	0,29	0,05
solapamiento con el hermano (%)	*nodos sin solapamiento*	76,19	26,71	0,15	0	0	58,41	14,85	0,01	0	0
	$0-10$	0	13,09	9,44	7,47	21,45	0	9,71	3,79	4,76	3,84
	$10-20$	4,06	15,73	18,29	22,49	24,9	4,14	12,35	9,97	9,7	19,09
	$20-30$	5,85	13,58	18,33	17,36	9,68	6,96	11,13	15,49	11,87	16,53
	$30-40$	4,94	10,56	16,5	12,1	14,77	7,4	8,53	16,04	14,82	14,17
	$40-50$	1,73	7,18	14,36	9,51	4,13	2,47	6,37	12,98	17,5	11,51
	$50-60$	2,41	5,16	8,52	6,94	5,54	4,44	5,85	8,49	14,22	15,57
	$60-70$	2,8	2,87	3,94	4,92	3,91	7,66	5,08	4,53	7,89	12,06
	$70-80$	1,09	1,87	3,25	4,15	2,13	3,93	5,4	1,78	2,33	1,81
	$80-90$	0,87	1,6	2,75	4,05	4,62	4,28	7,72	1,37	0	1,71
	$90-100$	0,05	1,63	4,47	11,02	8,87	0,3	13,01	25,55	16,91	3,7
		dimensión 2					dimensión 8				

Tabla 4.3: Porcentaje de nodos, respecto al total, que pertenecen a cada uno de los intervalos de población de nodo considerados en un árbol *PRF* de 100000 prototipos. Para cada intervalo de población se muestra el porcentaje de nodos de ese intervalo que se encuentran dentro de un determinado rango de solapamiento con su nodo hermano.

resto de poblaciones consideradas, los árboles *SMF* presentan menor solapamiento en los nodos más grandes y en los nodos de población intermedia. En concreto, los nodos no solapados o con un solapamiento no superior al 10 % suponen en dimensión 2 un 63 % en el caso de los árboles *SMF* frente a sólo un 39 % en el caso de los árboles *PRF* y, en dimensión 8 un 43 % frente a un 24 %.

Como se puede observar estudiando el solapamiento entre nodos, las tres estrategias para construir el árbol que se han detallado hasta ahora agrupan los prototipos de un modo muy diferente. Se puede observar esta diferencia a través de la figura 4.11, donde se muestra la partición que se produce en un conjunto de 1000 prototipos de dimensión 2 en los primeros pasos de cada una de las técnica de construcción del árbol aquí referidas.

4.4.2. La mediana de los prototipos como representante del nodo raíz - Árbol *PMF*

Ya que al elegir un prototipo al azar como representante del nodo raíz el solapamiento es mucho mayor que el que encontramos en los árboles *SMF*, que se construyen tomando la mediana como representante de la raíz, nos planteamos seguir la estrategia de construcción del árbol *PRF* pero estableciendo como representante del nodo raíz la mediana del conjunto inicial de prototipos, en lugar de que se le asigne un prototipo al azar. El resto del proceso es exactamente igual al de los árboles *PRF*. De este modo, los dos primeros nodos hijo resultantes de la división del nodo raíz, contienen los mismos prototipos que en el caso de un árbol *SMF*, y cabe esperar que el solapamiento sea menor que si se elige el representante del nodo raíz al azar. Esta forma de construir el árbol da lugar a los árboles *PMF* [12]. Los algoritmos con los que se construyen estos árboles serán los mismos que los utilizados para los árboles *PRF*, a excepcion de la línea 2 del algoritmo 10, donde se escogerá la mediana del conjunto en lugar de un prototipo arbitrario.

Profundidad del árbol

Al partir de la mediana para construir el árbol, la diferencia de población entre los dos hijos del nodo raíz es bastante superior a la que se obtenía al tomar un prototipo al azar y, puesto que el hijo de la izquierda del nodo raíz contiene muchos más prototipos que el de un árbol *PRF*, el árbol *PMF* es más profundo. La figura 4.15 muestra la rama más profunda que se obtiene al construir un árbol *PMF* para el mismo conjunto de 1000 prototipos cuya rama más profunda, en un árbol *PRF*, se mostraba en la sección anterior. La profundidad de estos árboles aumenta de manera notable conforme aumenta la dimensión de los prototipos con los que se trabaja (figura 4.16), comportamiento ya observado en los árboles *PRF*. Este aumento de la profundidad según aumenta la dimensión de los prototipos se debe a la misma razón que motiva el aumento en éstos. Al aumentar la dimensión, la distancia del prototipo más alejado del representante del nodo raíz es cada vez mayor provocando un aumento del desequilibrio en el reparto de prototipos del nodo raíz, y con ello un aumento de la profundidad.

[12] de las siglas en inglés de Parent Median Far, es decir, se mantiene el representante del padre, la mediana del conjunto, y como representante del hermano se elige el prototipo más alejado.

Figura 4.15: Rama más profunda de la construcción del árbol *PMF* para 1000 prototipos de una distribución uniforme en dimensión 2.

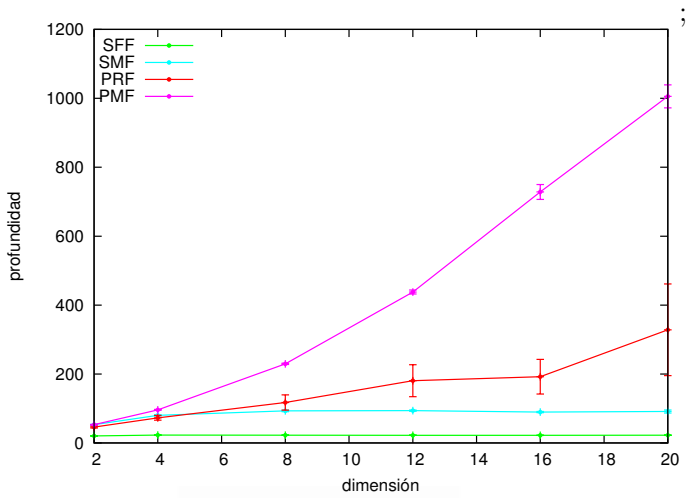


Figura 4.16: Profundidad de los árboles *PMF*, PRF, *SFF* y *SMF* para una población del nodo raíz de 100000 prototipos.

Solapamiento entre nodos

Al comparar los datos de solapamiento para los árboles *PMF* que se muestran en la tabla 4.4 con los obtenidos para los árboles *SMF*, tabla 4.2, se observa principalmente el aumento notable de nodos con una población baja, que constituyen más del 80 % de los nodos del árbol. Además, en este intervalo de población tan numeroso, el porcentaje de nodos sin solapamiento ha aumentado respecto a los valores que se observan en los árboles *SMF*.

En los nodos con una población entre 11 y 100 es donde los árboles *SMF* presentan un solapamiento inferior a los árboles *PMF*, si bien en estos últimos el número de estos nodos se ha reducido a la mitad. Para los intervalos de población superiores[13], aunque representan un porcentaje muy pequeño respecto al total, el 75 % de los nodos para dimensión 2 y el 95 % de los nodos para dimensión 8 presentan un solapamiento con el nodo hermano como mucho del 20 %. En líneas generales podemos decir que los árboles *PMF* son los que presentan el menor solapamiento.

[13] Aunque el intervalo de error para el porcentaje de nodos con poblaciones altas se acerca al 10 % de la medida, los datos que representan el porcentaje de nodos sin solapamiento o con poco solapamiento tienen un intervalo de error que no supera el 2 % de la medida.

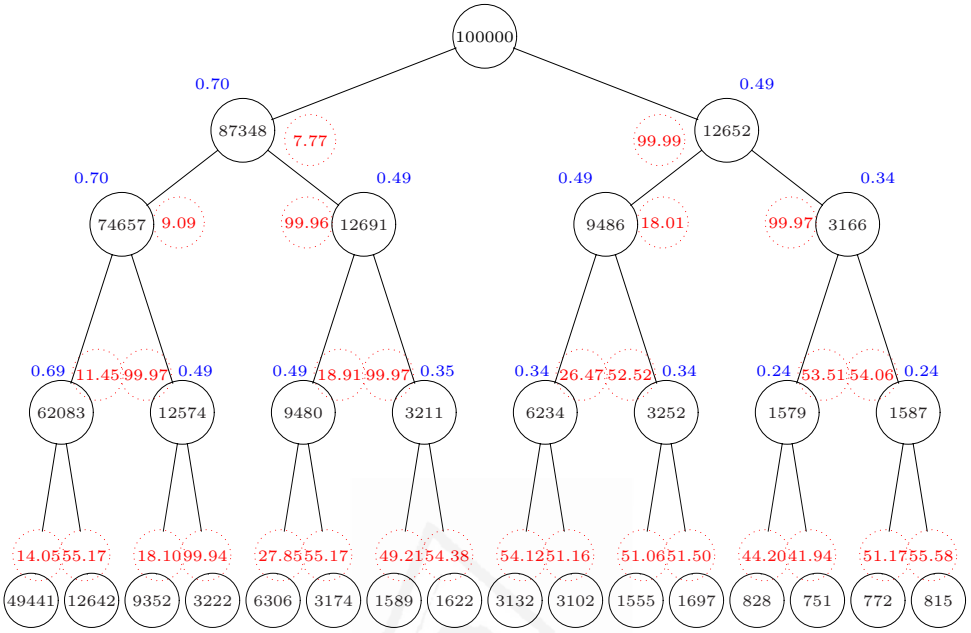


Figura 4.17: Parte superior de un árbol *PMF* para 100000 prototipos extraídos de una distribución uniforme en el hipercubo unidad en dimensión 2 usando la distancia euclídea.

4.4.3. *PMF* recalculando el representante de la derecha - Árbol *PMM*

Ya hemos visto que al construir los árboles *PMF* se obtiene, en los dos primeros nodos hermanos, dos nodos de tamaño muy diferente, en el nodo hijo de la izquierda se encuentran la mayoría de los prototipos del nodo raíz mientras que en el nodo de la derecha encontramos un número reducido de los mismos. Sin embargo, este número reducido de prototipos puede constituir una cantidad considerable cuando trabajamos con nodos raíz que contienen un número de prototipos muy alto. En estos casos, si aplicamos las reglas de construcción del árbol *PMF*, el nodo de la derecha se irá dividiendo basándose en los dos prototipos más alejados del nodo. Se propone una pequeña variación al árbol *PMF* que consiste en, una vez asignados los representantes del nodo de la derecha según la técnica *PMF* calcular, entre estos componentes del nodo, la mediana del mismo y asignarlo como representante del nodo de la derecha. Esto da lugar a los árboles PMM[14].

[14]de las siglas en inglés de Parent Median Median, es decir, se mantiene el representante del padre (la mediana) en uno de los hijos, y se elige como representante del otro hijo la mediana de su conjunto de prototipos.

		prototipos por nodo					prototipos por nodo				
		$3-10$	$11-100$	$101-1000$	$1001-10000$	$10001-100000$	$3-10$	$11-100$	$101-1000$	$1001-10000$	$10001-100000$
	% nodos respecto al total	84	14,52	1,34	0,12	0,02	82,3	15,73	1,67	0,21	0,08
solapamiento con el hermano (%)	*nodos sin solapamiento*	84,77	24,9	0,04	0	0	68,7	16,66	0,12	0	0
	$0-10$	0	11,51	3,09	6,02	41,76	0	11,3	6,95	19,37	34,66
	$10-20$	2,38	16,2	9,23	29,29	34,71	2,27	13,64	18,35	18,5	61,46
	$20-30$	3,4	15,84	12,09	16,29	0	3,76	11,17	23,89	19,78	3,76
	$30-40$	2,72	13,11	19,28	0,24	0	4,1	7,69	16,87	16,32	0,12
	$40-50$	0,96	9,19	31,32	5,54	0	1,27	4,98	8,5	7,48	0
	$50-60$	3,81	5,35	18,69	28,97	5,88	9,56	4,2	3,13	1,14	0
	$60-70$	1,27	1,75	0,88	0	0	4,69	3,45	0,8	0	0
	$70-80$	0,42	0,62	0	0	0	2,46	4,02	0,18	0	0
	$80-90$	0,26	0,49	0,01	0	0	2,95	7,04	0,21	0	0
	$90-100$	0,01	1,04	5,38	13,64	17,65	0,24	15,86	21	17,41	0
		dimensión 2					dimensión 8				

Tabla 4.4: Porcentaje de nodos, respecto al total, que pertenecen a cada uno de los intervalos de población de nodo considerados en un árbol *PMF* de 100000 prototipos. Para cada intervalo de población se muestra el porcentaje de nodos de ese intervalo que se encuentran dentro de un determinado rango de solapamiento con su nodo hermano.

Inicialmente se construye un nodo raíz, donde están todos los prototipos. En este conjunto se selecciona la mediana y el prototipo más alejado a éste (líneas 2 y 3 del algoritmo 12). A continuación se distribuyen los prototipos en base a su distancia a los dos prototipos seleccionados y se recalcula el representante del conjunto basado en el prototipo más alejado de la mediana. Con esta información se construyen dos nodos del árbol a partir de los cuales, y de manera recursiva, se continúa creando el árbol *PMM*, tal y como se recoge en el algoritmo 13.

Dado un nodo t que contiene un conjunto de prototipos, S_t, del que se conoce su representante, c_t, la construcción del árbol que parte del nodo t se realiza del siguiente modo (algoritmo 13):

- se asigna el representante, pasado como parámetro (línea 2)

$$c_t = rep;$$

- se calcula el radio del nodo t (línea 4);

- se seleccionan los representantes de los dos hijos, el de la izquierda es el mismo que el del padre, y el de la derecha el más alejado de éste (líneas 5 y 6)

$$c_{\mathrm{d}} = \arg\max_{s \in S_t} d(c_t, s);$$

- se crean dos conjuntos S_{i} y S_{d} donde se distribuyen los prototipos en función de su distancia a c_{i} y c_{d} (líneas 7 y 8)

- calculamos la mediana de los prototipos en S_{d} y se asigna como representante del hijo de la derecha

$$c_{\mathrm{d}} = \{p | p \in S_{\mathrm{d}} \wedge p = \arg\min_{s \in S_{\mathrm{d}}} \sum d(p, s)\};$$

- se crean dos nodos hi_t y hd_t que tienen como representantes c_{i} y c_{d} y como conjunto de prototipos S_{i} y S_{d};

- para cada nodo se repiten recursivamente los pasos anteriores hasta que en cada nodo sólo quede un prototipo, su representante.

Algoritmo 12 `CreaRaiz_PMM(S)`

Entrada: S (conjunto de prototipos no vacío);
Salida: t (árbol que representa a S);

1: t = `CreaNodo`();
2: c_{i}= `Mediana`(S_t);
3: $c_{\mathrm{d}} = \arg\max_{(p \in S)} d(p, c_i)$;
4: $S_{\mathrm{i}} = S_{\mathrm{d}} = \{\}$;
5: $S_{\mathrm{i}} = \{p \in S | d(p, c_{\mathrm{i}}) < d(p, c_{\mathrm{d}})\}$;
6: $S_{\mathrm{d}} = S - S_{\mathrm{i}}$;
7: c_{d}= `Mediana`(S_{d});
8: hi_t=`CreaArbol_PMM`($S_{\mathrm{i}}, c_{\mathrm{i}}$);
9: hd_t=`CreaArbol_PMM`($S_{\mathrm{d}}, c_{\mathrm{d}}$);
10: **devolver** t;

Algoritmo 13 `CreaArbol_PMM`(S, rep)

Entrada: S (conjunto de prototipos no vacío);
$rep \in S$;
Salida: t (árbol que representa a S);

```
1: t = CreaNodo();
2: c_t = rep;
3: si |S| > 1 entonces
4:    r_t = máx_{p∈S} d(c_t, p); // radio del nodo
5:    c_i = c_t; // mismo representante que el padre
6:    c_d = arg máx_{(p∈S)} d(p, c_i);
7:    S_i = {p ∈ S | d(p, c_i) < d(p, c_d)};
8:    S_d = S − S_i;
9:    c_d = Mediana(S_d);
10:   hi_t = CreaArbol_PMM(S_i, c_i);
11:   hd_t = CreaArbol_PMM(S_d, c_d);
12: fin si
13: devolver t;
```

Profundidad del árbol

La profundidad de estos árboles es la misma que la de los árboles *PMF*, como se puede observar en la figura 4.18. Ya que los cambios en la elección del representante se producen en la rama más corta que parte del nodo raíz, no afectan a la profundidad del árbol.

4.5. Experimentos con datos sintéticos

Llevar a cabo la comparación entre los métodos propuestos y otros métodos es una tarea difícil, ya que cada autor presenta, de un modo diferente, los resultados asociados al funcionamiento de su algoritmo. Hay que tener en cuenta aspectos diferentes al estudiar cada método: coste asociado a la fase de preproceso, posibilidad de adaptación de las estructuras utilizadas a nuevos datos, degeneración del método con la dimensionalidad, o si existe algún tipo de cuello de botella, bien vinculado al coste espacial o temporal.

Para valorar la incidencia de las distintas técnicas de construcción del árbol en la búsqueda del vecino más cercano se ha realizado una serie de experimentos con datos artificiales generados a partir de distribuciones uniformes en el hipercubo unidad, para diferentes dimensiones.

La medida de disimilitud aplicada entre dos prototipos es la distancia euclídea.

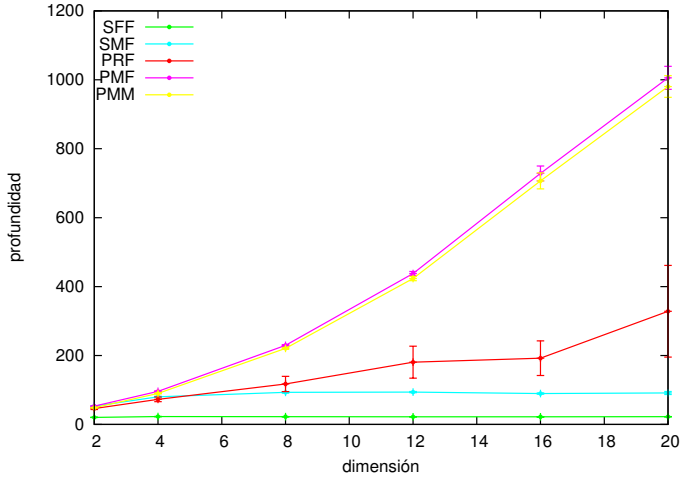


Figura 4.18: Profundidad de los árboles *SMF*, *PRF*, *PMF* e *PMM* en función de la dimensión para una población del nodo raíz de 100000 prototipos.

Al realizar la exploración del árbol en busca del prototipo más cercano a una muestra dada se ha utilizado en todos los árboles la regla de eliminación de Fukunaga y Narendra FNR (introducida en el capítulo anterior) para evitar la exploración de algunas ramas del árbol.

El tamaño del conjunto de prototipos en el que se busca el vecino más cercano varía desde 10000 hasta 110000. En cada experimento se ha calculado la media de los valores del número de distancias calculadas cuando se ha realizado una búsqueda con 1000 muestras. Cada punto en las gráficas presentadas es, a su vez, la media de 10 experimentos con distintos conjuntos de prototipos y muestras.

En las gráficas, además de mostrar los resultados obtenidos con los árboles que hemos propuesto (resumidos en la tabla 4.5), se muestran también los resultados obtenidos empleando la técnica Spatial Approximation Tree (SAT) (sólo para dimensiones altas), la técnica Vantage Point Tree (VPT) y la técnica *kd*-tree (kdT). Se han representado dos conceptos: el *número medio de distancias* a calcular hasta encontrar el vecino más cercano, y el *número medio de nodos visitados* en esta búsqueda. Los valores obtenidos para estos dos conceptos pueden coincidir, o no, según la técnica empleada.

Las gráficas de la figura 4.19 y la figura 4.20 comparan los nodos visitados y el número medio de distancias calculadas de todos los árboles propuestos.

Para las gráficas de los experimentos con prototipos de dimensión 2 y 4 se han omitido los datos referentes a la técnica SAT, ya que tanto el número de nodos visitados como el número medio de distancias calculadas presentaban

Método	Representantes de los hijos del nodo raíz	Representantes de los hijos de un nodo no raíz
SMM	Los representantes son las 2 medianas obtenidas en 2_medoids.	Igual que en la raíz.
SFF	Los representantes son los 2 prototipos más alejados.	Igual que en la raíz.
SMF	Los representantes son la mediana de todo el conjunto y el prototipo más alejado a la misma.	Igual que en la raíz.
PRF	Los representantes son un prototipo al azar y el más alejado de éste.	En el nodo de la izquierda se mantiene el mismo representante que el del padre, y en el de la derecha el más alejado de éste.
PMF	Los representantes son la mediana de todo el conjunto y el prototipo más alejado de ella.	En el nodo de la izquierda se mantiene el mismo representante que el del padre, y en el de la derecha el más alejado de éste.
PMM	Los representantes son la mediana de todo el conjunto y el prototipo más alejado de ella.	En el nodo de la izquierda se mantiene el mismo representante que el del padre, y en el de la derecha, tras haber asignado al nodo los prototipos que se encuentran más cercanos al prototipo más alejado a la mediana que a la mediana, se calcula la mediana de los prototipos asignados a este nodo y se toma como representante del nodo.

Tabla 4.5: Resumen de las técnicas propuestas para la construcción del árbol.

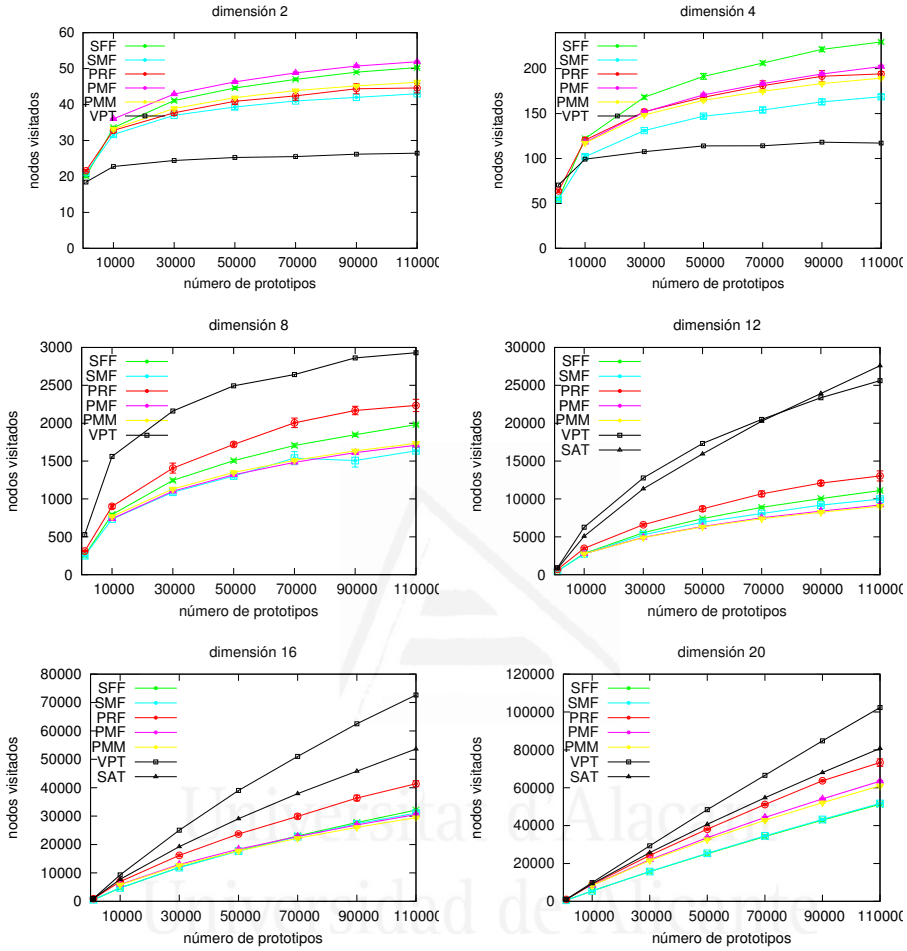


Figura 4.19: Número de nodos visitados en la fase de búsqueda, en función del tamaño de la base de datos, según la técnica de construcción de árbol empleada.

valores muy superiores al resto de técnicas (por ejemplo, para dimensión 2 variaban desde 112 para 1000 prototipos hasta 4470 para 110000) y la inclusión de estos datos impedía distinguir bien los resultados de las otras técnicas.

Los experimentos muestran que, para las dimensiones más bajas, 2, 4 y 8, la técnica *kd*-tree calcula menos distancias que el resto de técnicas, sin embargo conforme aumenta la dimensión también aumenta drásticamente el número de distancias que calcula, pasando a ser la técnica que calcula más distancias. También la técnica VPT para dimensiones bajas visita un menor

número de nodos y calcula menos distancias que las técnicas propuestas (figuras 4.19 y figura 4.20). Sin embargo, conforme aumenta la dimensión el buen comportamiento de esta técnica para dimensiones bajas desaparece al igual que ocurre con la técnica *kd*-tree. En estos casos la técnica que presenta el menor número de nodos visitados es la del árbol *SMF* que, como ya se ha comentado, presenta un menor solapamiento entre nodos hermanos que los árboles *SFF* y *PRF*, y ligeramente inferior a *PMF*.

Si nos fijamos en las distancias calculadas, la técnica *PRF* mejora a la *SMF* al disminuir notablemente el número de distancias calculadas por el ahorro que supone que el representante del hijo de la izquierda sea el mismo que el representante del padre y, por lo tanto, sea una distancia menos a calcular cada vez que desde un nodo se explora el siguiente nivel del árbol (figura 4.20).

Las técnicas *PMF* y PMM se basan en los árboles *PRF* pero con una inicialización que se basa en la mediana, y esta combinación resulta en una reducción del número de nodos visitados así como del número de distancias calculadas respecto a los árboles *PRF*.

En general, las técnicas que presenta el mejor comportamiento tanto en el número de nodos visitados como de distancias calculadas son las técnica *PMF* y *PMM*, como se puede observar en la figura 4.21 para un conjunto de 110000 prototipos en distintas dimensiones.

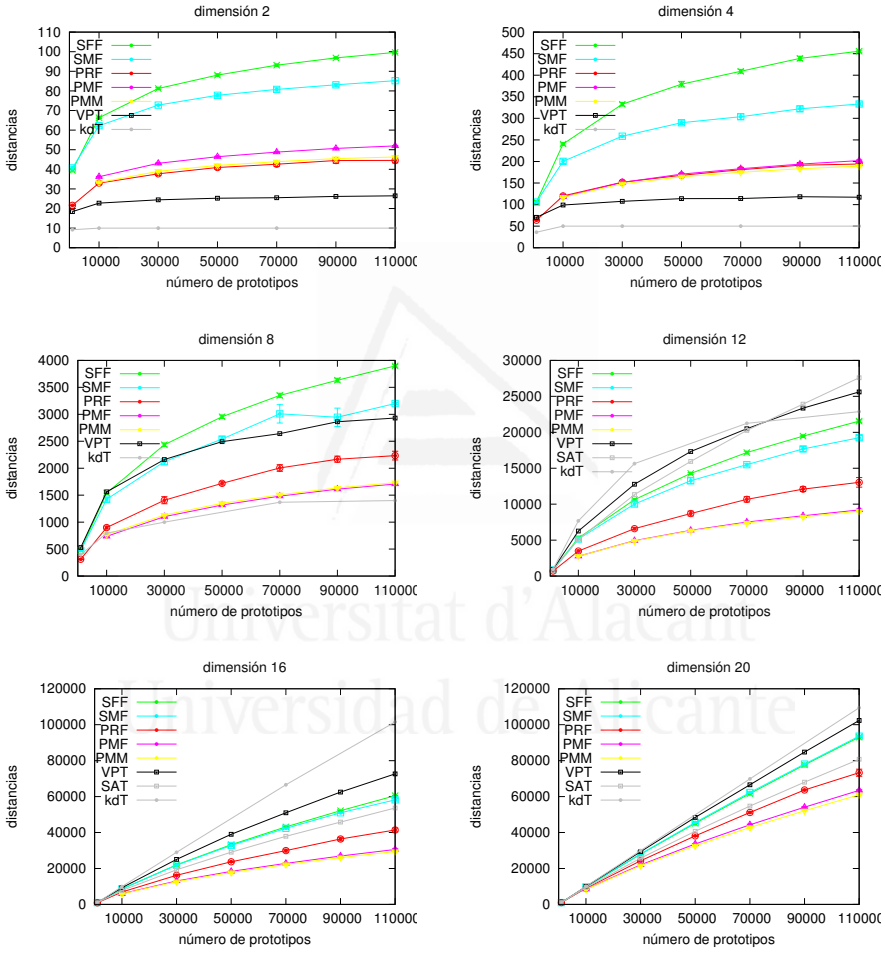


Figura 4.20: Número de distancias calculadas en la fase de búsqueda, en función del tamaño de la base de datos, según la técnica de construcción de árbol empleada.

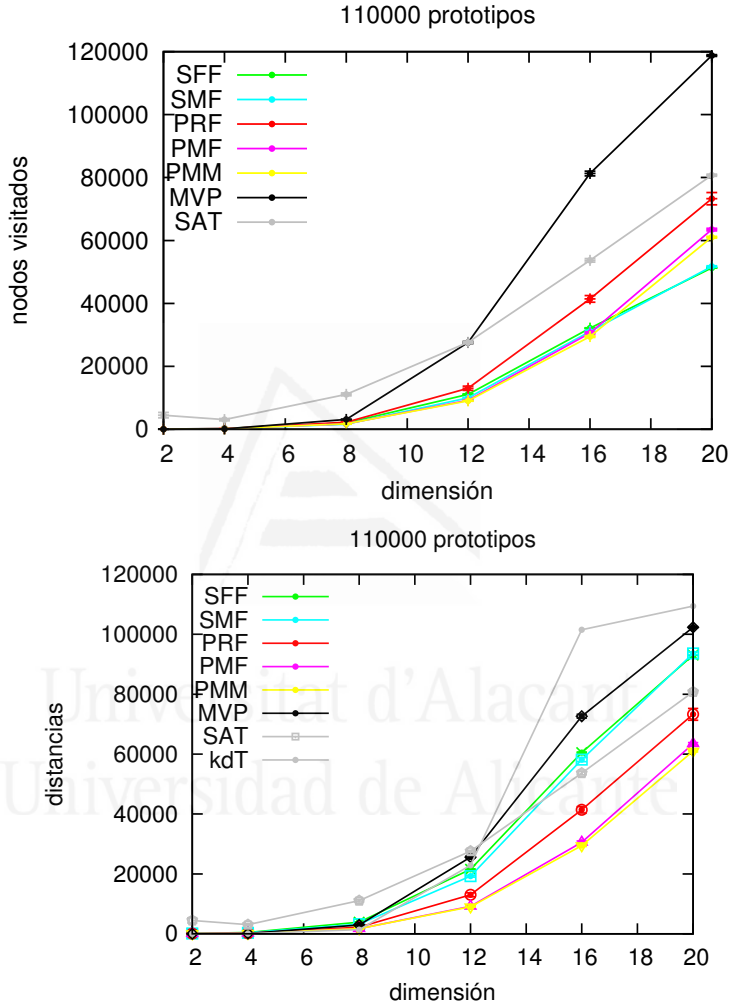


Figura 4.21: Número medio de nodos visitados y de distancias calculadas en la fase de búsqueda en función de la dimensión de los prototipos, según la técnica de construcción de árbol empleada.

Capítulo 5

Reglas de eliminación

Una vez construido el árbol con los prototipos que constituyen la base de datos con la que se va a trabajar, ante una consulta de búsqueda del vecino más cercano a una muestra dada, se debe evitar la exploración de algunos subárboles (ramas) del árbol en aras de reducir el tiempo de respuesta.

Para evitar la exploración completa del árbol, se suelen emplear reglas de eliminación que se apoyan en la propiedad de la desigualdad triangular (Heath and Heath, 1956). La evaluación de una regla de eliminación se lleva a cabo teniendo en cuenta características precalculadas de una rama del árbol, y tiene como objetivo determinar si es posible o no encontrar, en esa rama, un prototipo más cercano a la muestra que el que se haya localizado hasta el momento. Una rama está representada por un nodo en el que se consideran todos los prototipos que podemos encontrar en sus nodos descendientes. Cuando se cumple la regla sobre un nodo, tenemos la certeza de que en ese nodo no vamos a encontrar ningún prototipo más cercano a la muestra que el que hayamos encontrado hasta ese momento, y entonces se dice que se *poda* el nodo, lo que es equivalente a dejar de explorar la rama correspondiente a ese nodo. Por el contrario, si no se cumple la regla, no podemos asegurar que no vayamos a encontrar en esa rama un prototipo más cercano a la muestra y, por lo tanto, se pasa a realizar la búsqueda en los nodos hijo.

Ya que tanto en los experimentos a los que se ha hecho referencia en el capítulo anterior como en los experimentos realizados en este capítulo se toma como referencia la regla de Fukunaga y Narendra (FNR), en primer lugar se va a definir formalmente en qué consiste esta regla, y posteriormente se formularán las reglas que se proponen en este trabajo.

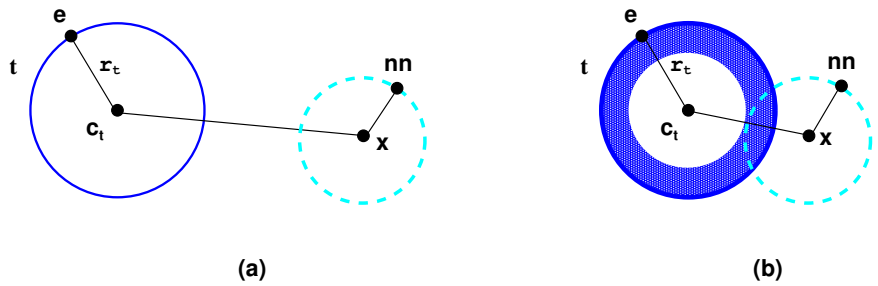


Figura 5.1: Regla de eliminación de Fukunaga y Narendra (FNR).

5.1. La regla de eliminación de Fukunaga y Narendra (FNR)

La regla de Fukunaga y Narendra (FNR) se basa en el siguiente lema:

Lema 1. *Sea (M, d) un espacio métrico, sea $S \subset M$ un subconjunto finito de puntos, sea $l \in M$, y sea $e = \arg\max_{s \in S} d(s, l)$ entonces*

$\forall x, n \in M$*:*

$$d(e,l) \leq d(x,l) - d(x,n) \Rightarrow \forall s \in S \; d(x,s) \geq d(x,n).$$

Demostración. Sea $s \in S$ y $x, n \in M$,

$$\begin{aligned} d(x,s) &\geq d(x,l) - d(s,l) && \text{por la desigualdad triangular} \\ &\geq d(x,l) - d(e,l) && \text{como } s \in S \Rightarrow d(s,l) \leq d(e,l) \\ &\geq d(x,l) - d(x,l) + d(x,n) && \text{por hipótesis} \\ &= d(x,n) \end{aligned}$$

□

Regla de eliminación de Fukunaga y Narendra (FNR).
Sea x una muestra, sea nn el actual vecino más cercano a x, sea t un nodo de un árbol, y sea r_t el radio del nodo (fig. 5.1). Entonces,

- si t *no* es un *nodo hoja*, si

$$r_t \leq d(x, c_t) - d(x, nn) \quad \text{fig. 5.1(a)}$$

por el lema 1, ningún prototipo en S_t puede estar más cercano a la muestra que nn, y por lo tanto la exploración del nodo se puede evitar. Sin embargo, si

$$r_t > d(x, c_t) - d(x, nn) \quad \text{fig. 5.1(b)}$$

puede haber prototipos en el nodo t más cercanos a la muestra x que nn, por lo que no podemos dejar de explorar el nodo.

- si t es un *nodo hoja*, Fukunaga y Narendra proponen una particularización de esta regla para aplicar a los prototipos del nodo hoja (si hay más de uno). En nuestro caso, al trabajar con árboles que contienen un sólo prototipo en sus nodos hoja, esta regla no se aplica.

De ahora en adelante, nos referiremos a la regla de eliminación de Fukunaga y Narendra como la regla de eliminación FNR.

Obsérvese que ya que r_t no depende de la muestra, ya que es el radio del nodo que se calcula durante el proceso de construcción del árbol. También se debe tener en cuenta que cuando se visita el nodo t, las distancias $d(x, nn)$ y $d(x, c_t)$ ya son conocidas, y por lo tanto la regla FNR se puede calcular sin consultar ninguna otra distancia adicional. En las gráficas de los experimentos que se muestran al final de este capítulo nos referimos a esta regla con la letra "f".

5.2. Regla de eliminación basada en el nodo hermano (SBR)

Con esta regla pretendemos evitar la exploración de un nodo en base a la distancia que existe entre este nodo y el representante de su nodo hermano. Para poder aplicar esta regla es necesario conocer, dado un nodo, la distancia entre el representante del nodo y el prototipo más cercano en el nodo hermano. Esta distancia no depende de la búsqueda, sino del propio

árbol y, por lo tanto, es un valor que se calcula y almacena en preproceso. Si esta distancia es lo suficientemente grande como para cumplir la desigualdad propuesta, se puede evitar la exploración del nodo hermano, ya que se tiene la seguridad de que ningún prototipo en él estará más cercano a la muestra que el prototipo más cercano hasta el momento.

Esta regla se basa en el siguiente lema:

Lema 2. *Sea (M, d) un espacio métrico, sea $S \subset M$ un subconjunto finito, sea $l \in M$, y $e = \arg\min_{s \in S} d(s, l)$. Entonces $\forall x, n \in M$:*

$$d(e, l) \geq d(x, r) + d(x, n) \Rightarrow \forall s \in S \; d(s, x) \geq d(x, n).$$

Demostración. Dados $s \in S$ y $x \in M$,

$$\begin{aligned} d(s, x) &\geq d(s, l) - d(x, l) && \text{por la desigualdad triangular} \\ &\geq d(e, l) - d(x, l) && \text{como } s \in S \Rightarrow d(s, l) \leq d(e, l) \\ &\geq d(x, l) + d(x, n) - d(x, l) && \text{por hipótesis} \\ &= d(x, n). \end{aligned}$$

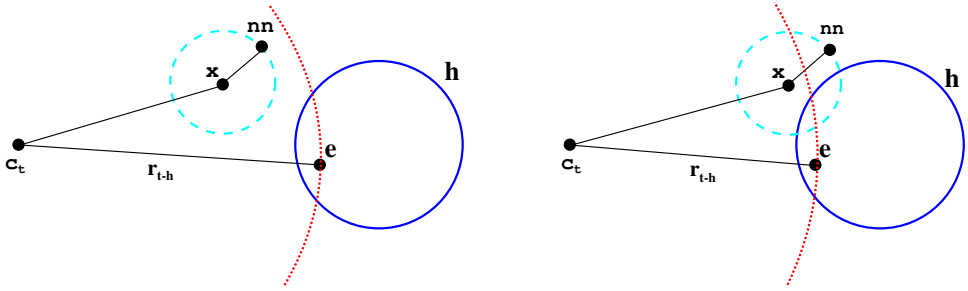
□

Regla de eliminación basada en el nodo hermano (SBR). Sea x una muestra, nn el prototipo más cercano hasta el momento, t un nodo del árbol, S_t el conjunto de prototipos del nodo t, c_t su prototipo representante, h su nodo hermano, S_h el conjunto de prototipos del nodo h, y r_{th} la distancia entre el representante de t y el prototipo de h más cercano a t, $r_{th} = \min_{s \in S_h} d(s, c_t)$ (fig. 5.2). Entonces si

$$r_{th} \geq d(x, c_t) + d(x, nn)$$

por el lema 2, ningún prototipo en S_h puede estar más cercano a la muestra que el vecino más cercano hasta el momento y por lo tanto se puede evitar la exploración del nodo h.

Ya que la distancia r_{th} no depende de la muestra, se puede calcular y almacenar en el tiempo de preproceso. Además, cuando se está examinando el nodo h, la distancia $d(x, nn)$ ya está calculada, por lo que la regla del nodo



(a) El nodo h se poda **(b) El nodo h no se poda**

Figura 5.2: Regla basada en el nodo hermano (SBR).

	Dimensión 2		Dimensión 8	
Num prototipos	FNR	SBR	FNR	SBR
10000	32.9	1133.6	900.5	9570.6
50000	40.9	5271.5	1719.1	46803.0
90000	44.4	8780.4	2167.6	82958.9
110000	44.5	10859.1	2234.3	101582

Tabla 5.1: Distancias calculadas empleando la regla FNR o la regla SBR para el árbol PRF con distintas dimensiones y número de prototipos, al realizar la búsqueda del vecino más cercano.

hermano puede ser evaluada sin calcular antes la distancia $d(x, c_t)$, como sí ocurre en la regla FNR.

En las gráficas de experimentos que se muestran al final de este capítulo nos referimos a esta regla con la letra "h".

Se realizaron unos experimentos preliminares utilizando el árbol PRF y sólo la regla SBR para valorar la eficacia de esta regla. Los resultados obtenidos muestran que la regla SBR no tiene mucha capacidad de poda al realizar la búsqueda y evita calcular muy pocas distancias con respecto al empleo de la fuerza bruta. Este comportamiento se debe a que la distancia entre los nodos del árbol no es muy grande, por lo que el valor de r_{th} es pequeño. En la tabla 5.1, se muestra el número de distancias calculadas para distintas dimensiones de los prototipos con la regla SBR y se compara con los de la regla FNR.

5.3. Regla de eliminación generalizada (GR)

Esta regla intenta evitar la exploración de un nodo empleando de manera iterativa una combinación de la regla de Fukunaga y Narendra (FNR) y la regla del nodo hermano (SBR)[1]. Cada vez que se comprueba si el nodo se puede eliminar o no con esta regla, el nodo se considera dividido en dos zonas y, de un modo similar a que se estuviesen considerando dos nodos, se valora para una zona la regla FNR y para la otra la SBR. Esta combinación de reglas se basa en el siguiente lema:

Lema 3. *Sea (M, d) un espacio métrico, $S \subset M$ un subconjunto finito de M, $l, r \in M$, $e_i \in S$, y $G_i \subset S$ de modo que:*

$$\begin{aligned} G_1 &= S \\ e_i &= \arg\max_{g \in G_i} d(g, l) \\ G_{i+1} &= \{g \in G_i : d(g, r) < d(e_i, r)\} \end{aligned}$$

entonces $\forall x, n \in M$:

$$\begin{aligned} &\exists i : d(r, e_i) \geq d(r, x) + d(x, n) \wedge \\ &\quad d(l, e_{i+1}) \geq d(l, x) - d(x, n) \Rightarrow \forall s \in S \quad d(s, x) \geq d(x, n). \end{aligned}$$

Demostración. Sea $s \in S$,

1. Si $s \in G_{i+1}$, sea $e_{i+1} = \arg\max_{g \in G_{i+1}} d(g, l)$. Sea $x \in M$,

$$\begin{aligned} d(x, s) &\geq d(x, l) - d(s, l) && \text{por la desigualdad triangular} \\ &\geq d(x, l) - d(e_{i+1}, l) && \text{como } s \in S \Rightarrow d(s, l) \leq d(e_{i+1}, l) \\ &\geq d(x, l) - d(x, l) + d(x, n) && \text{por hipótesis} \\ &= d(x, n). \end{aligned}$$

2. Si $s \notin G_{i+1}$, entonces como $G_{i+1} = \{g \in G_i : d(g, r) < d(e_i, r)\}$ se obtiene que $d(s, r) \leq d(e_i, r)$. Sea $x \in M$,

$$\begin{aligned} d(s, x) &\geq d(s, r) - d(x, r) && \text{por la desigualdad triangular} \\ &\geq d(e_i, r) - d(x, r) && \text{como } s \notin G_{i+1} \Rightarrow d(s, r) \leq d(e_i, r) \\ &\geq d(x, r) + d(x, n) - d(x, r) && \text{por hipótesis} \\ &= d(x, n). \end{aligned}$$

[1]Publicación derivada de este estudio (Gómez-Ballester et al., 2006)

□

Considerando este lema y siendo x la muestra, nn el prototipo más cercano hasta el momento, t un nodo del árbol, h el nodo hermano de t, S_t y S_h el conjunto de prototipos de los nodos t y h respectivamente, y c_t y c_h sus representantes, $e_i \in S_t$, y $G_i \subset S_t$ de modo que:

$$\begin{aligned} G_1 &= S_t \\ e_i &= \arg\max_{g \in G_i} d(g, c_t) \\ G_{i+1} &= \{g \in G_i : d(g, c_h)) < d(e_i, c_h)\} \end{aligned}$$

y sea m el número natural más pequeño de manera que $|G_m| = 0$.

Regla de eliminación generalizada (GR).
Sea x una muestra, nn el prototipo más cercano hasta el momento, t un nodo del árbol, S_t el conjunto de prototipos del nodo t, c_t su prototipo representante, h su nodo hermano, S_h el conjunto de prototipos del nodo h, y c_h el representante del nodo h. Si existe i que cumpla:

$$d(c_h, e_i) \geq d(c_h, x) + d(x, nn) \wedge \tag{5.1}$$

$$d(c_t, e_{i+1}) \leq d(c_t, x) - d(x, nn) \tag{5.2}$$

ningún prototipo en S_t puede estar más cercano a la muestra que el vecino más cercano actual, y por lo tanto la exploración del nodo t se puede evitar.

Se debe observar que el cálculo de las distancias $d(c_h, e_i)$ y $d(c_t, e_{i+1})$ no depende de la muestra, estas distancias se pueden calcular y almacenar en tiempo de preproceso.

En la figura 5.3 se puede ver un ejemplo en el que la regla generalizada poda el nodo t para $i = 1$. Se han marcado todos los prototipos e_i del nodo t aunque, en este caso, e_3 no se considera en las condiciones de eliminación, ya que el valor $i = 1$ hace ciertas las ecuaciones 5.1 y 5.2 por lo que sólo se tiene en cuenta en su evaluación los prototipos e_1 y e_2. Sin embargo, en la figura 5.4 se revisan las ecuaciones para todos los posibles e_i (e_1, e_2, e_3) y no es posible podar el nodo ya que no se cumple en ningún caso la ecuación 5.2.

Se debe considerar también que, cuando se revisa el nodo t para ver si se puede podar, las distancias $d(x, nn)$, $d(c_t, x)$ y $d(c_h, x)$ ya están calculadas.

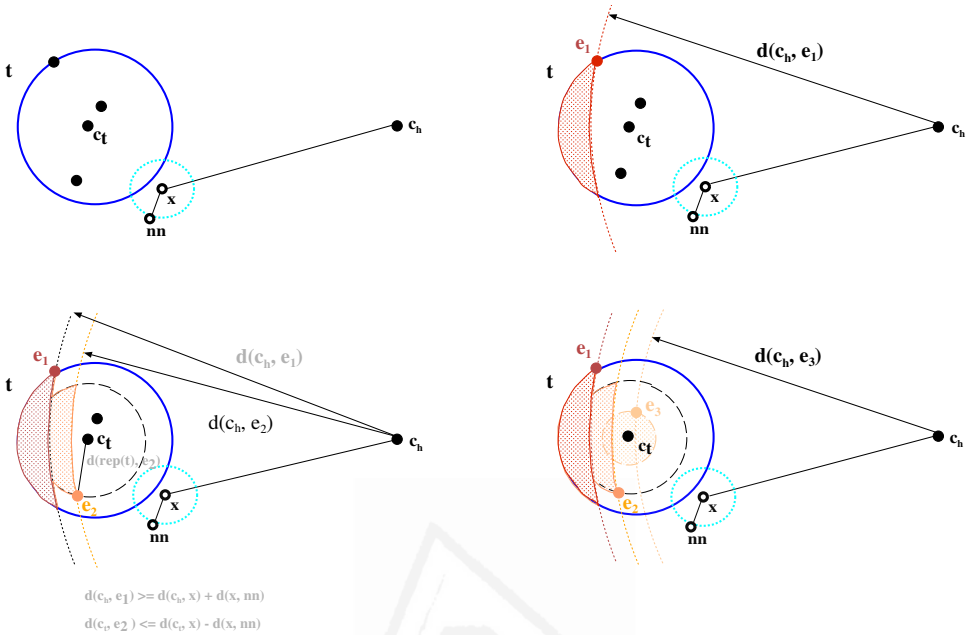


Figura 5.3: Aplicación de la regla de eliminación generalizada. Para $i = 2$ se cumplen las 2 ecuaciones y, por lo tanto, la regla poda el nodo.

Entonces, para cada i, la regla se puede evaluar sin calcular nuevas distancias. Para averiguar si existe un valor de i que cumpla las condiciones, se puede utilizar una búsqueda binaria.

Las reglas FNR y SBR son casos especiales de GR en las siguientes situaciones:

- si $i = 0$ y se considera sólo la ecuación 2, entonces obtenemos la regla FNR;
- si $i = m - 1$ y se considera sólo la ecuación 1, entonces obtenemos la regla SBR.

En la (fig. 5.5) se puede observar como, en este ejemplo, el valor de m, el menor número natural para el que $|G_m| = 0$, es 3, ya que no existe ningún prototipo en S_t con menor distancia a c_h que $d(c_h, e_2)$. Si se considera entonces $i = m - 1$ y sólamente la ecuación 1, sería como considerar sólo la regla SBR.

En las gráficas de los experimentos que se muestran en apartados posteriores, esta regla se nombra con la letra "g".

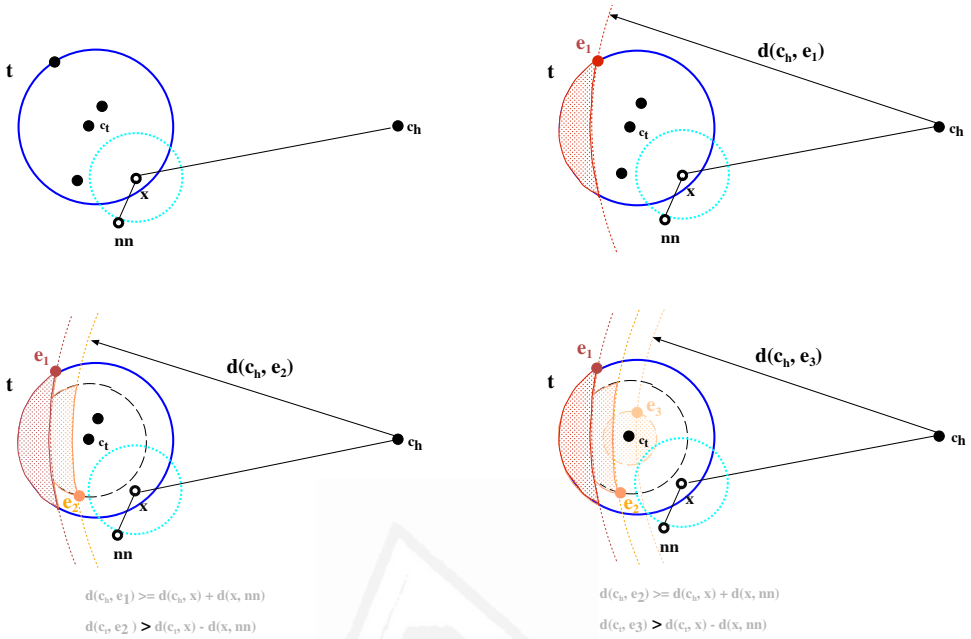


Figura 5.4: El nodo t no puede ser podado usando la regla de eliminación generalizada (GR)

5.4. Regla de eliminación de la tabla (TR)

Basándonos en el algoritmo AESA, que ha sido durante años un algoritmo imbatido en cuanto al número de distancias calculadas para encontrar el vecino más cercano, y en el uso que hace de distancias precalculadas entre prototipos de la base de datos, surgió la nueva regla propuesta [2].

Para esta nueva regla se define una tabla donde almacenamos la distancia entre cada prototipo y cada nodo que resulta de la construcción del árbol. Por lo tanto, para aplicar esta regla, necesitamos definir la distancia de un prototipo p a un nodo del árbol, t, es decir, al conjunto de prototipos S_t que constituyen el nodo:

$$d(p, S) = \min_{y \in S} d(p, y).$$

[2]Publicación derivada de este estudio (Oncina et al., 2007)

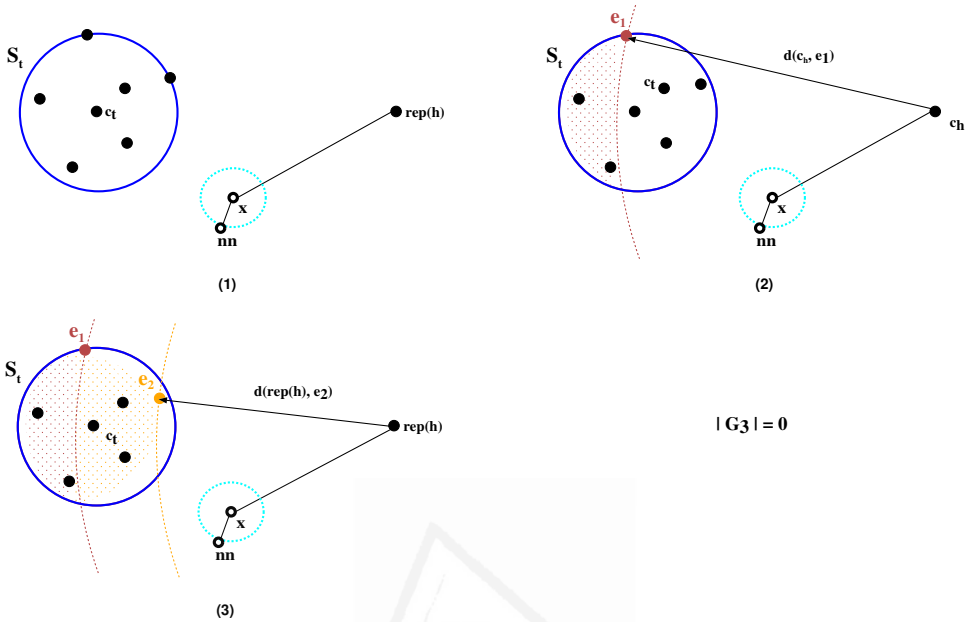


Figura 5.5: Aplicación de la regla de eliminación generalizada para $m = 3$, $i = 2$. Se considera sólo la regla del nodo hermano (SBR).

Regla de eliminación de la tabla (TR).
Sea x una muestra, nn el prototipo más cercano hasta el momento, t un nodo del árbol y S_t su conjunto de prototipos. Ningún prototipo p en S_t puede ser el vecino más cercano a la muestra x si

$$2d(x, nn) < d(nn, S_t).$$

La demostración de esta regla se enuncia en la siguiente proposición.

Proposición 1. *Si se cumple la regla de la tabla* $(2d(x, nn) < d(nn, S_t))$, *ningún prototipo* p *del nodo* t *puede estar más cerca a la muestra* x *que* nn, *es decir*

$$\forall p \in S_t, \quad d(x, p) \geq d(x, nn)$$

y por lo tanto se puede evitar la exploración del nodo t.

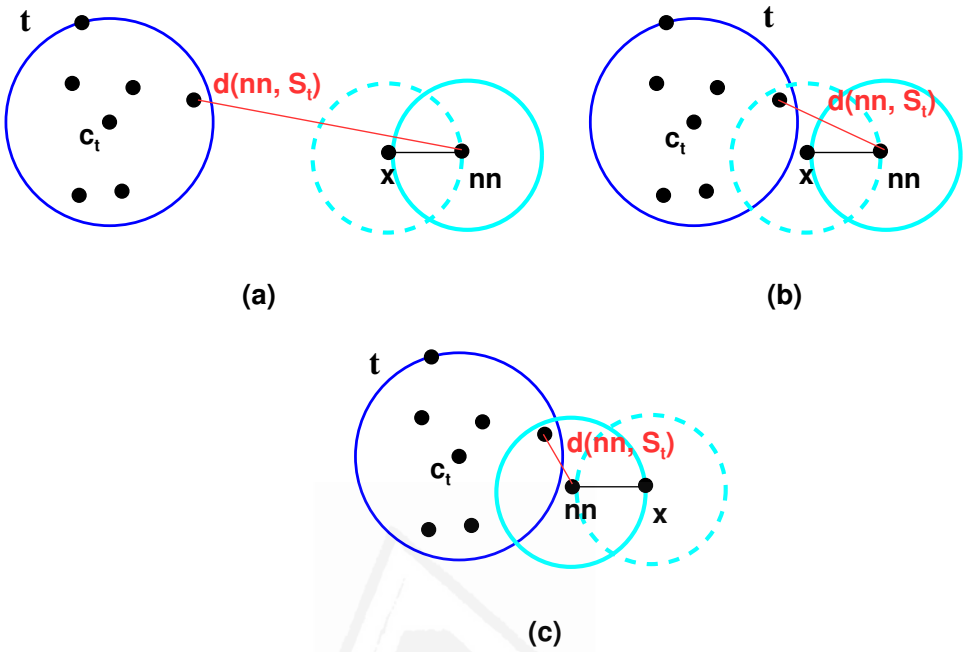


Figura 5.6: Aplicación de la regla de la tabla.

Demostración. Sea t un nodo del árbol y sea p un prototipo de S_t. Por la definición de la distancia entre un prototipo y un nodo

$$d(nn, S_t) = \min_{\mathrm{p}\in \mathrm{S_t}} \mathrm{d(p, nn)}$$

y por tanto

$$d(nn, S_t) \leq d(p, nn).$$

Por otro lado, por la desigualdad triangular:

$$d(p, nn) \leq d(p, x) + d(x, nn).$$

Combinando estas desigualdades, obtenemos:

$$\begin{aligned} & d(nn, S_t) && \leq d(p, nn) \leq d(p, x) + d(x, nn) \\ \Rightarrow \quad & d(p, x) && \geq d(nn, S_t) - d(x, nn) \end{aligned}$$

y utilizando la regla de la tabla, obtenemos al final

$$d(p, x) \geq 2d(x, nn) - d(x, nn) = d(x, nn).$$

□

En la figura 5.6 se representa gráficamente la regla de la tabla. En el caso (a) la distancia $d(nn, t)$ entre el prototipo más cercano hasta el momento nn y el prototipo más cercano del nodo t es mayor que $2d(x, nn)$, por lo que los prototipos del nodo t no pueden estar más cercanos a x que nn y el nodo t es podado aplicando la regla de la tabla TR. Sin embargo, en los casos (b) y (c), se observa como $d(t, nn)$ no es mayor que $2d(x, nn)$ y, por lo tanto, el nodo t no se puede podar, aunque como vemos en (c) ningún prototipo en t estaría más cercano a x que nn.

En las gráficas de los experimentos que se muestran a continuación nos referimos a esta regla con la letra "t".

5.5. Experimentos con datos sintéticos

En esta sección se han realizado experimentos con datos sintéticos para comprobar la efectividad de las reglas propuestas en la eliminación de nodos del árbol durante la búsqueda del vecino más cercano. El algoritmo 14 es el que realiza la búsqueda partiendo de un nodo, la función `Eliminado()` corresponde a la aplicación de una de las reglas de poda propuestas.

Los conjuntos de prototipos con los que se han realizado los experimentos son los mismos que los que se utilizaron para estudiar la construcción del árbol en el capítulo anterior. En los experimentos se han utilizado 1000 muestras y diferentes tamaños del conjunto de prototipos, desde 10000 hasta 110000. La dimensión de los datos varía de 2 a 20. Para cada dimensión y cada conjunto de prototipos se han realizado un total de 10 experimentos para 1000 muestras. Cada punto de la gráfica representa la media de 10 experimentos, donde el intervalo de confianza no supera el 2 % de la medida.

Los mismos experimentos se han repetido con algunos de los árboles propuestos. En las figuras 5.7, 5.9 y 5.11 se muestra el número medio de distancias calculadas empleando las reglas de eliminación propuestas para los árboles SFF, PRF y PMF respectivamente. En las figuras 5.8, 5.10 y 5.12 se muestra el número medio de nodos visitados. Para el árbol SFF, que era el que peor resultados mostraba en el capítulo anterior comparado con los otros árboles propuestos, el uso de la regla TR no supone un ahorro en el número de distancias calculadas respecto a la regla FNR. Sin embargo, la regla GR sí supone un ahorro considerable en dimensiones bajas, de un 70 % en dimensiones 2 y 4, de un 50 % en dimensión 8 y de casi un 20 % en dimensión 12, consiguiendo mejorar los resultados obtenidos por VPT en dimensiones bajas.

Sin embargo, para dimensiones 16 y 20, el número de distancias calculado es similar al obtenido con la regla FNR, y el método SAT sigue calculando

Algoritmo 14 `Buscar`(t, x , nn, d_{nn})

Entrada: t (nodo del árbol);
x (muestra);

1: **si** no existe hi_t **entonces**
2: volver;
3: **fin si**
4: $d_{iz} = d(c_{hi_t}, x)$;
5: **si** $d_{iz} < d_{nn}$ **entonces**
6: $d_{nn} = d_{iz}$;
7: $nn = c_{hi_t}$;
8: **fin si**
9: **si** no existe hd_t **entonces**
10: **si** no `Eliminado`(hi_t) **entonces**
11: `Buscar`(hi_t, x, nn, d_{nn});
12: **fin si**
13: volver;
14: **fin si**
15: $d_{de} = d(c_{hd_t}, x)$;
16: **si** $d_{de} < d_{nn}$ **entonces**
17: $d_{nn} = d_{de}$;
18: $nn = m_{hd_t}$;
19: **fin si**
20: **si** $d_{iz} < d_{de}$ **entonces**
21: **si** no `Eliminado`(hi_t) **entonces**
22: `Buscar`(hi_t, x, nn, d_{nn});
23: **fin si**
24: **si** no `Eliminado`(hd_t) **entonces**
25: `Buscar`(hd_t, x, nn, d_{nn});
26: **fin si**
27: **si no**
28: **si** no `Eliminado`(hd_t) **entonces**
29: `Buscar`(hd_t, x, nn, d_{nn});
30: **fin si**
31: **si** no `Eliminado`(hi_t) **entonces**
32: `Buscar`(hi_t, x, nn, d_{nn});
33: **fin si**
34: **fin si**

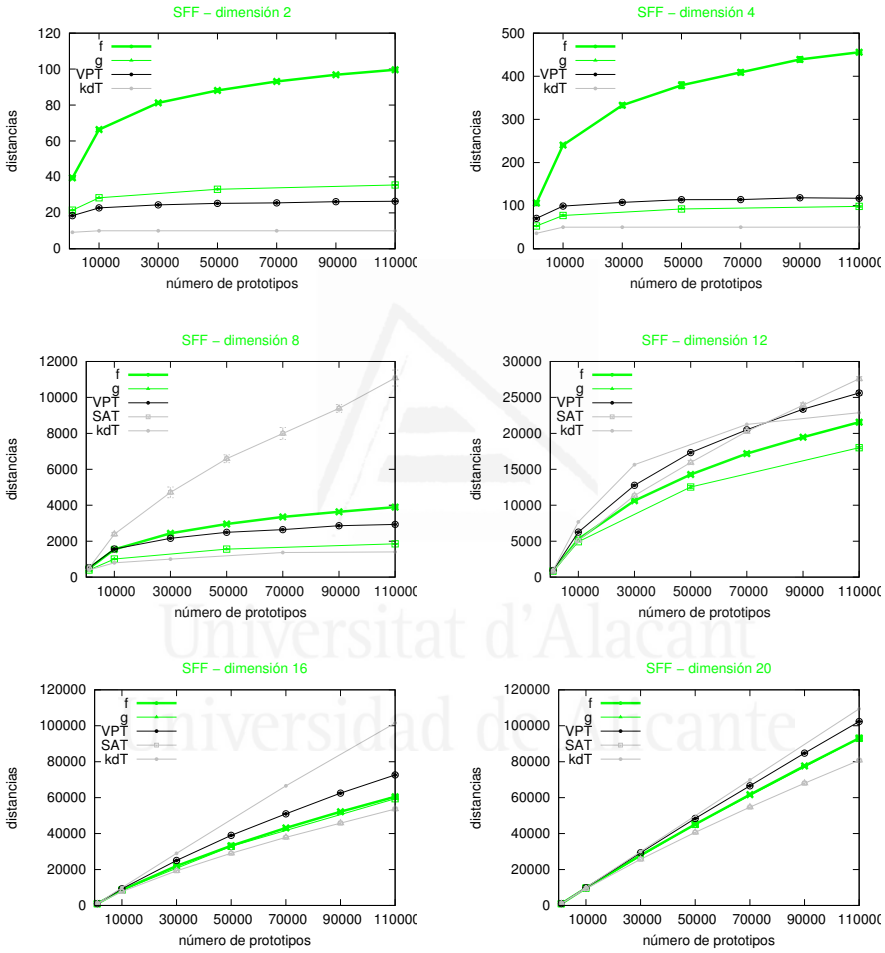


Figura 5.7: Número de distancias calculadas en la fase de búsqueda al emplear las reglas de eliminación propuestas en un árbol SFF comparando con VPT, SAT y *kd*-tree.

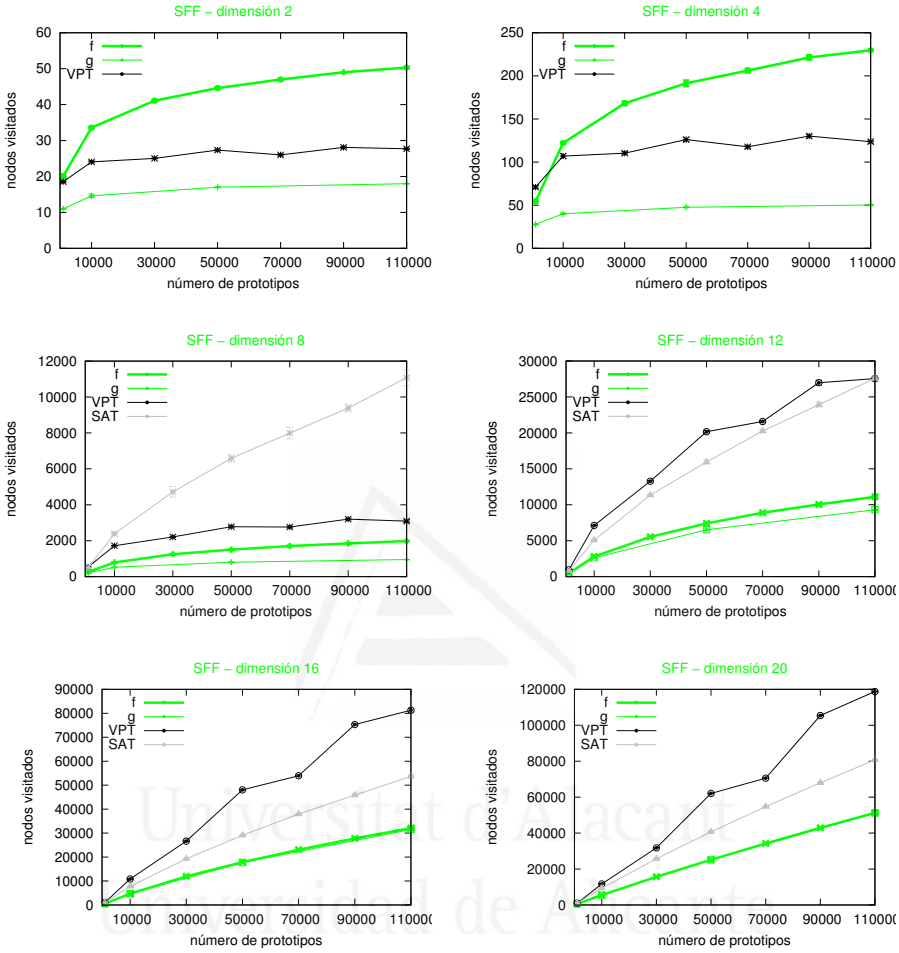


Figura 5.8: Número medio de nodos visitados en la fase de búsqueda al emplear las reglas de eliminación propuestas en un árbol SFF comparando con VPT y SAT.

menos distancias que al utilizar el árbol SFF. Respecto al número medio de nodos visitados con el árbol SFF se repite el comportamiento observado respecto a la regla FNR para el número medio de distancias calculadas. En este caso ya el empleo de la regla FNR hace que se visiten menos nodos que con el empleo de la técnica SAT.

En el árbol PRF podemos observar que tanto la regla GR[3] como la TR reducen considerablemente el número de distancias a calcular y, en el caso

[3]En las gráficas para dimensiones altas aparece solapada muchas veces la curva para la regla GR con la regla FNR.

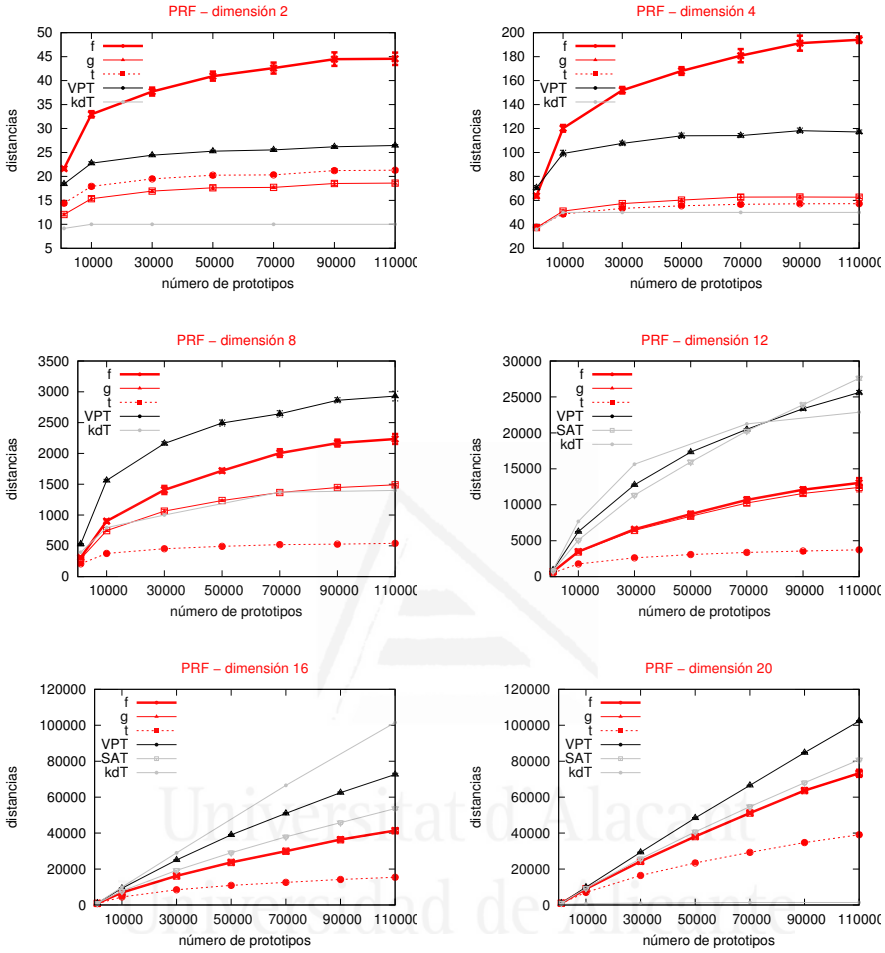


Figura 5.9: Número medio de distancias calculadas en la fase de búsqueda al emplear las reglas de eliminación propuestas en un árbol PRF comparando con VPT, SAT y *kd*-tree.

de la regla TR, mantiene este buen comportamiento incluso en dimensiones altas llegando a ahorrar más de un tercio de las distancias a calcular con la regla FNR para dimensión 20. Sin embargo, el número de nodos que se visitan empleando la regla TR no es inferior de un modo general a los visitados con la regla FNR llegando incluso en dimensión 20 a realizar más visitas.

En el caso del árbol PMF se repite el comportamiento observado en el árbol PRF para la regla GR y la regla TR tanto en el número de distancias calculadas como en el de nodos visitados.

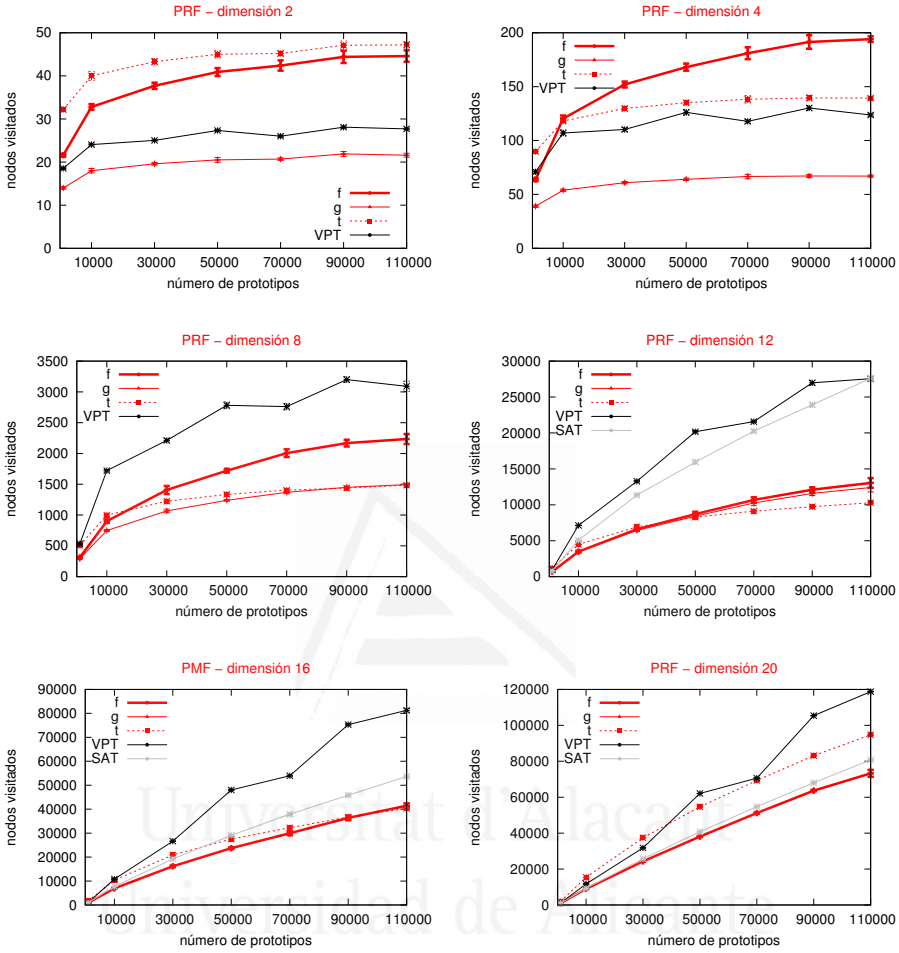


Figura 5.10: Número medio de nodos visitados en la fase de búsqueda al emplear las reglas de eliminación propuestas en un árbol PRF comparando con VPT y SAT.

De estos experimentos podemos deducir que el empleo de las reglas propuestas suponen un ahorro importante en el número de distancias calculadas y, si la dimensión del espacio de trabajo no es muy elevada, en el número de nodos visitados.

5.6. Combinación de reglas

Ante la pregunta de si un nodo puede ser podado o no, se deduce que la respuesta con reglas de eliminación diferentes puede ser también diferente.

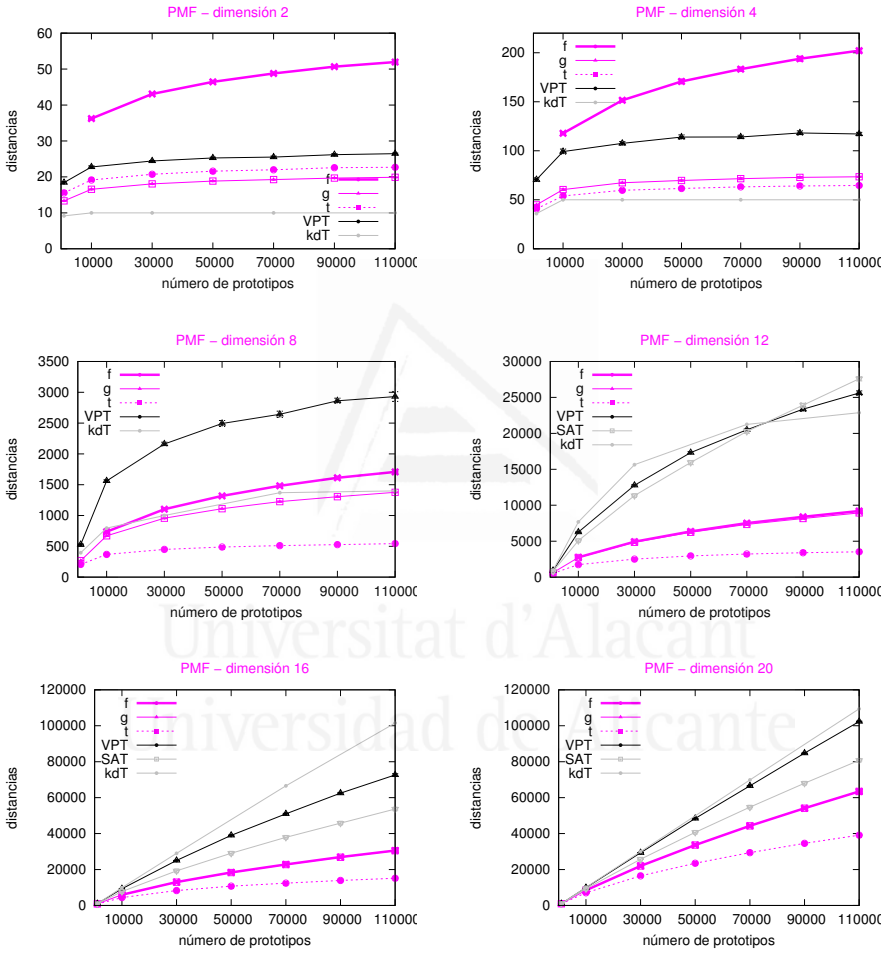


Figura 5.11: Número de distancias calculadas en la fase de búsqueda al emplear las reglas de eliminación propuestas en un árbol PMF comparando con VPT, SAT y *kd*-tree.

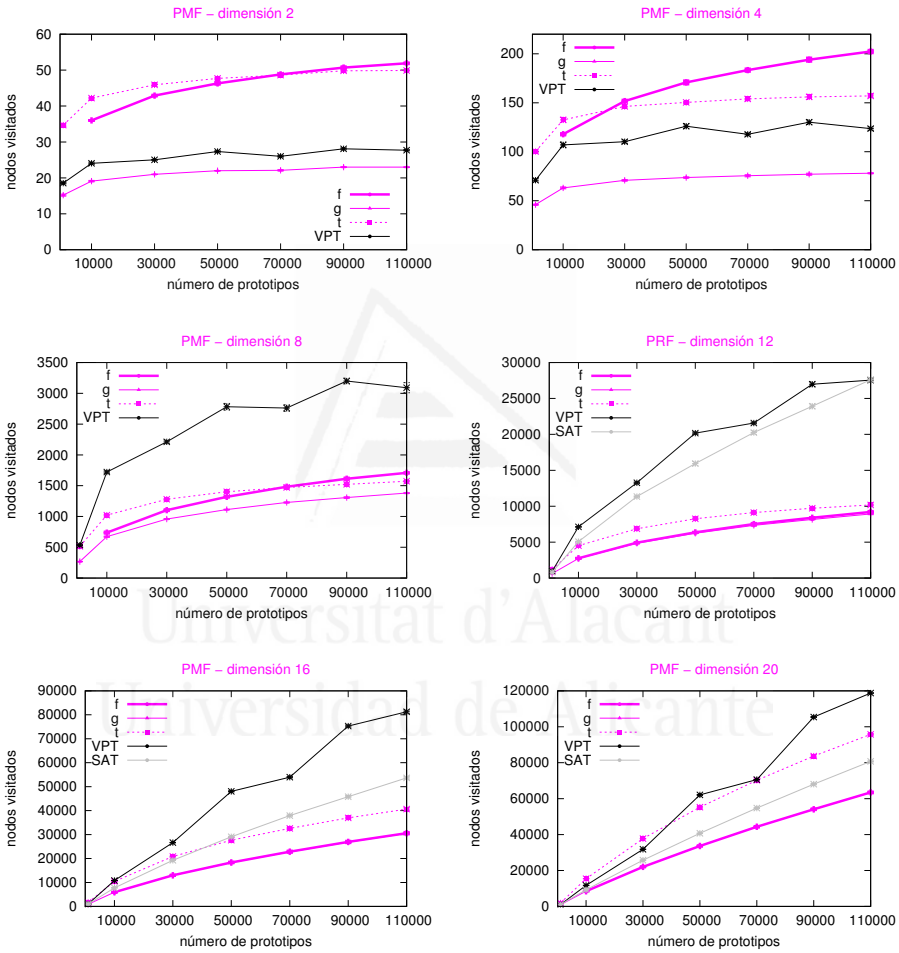


Figura 5.12: Número medio de nodos visitados en la fase de búsqueda al emplear las reglas de eliminación propuestas en un árbol PMF comparando con VPT y SAT.

Es interesante estudiar si el ahorro en distancias calculadas cuando se aplica una regla respecto al ahorro en distancias cuando se aplica otra, responde a que una regla poda nodos más poblados, más cercanos a la raíz que la otra, o bien a que, independientemente del nivel del árbol, una regla es más adecuada para la poda que otra.

En el siguiente grupo de experimentos realizados se ha estudiado el comportamiento de las reglas de eliminación en un conjunto de 10000 prototipos. Estos experimentos se han llevado a cabo sobre árboles PRF y en ellos se contabilizan los nodos que se podaban dentro de unos intervalos de población de los nodos. En las figuras 5.13 y 5.14 se representa el histograma de los nodos podados para experimentos realizados en dimensiones 2 y 8. Para cada dimensión se presentan 2 histogramas, que corresponden a 2 experimentos independientes en los que se buscaba el vecino más cercano a una muestra. La idea era comprobar si había algún comportamiento que caracterizase a cada regla de manera general.

Como se puede ver en las figuras 5.13 y 5.14, la regla GR y la regla TR son las que podan los nodos que contienen un mayor número de prototipos. Esto supone la reducción drástica del espacio de búsqueda que conlleva por tanto menor número de distancias calculadas. Por otro lado, resulta interesante observar que al aplicar la regla FNR y la regla GR, en estos experimentos no se visitan nodos hoja mientras que al aplicar las otras dos reglas sí.

Los resultados obtenidos para la regla SBR confirman la poca capacidad de poda que tiene esta regla (tabla 5.1), pues como se puede observar, independientemente de la dimensión, es la regla que menos nodos elimina y la que más nodos hoja visita.

5.6.1. Algoritmo de Búsqueda

Ya que, tal y como se ha comentado en el apartado anterior, cada regla tiene sus particularidades, hemos definido un algoritmo que trabaja con una combinación de las reglas de eliminación para aplicarlas en el orden más conveniente.

El algoritmo 14 propone una estrategia general de ramificación y poda para la búsqueda del vecino más cercano en un árbol binario. Cuando se visita un nodo, se calcula la distancia de la muestra x al representante de cada uno de los nodos hijo[4] y se actualiza, si fuese necesario, el vecino más cercano. Se continúa explorando el nodo cuyo representante a la muestra está más cerca (paso 16 del algoritmo).

[4]Si se aplica a árboles PRF, PMF o PMM sólo se calcula la distancia al representante de uno de los hijos, ya que el representante del otro coincide con el padre y ya se conoce

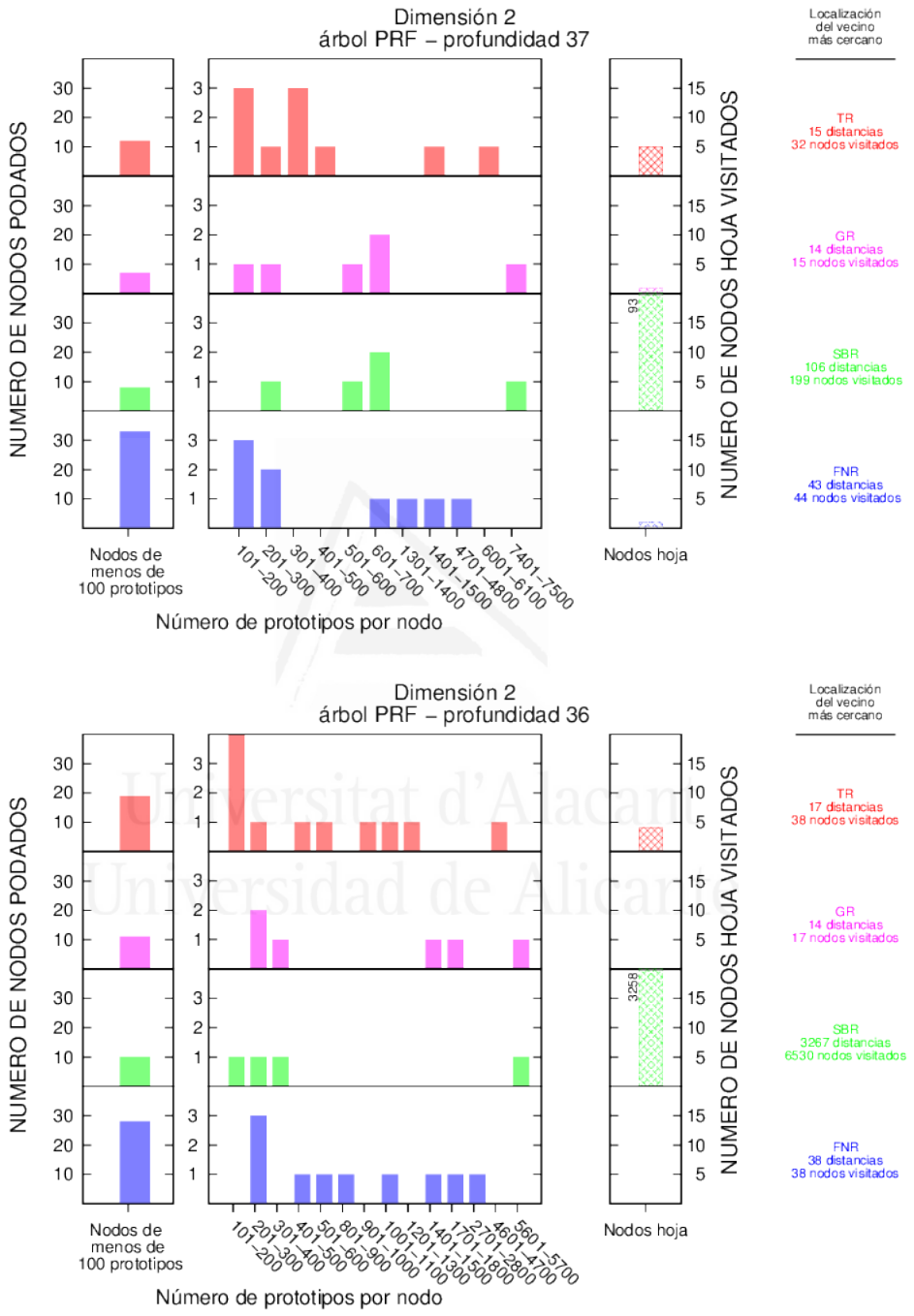


Figura 5.13: Número de nodos podados, clasificados por tamaño del nodo, al aplicar las distintas reglas de eliminación en dos árboles PRF de 10000 prototipos de dimensión 2 y una muestra.

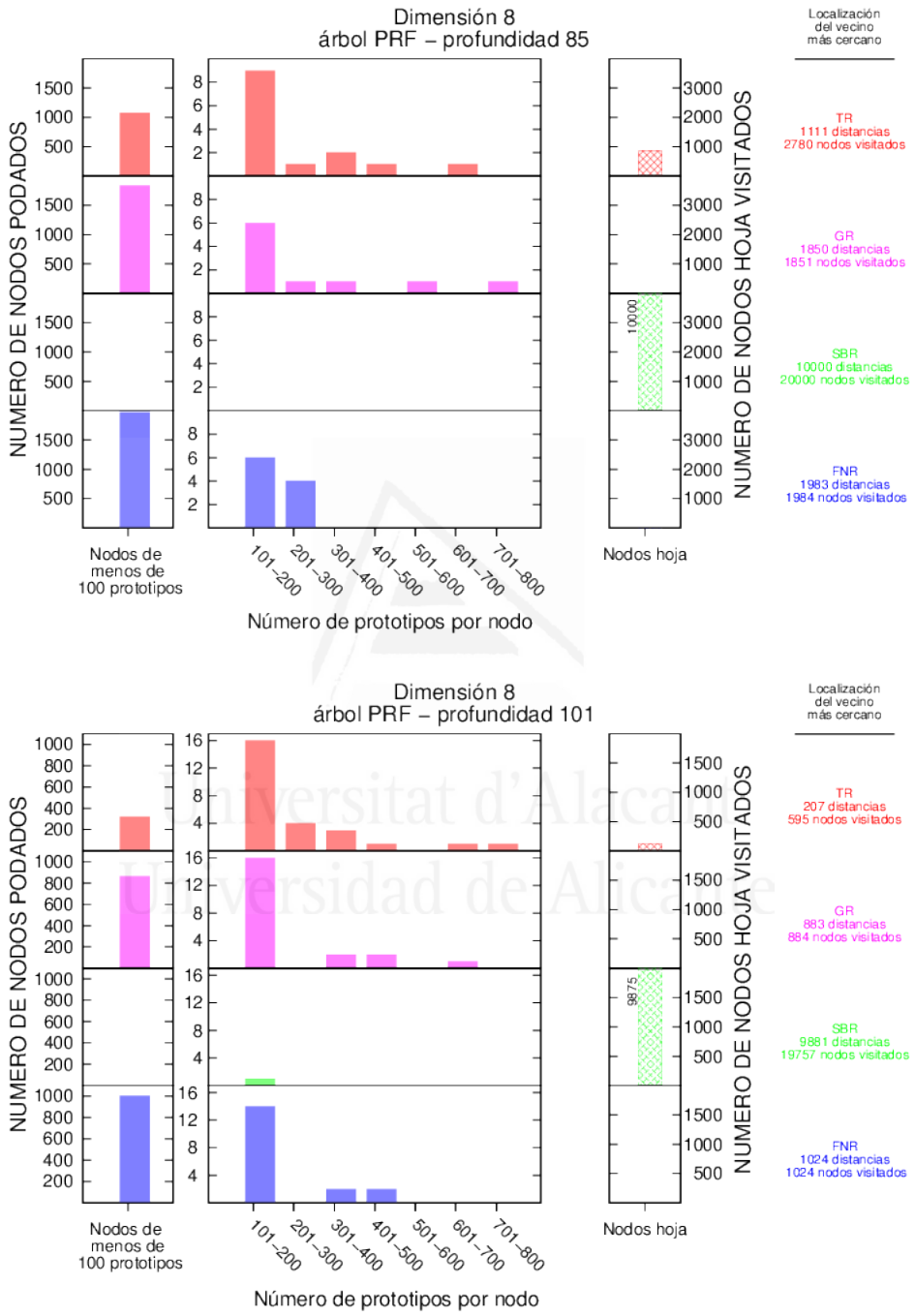


Figura 5.14: Número de nodos podados, clasificados por tamaño del nodo, al aplicar las distintas reglas de eliminación en dos árboles PRF de 10000 prototipos de dimensión 8 y una muestra.

Evidentemente un nodo se explora sólo si no puede ser eliminado utilizando alguna o varias de las reglas que se han descrito. Para optimizar el uso de las nuevas reglas propuestas, la función `Eliminado()` utilizada en el algoritmo, debe combinar la evaluación de estas reglas en el orden en que citan en el algoritmo 15.

El orden en el que se aplican las reglas obedece al intento de podar el nodo evitando calcular nuevas distancias. En este sentido, se comienza aplicando la regla SBR y la regla TR, ya que ambas se basan en distancias que se han calculado en tiempo de preproceso durante la construcción del árbol y, por lo tanto, son distancias que ya conocemos y tenemos almacenadas [5].

5.6.2. Experimentos con datos sintéticos

Para comprobar la eficacia de la combinación de estas reglas, se ha realizado una serie de experimentos en los que se aplicaban las reglas de eliminación de manera independiente, una a una y, también todas combinadas (fsgt) según el algoritmo 15. Estos experimentos se han realizado con los árboles SFF, PRF y PMF. En las gráficas 5.15 y 5.17 se muestra el número medio de distancias que se deben calcular hasta encontrar el vecino más cercano en los árboles PRF y PMF respectivamente, y en las gráficas 5.16 y 5.18 el número medio de nodos que se visitan. Cada punto de una gráfica representa la media de 10 experimentos.

Como se puede observar, la combinación de todas las reglas es la que calcula el menor número de distancias y visita el menor número de nodos en cualquiera de los árboles considerados y para todas las dimensiones con las que se ha trabajado. Además el ahorro es muy significativo, por ejemplo, para dimensión 20, se calculan alrededor de un 50 % menos de distancias respecto al uso de la regla FNR y se visitan también un 50 % menos de nodos. También se puede observar que la combinación de todas las reglas tiene un efecto similar al uso combinado de la regla TR junto a la FNR o de la regla TR junto a la GR. Si no se emplean las reglas combinadas, se observa que la capacidad de eliminación de nodos de la regla TR por sí sola es muy similar a la de la combinación de todas las reglas independientemente de la dimensión de los prototipos.

5.7. Reduciendo el tamaño de la tabla

Como se ha comentado en la sección anterior, para aplicar la regla TR hay que almacenar las distancias entre todos los objetos del conjunto de datos

[5] Publicación derivada de este estudio (Gómez-Ballester et al., 2010)

Algoritmo 15 `Buscar_CRE`(t, x, nn, d_{nn})

Entrada: t (nodo del árbol);
x (muestra);

1: **si** t no es un nodo hoja **entonces**
2: **si** (`SBR`(hi_t) || `TR`(hi_t)) **entonces**
3: **si** (no `FNR`(hd_t)) $\wedge$ (no `TR`(hd_t)) **entonces**
4: `Buscar_CRE`(hd_t, x, nn, d_{nn}); *// el nodo hijo de la izquierda se ha podado*
5: **fin si**
6: Volver; *// se han podado los 2 hijos*
7: **fin si**
8: $d_{de} = d(x, m_{hd_t})$; *// se calcula la distancia a los representantes de los nodos hijo*
9: $d_{iz} = d(x, m_{hi_t})$
10: **si** $d_{iz} < d_{nn}$ **entonces**
11: $nn = m_{hi_t}$; $d_{nn} = d_{iz}$; *// se actualiza el vecino más cercano y su distancia*
12: **fin si**
13: **si** $d_{de} < d_{nn}$ **entonces**
14: $nn = m_{hd_t}$; $d_{nn} = d_{de}$; *// se actualiza el vecino más cercano y su distancia*
15: **fin si**
16: **si** `Activa`(GR) **entonces**
17: **si** $d_{iz} < d_{de}$ **entonces**
18: **si** (no `GR`(hi_t)) **entonces** `Buscar_CRE`(hi_t, x, nn, d_{nn}); **fin si**
19: **si** (no `GR`(hd_t)) **entonces** `Buscar_CRE`(hd_t, x, nn, d_{nn}); **fin si**
20: **si no**
21: **si** (no `GR`(hd_t)) **entonces** `Buscar_CRE`(hd_t, x, nn, d_{nn}); **fin si**
22: **si** (no `GR`(hi_t)) **entonces** `Buscar_CRE`(hi_t, x, nn, d_{nn}); **fin si**
23: **fin si**
24: **si no**
25: **si** $d_{iz} < d_{de}$ **entonces**
26: **si** (no `FNR`(hi_t)) $\wedge$ (no `SBR`(hi_t)) **entonces** `Buscar_CRE`(hi_t, x, nn, d_{nn});
27: **fin si**
28: **si** (no `FNR`(hd_t)) $\wedge$ (no `SBR`(hd_t)) **entonces** `Buscar_CRE`(hd_t, x, nn, d_{nn});
29: **fin si**
30: **si no**
31: **si** (no `FNR`(hd_t)) $\wedge$ (no `SBR`(hd_t)) **entonces** `Buscar_CRE`(hd_t, x, nn, d_{nn});
32: **fin si**
33: **si** (no `FNR`(hi_t)) $\wedge$ (no `SBR`(hi_t)) **entonces** `Buscar_CRE`(hi_t, x, nn, d_{nn});
34: **fin si**
35: **fin si**
36: **fin si**
37: **fin si**

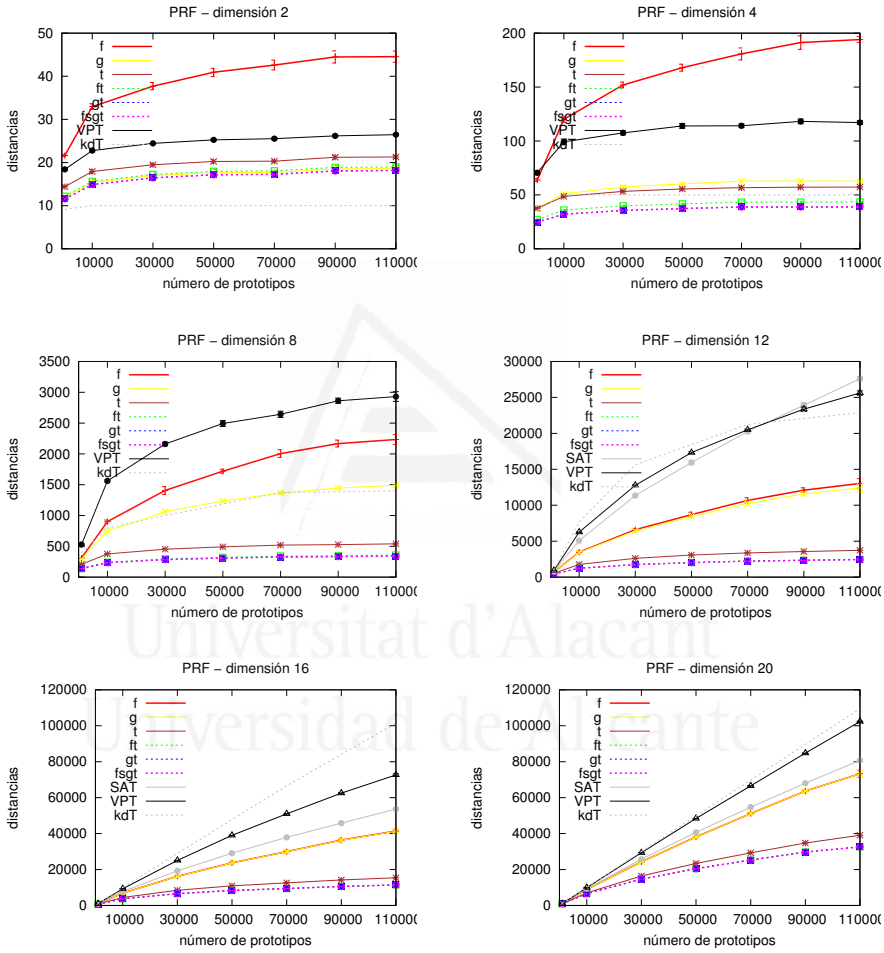


Figura 5.15: Número medio de distancias calculadas en la fase de búsqueda al emplear combinadas las reglas de eliminación propuestas en un árbol PRF comparando con VPT, SAT y *kd*-tree.

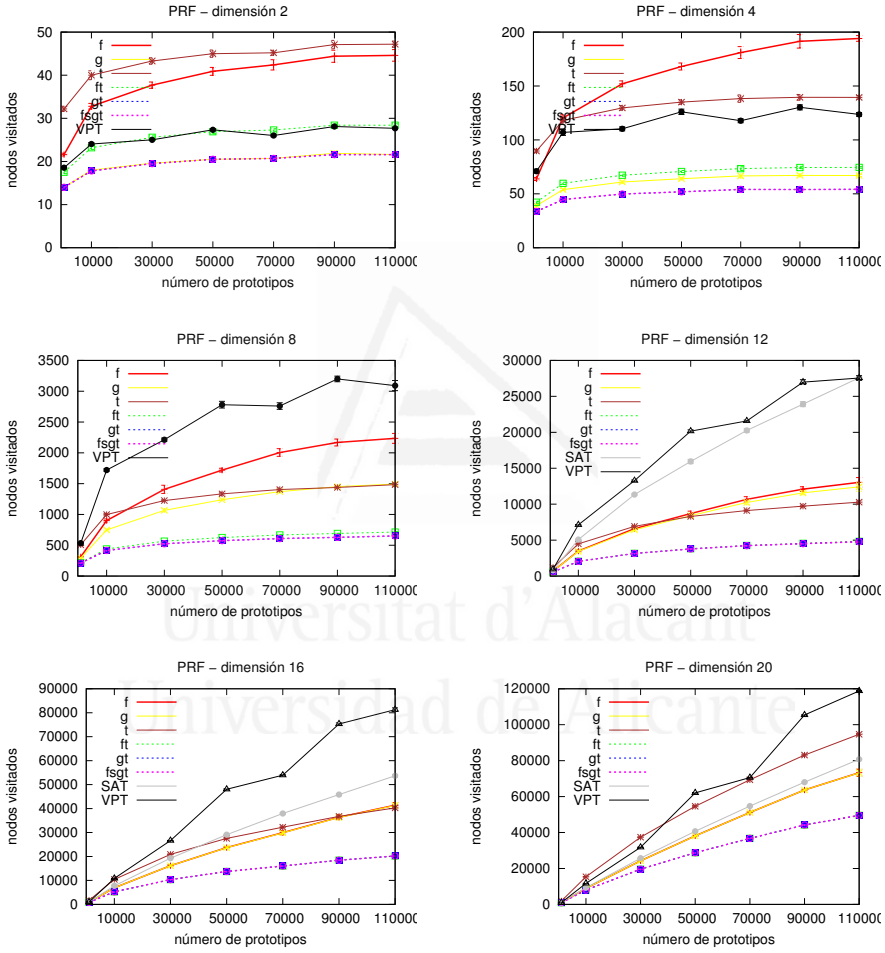


Figura 5.16: Número medio de nodos visitados en la fase de búsqueda al emplear las reglas de eliminación propuestas en un árbol PRF comparando con VPT y SAT.

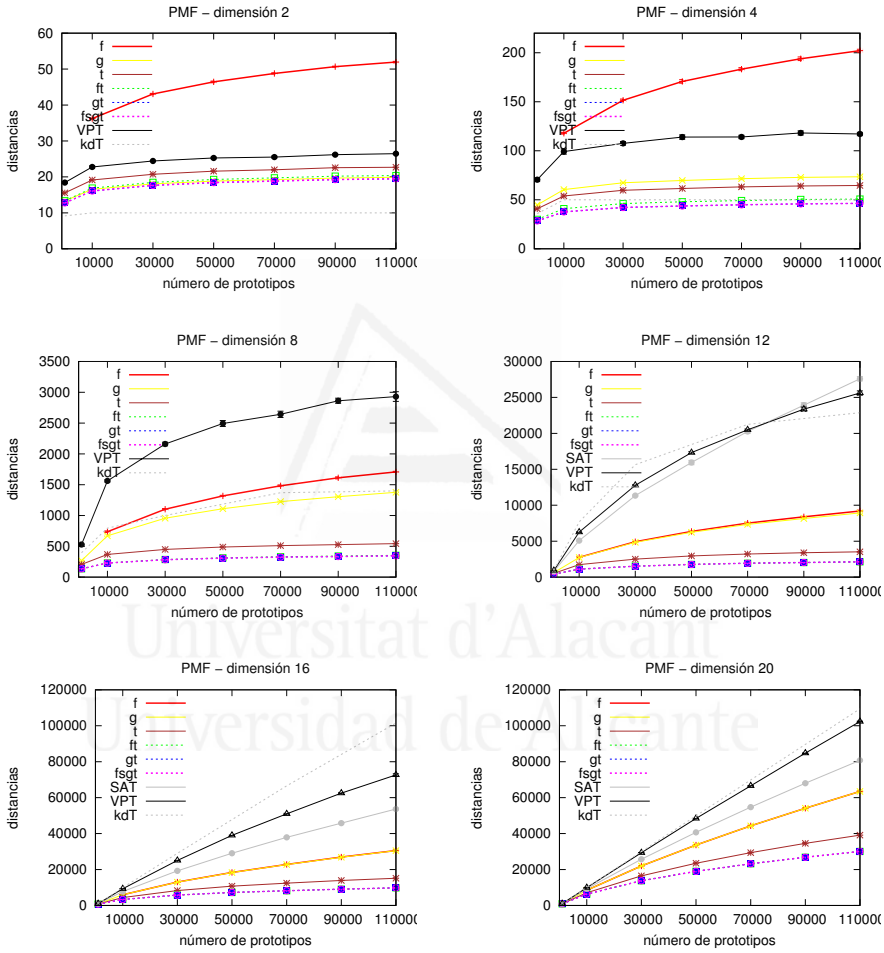


Figura 5.17: Número de distancias calculadas en la fase de búsqueda al emplear las reglas de eliminación propuestas con distintas combinaciones en un árbol PMF comparando con VPT, SAT y *kd*-tree.

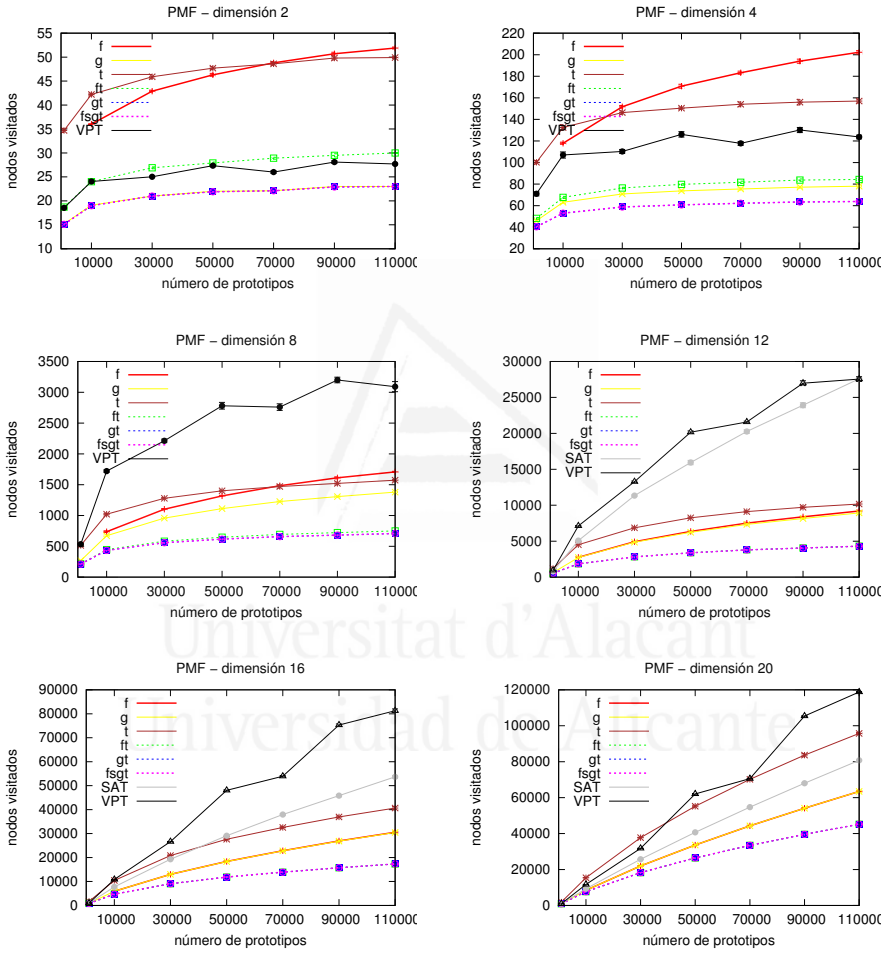


Figura 5.18: Número medio de nodos visitados en la fase de búsqueda al emplear combinadas las reglas de eliminación propuestas con distintas combinaciones en un árbol PMF comparando con VPT y SAT.

y todos los nodos del árbol. Ya que el tamaño de la tabla crece de manera proporcional a n^2 siendo n el número de prototipos, la complejidad espacial puede convertirse en un cuello de botella para utilizar esta regla. En esta sección presentamos algunas propuestas sobre cómo detectar, en tiempo de preproceso, en qué circunstancias la aplicación de TR no será útil [6]. En estos casos, podemos evitar el cálculo y almacenamiento de distancias. Es evidente que para conseguir una reducción efectiva del espacio de almacenamiento, la tabla debe implementarse como una matriz dispersa.

Para definir las situaciones en las que el uso de la regla de eliminación TR será "probablemente" inútil, necesitamos definir $\hat{d}_{nn}$, que será el valor esperado de la distancia de la muestra al vecino más cercano, $\hat{d}(x, nn)$. Este valor se desconoce en tiempo de preproceso, por lo que lo utilizaremos como un parámetro en nuestra propuesta para reducir el tamaño de la tabla:

1. Si se cumple que $\hat{d}_{nn} > \frac{d(p,t)}{2}$ la regla TR no va a podar, por lo que no es necesario almacenar la distancia $d(p, t)$ en la tabla. Esta es una situación en la que la regla TR no se cumple porque el prototipo p está muy cercano al nodo t.

2. Si se cumple que $\hat{d}_{nn} < d(c_t, p) - r_t$, esperamos que la regla FNR elimine el nodo, y entonces el resultado de TR no es útil. Esta fórmula se deriva de una aproximación de FNR donde suponemos que p está muy cercano a la muestra y por tanto $d(c_t, p) \sim d(c_t, x)$ (normalmente este caso se plantea después de varios pasos del algoritmo 14). En este caso la regla FNR será efectiva y por tanto probablemente no se necesite la regla TR para podar el nodo t, por lo que no es necesario almacenar la distancia $d(p, t)$ en la tabla.

Si no se estima bien el parámetro, o bien estamos almacenando casi toda la tabla, o bien puede que estemos dejando de almacenar distancias que en la fase de búsqueda será necesario calcular. En el siguiente apartado se analiza el precio a pagar, en número de distancias calculadas y de nodos visitados, respecto a la reducción de tamaño de la tabla que se almacena cuando se aplica la técnica propuesta para reducir el tamaño de la tabla en la que se basa la regla TR.

5.7.1. Experimentos con datos sintéticos

Se han realizado una serie de experimentos para estimar si se puede conseguir una reducción significativa del tamaño de la tabla, ajustando el parámetro $\hat{d}_{nn}$, sin que esto conlleve un aumento notable del número de distancias

[6] Publicación derivada de este estudio (Gómez-Ballester et al., 2008)

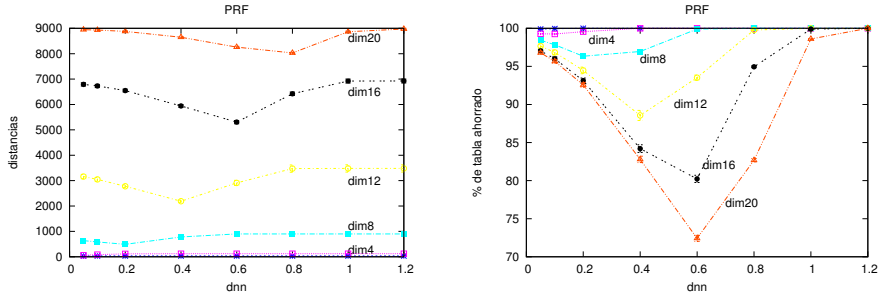


Figura 5.19: Número de distancias calculadas y de espacio ahorrado para distintos valores del parámetro $\hat{d}_{nn}$ y de la dimensión de los prototipos en árboles PRF con 10000 prototipos.

calculadas. Los experimentos se han realizado con el árbol PRF. En estos experimentos se compara el número de distancias que se calculan utilizando sólo la regla de eliminación FNR (f en las gráficas), frente a las que se calculan si se utiliza esta regla combinada con la regla TR almacenando la tabla completa (ft en las gráficas), y las que se obtienen utilizando la regla FNR y la regla TR pero utilizando sólo una parte de la tabla representada en tanto por ciento (ft-dnn en las gráficas).

En la figura 5.19 se muestran tanto el número de distancias que se calculan como el porcentaje de tabla ahorrado para 10000 prototipos en diferentes dimensiones. Como se puede observar los valores del parámetro $\hat{d}_{nn}$ para conseguir ahorrar un porcentaje concreto de distancias dependen de la dimensión de los prototipos.

Es importante destacar que si se aplica la regla de eliminación TR junto a la regla de eliminación FNR, y se almacena tan sólo un 10 % de la tabla, se podría ahorrar alrededor de un 40 % de distancias calculadas respecto a las distancias calculadas utilizando sólo la regla de eliminación FNR, como se puede observar para dimensión 8 y 12 en la figura 5.20.

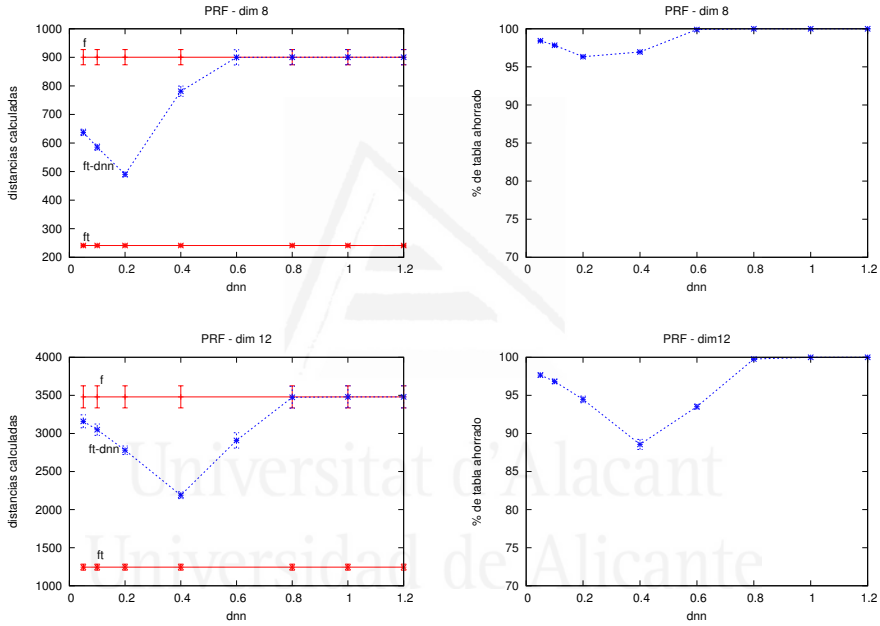


Figura 5.20: Número de distancias calculadas y porcentaje de espacio ahorrado para distintos valores del parámetro $\hat{d}_{nn}$ en árboles PRF con 10000 prototipos de dimensión 8 y 12.

Capítulo 6

Experimentos con datos reales

Se han realizado experimentos con datos reales para analizar el comportamiento de los algoritmos con distintos tipos de distribución de los datos. En estos experimentos se han analizado los conceptos que podían ser medidos en los algoritmos a los que se ha hecho referencia, como son la profundidad y solapamiento en los árboles, y el número medio de distancias calculadas y de nodos visitados durante la búsqueda.

A continuación se detallan las bases de datos que se han utilizado para realizar los experimentos.

Base de datos de fonemas

Esta base de datos se utilizó en el European ESPRIT 5516 project: ROARS (ELENA, 2004) cuyo objetivo era el desarrollo y la implementación de un sistema de análisis en tiempo real para el reconocimiento de habla francesa y española. La base de datos contiene vocales procedentes de 1809 sílabas aisladas (pa, ta, pan, ...). Para cada vocal se recogieron 5 características (la amplitud de los 5 primeros armónicos) en tres momentos diferentes, lo que da un total de 5427 vectores de características diferentes, de los se descartaron 23. La composición final de la base de datos con la que se ha trabajado son por tanto 5404 vectores de dimensión 5.

Base de datos de imágenes de satélite

Esta base de datos se ha extraído de la UCI Machine Learnig Repository (Asuncion and Newman, 2007). La base de datos representa una pequeña subárea de una escena con características como tipos de suelo o cultivos, y contiene 6435 imágenes con 36 características.

Diccionario de inglés

Es una colección de 69069 palabras de un diccionario de inglés tomada del repositorio del SISAP (Figueroa et al., 2007), página web oficial del International Workshop on Similarity Search and Applications [1].

6.1. Incidencia de la construcción del árbol en la fase de búsqueda.

Los primeros experimentos se han llevado a cabo con el objetivo de valorar la importancia que tiene, sobre la posterior fase de búsqueda, la estrategia de distribución de los prototipos en los nodos al construir el árbol. En los primeros experimentos se ha trabajado aplicando en la búsqueda únicamente la regla de eliminación FNR y se han estudiado los distintos tipos de árboles propuestos. Se ha trabajado en todos los experimentos con un conjunto de 1000 muestras y se ha variado el tamaño del conjunto de prototipos.

Experimentos con la distancia euclídea

La distancia euclídea se ha utilizado en los experimentos realizados con la base de datos de los fonemas y la base de datos de las imágenes de satélite. Tanto en la base de datos de fonemas como en la base de datos de imágenes de satélite, los conjuntos de muestras se han seleccionado aleatoriamente de la base de datos y, del mismo modo, se han seleccionado los conjuntos de prototipos, excluyendo en éstos los prototipos que se hayan incluido en su correspondiente conjunto de muestras. Cada punto en las gráficas corresponde a la media de 10 experimentos.

En primer lugar se ha estudiado la profundidad de los árboles que se obtienen según la técnica de construcción del árbol empleada. Tal y como ocurre en los experimentos con datos sintéticos, los árboles PMF y PMM coinciden en profundidad y resultan los árboles más profundos, como se puede observar en la figura 6.1. En los árboles PRF la elección al azar del representante del nodo raíz produce que los resultados obtenidos para estos árboles presenten un intervalo de error superior al resto, como también se observa en los experimentos con datos sintéticos.

También se ha estudiado si la distribución de prototipos en los distintos tipos de árbol es similar al comportamiento que se ha observado al trabajar con datos sintéticos. En la figura 6.2 vemos cómo quedan distribuidos dos conjuntos distintos de 4000 prototipos en dos árboles SMF, así como

[1] `http://www.sisap.org/`

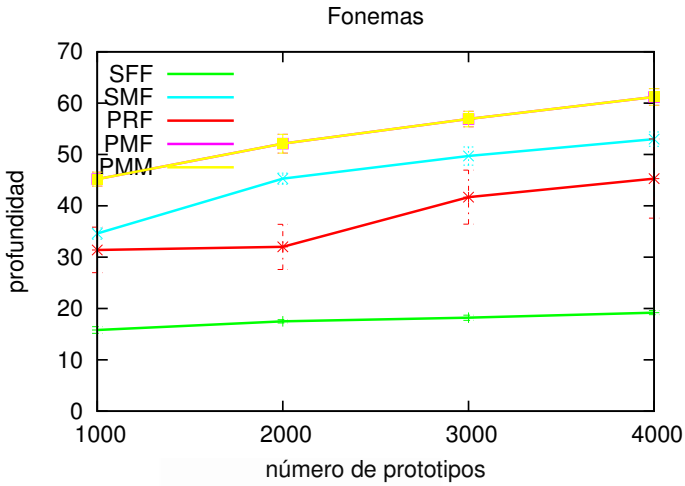


Figura 6.1: Profundidad media de los árboles para distinto tamaño del conjunto de prototipos trabajando con la base de datos de fonemas.

el solapamiento entre los nodos. Se puede observar que, al igual que en los experimentos con datos sintéticos, existe una gran diferencia entre el número de prototipos de dos nodos hermanos así como en su solapamiento.

En la figura 6.3 se muestra el modo en que se reparten esos dos mismos conjuntos de prototipos en dos árboles PRF, así como el solapamiento entre los nodos. Ya que en los árboles PRF el prototipo representante del nodo raíz se selecciona aleatoriamente, se puede observar la diferencia, en número de prototipos, que existe en los dos árboles que se muestran en cuanto a la descomposición del nodo raíz. En ambos casos se observa que la rama de la izquierda, en la que se repite el representante del padre, es la que mayor número de prototipos aloja, como se ha observado en los experimentos con datos sintéticos.

La distribución de prototipos en las distintas ramas del árbol, así como el solapamiento, se corresponde también con el comportamiento que se observaba con los datos sintéticos.

Se observa también (ver figura 6.4) que las construcciones de árbol en las que se repite el representante del nodo entre padre e hijo (PRF, PMF y PMM), son las que menor número de distancias calculan en la búsqueda [2].

En estos árboles se calculan tan sólo el 20 % de las distancias que calcula SAT. Incluso calculan, tanto para la base de datos de fonemas como la de

[2] recuérdese que en estos experimentos sólo se está usando la regla de eliminación FNR

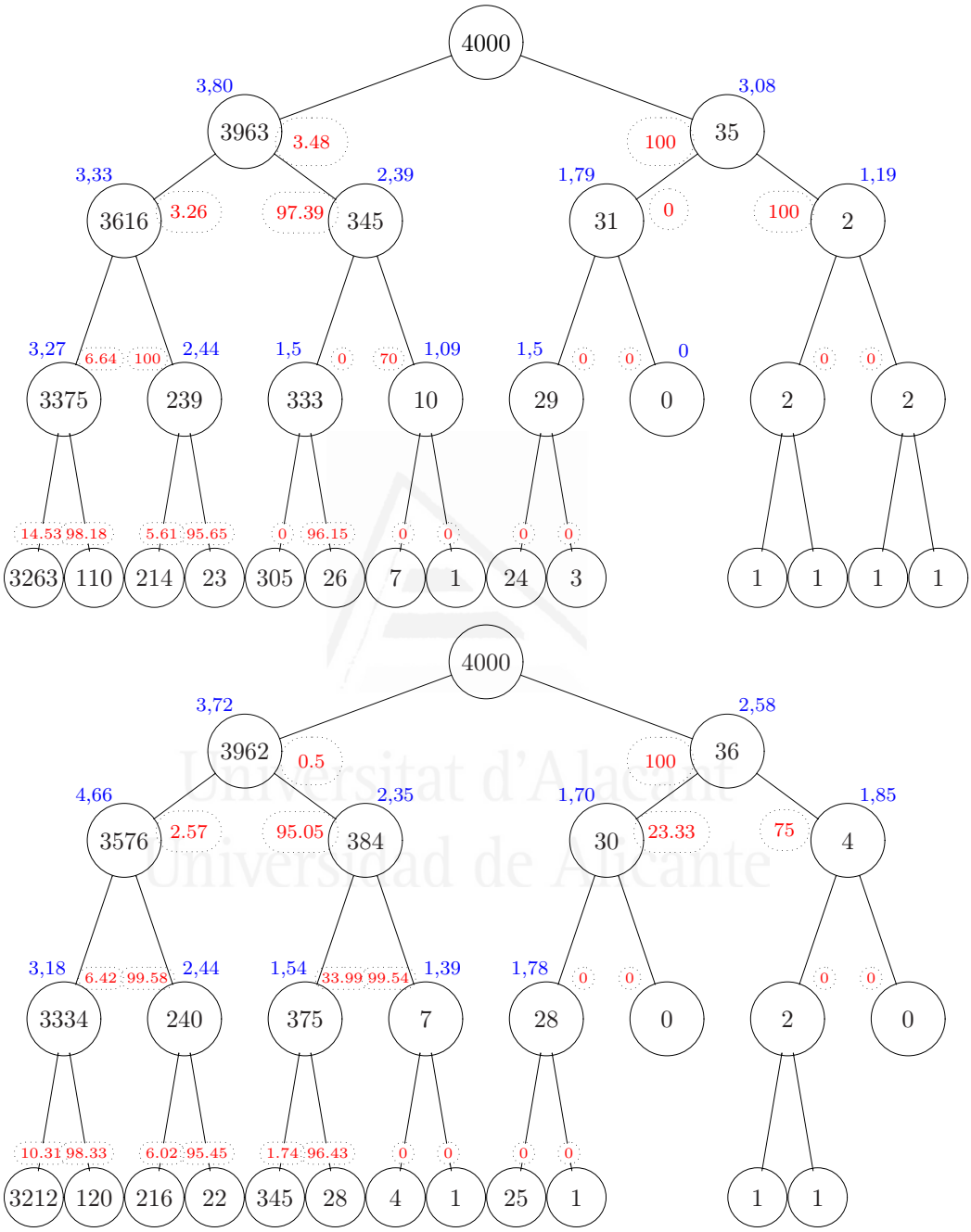


Figura 6.2: Parte superior de dos árboles SMF que corresponden a dos conjuntos diferentes de 4000 prototipos cada uno extraídos de la base de datos de fonemas.

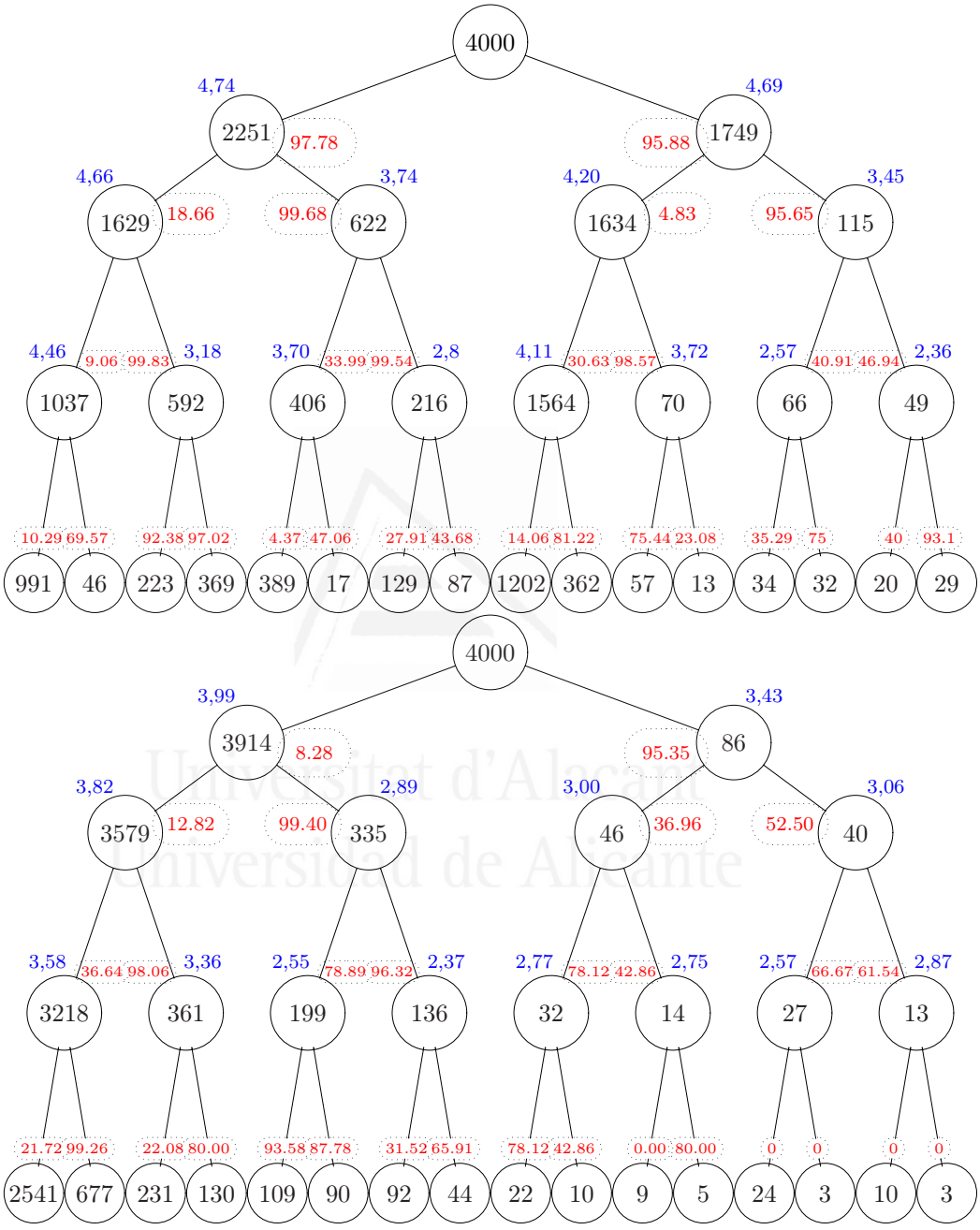


Figura 6.3: Parte superior de dos árboles PRF que corresponden a dos conjuntos diferentes de 4000 prototipos cada uno extraídos de la base de datos de fonemas.

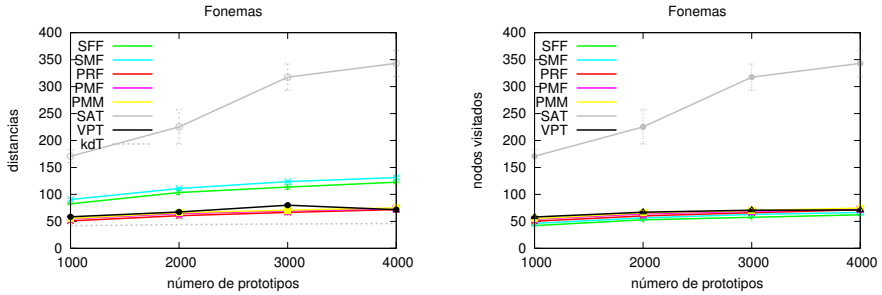


Figura 6.4: Número medio de distancias calculadas y de nodos visitados para encontrar el vecino más cercano a un fonema (representado por un vector de dimensión 5) utilizando la regla de eliminación FNR para distintos tipos de árbol.

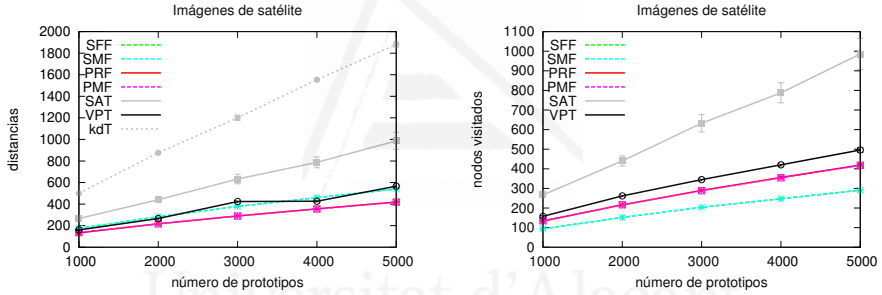


Figura 6.5: Número medio de distancias calculadas y de nodos visitados para encontrar el vecino más cercano en la base de datos Satimage (representado por un vector de dimensión 5) utilizando la regla de eliminación FNR para distintos tipos de árbol.

imágenes de satélite, menos distancias que el VPT, que es uno de los métodos que menos distancias calcula en dimensiones bajas según se muestra en los experimentos con datos sintéticos del capítulo 4. El *kd*-tree sigue mostando el buen comportamiento con datos de dimensión baja, y es el método que calcula menos distancias para la base de datos de los fonemas (dimensión 5), siendo el que más distancias calcula para la base de datos de imágenes de satélite (dimensión 36).

En cuanto al número de nodos visitados en la búsqueda del vecino más cercano es ligeramente menor si se utiliza un árbol SMF, como también ocurría con los datos sintéticos.

En la figura 6.5 se puede observar los resultados obtenidos con la base de

datos de imágenes de satélite que son similares a los observados para la base de datos de fonemas para los árboles propuestos.

Experimentos con la distancia de edición

Los correctores ortográficos son una herramienta de uso cotidiano. Esta herramienta busca cada palabra en un diccionario, lo que sería la base de datos de los prototipos y, en el caso de no encontrar la palabra en el mismo, propone otra u otras para dar la opción de corrección. En concreto, el corrector busca la palabra *más similar* a la que se da como entrada, es decir, realiza una búsqueda por similitud en el diccionario. Para determinar lo similar que es una palabra a otra se hace uso de una distancia conocida como *distancia de edición* o *distancia de Levenshtein* (Wagner and Fischer, 1974), denominada así en honor al ruso Vladimir Levenshtein quien definió esta distancia en 1965 aunque no se publicó en inglés hasta 1966 (Levenshtein, 1966).

Para simular el funcionamiento de un corrector ortográfico, se ha empleado el diccionario de inglés citado al inicio del capítulo. Como conjunto de prototipos se han empleado subconjuntos del diccionario que contenían distinto número de palabras, variando este número entre 2000 y 30000. Para construir estos diccionarios se extrajeron aleatoriamente palabras del diccionario original. Cada conjunto de palabras que se ha utilizado como diccionario tiene asociado su conjunto de muestras de test, que consiste en 1000 palabras sobre las que se han realizado aleatoriamente operaciones de inserción, borrado o sustitución de caracteres. Cada punto de las gráficas representa la media de 10 experimentos con un intervalo de confianza de como máximo 2 desviaciones estándar [3].

Se ha comenzado estudiando la profundidad de los árboles que se obtienen según la técnica de construcción de árbol empleada (figura 6.6). Tal y como ocurre en los experimentos con las bases de datos anteriores, los árboles PMF y PMM coinciden en profundidad y resultan los árboles más profundos, siendo los árboles SFF los menos profundos. En cuanto a los árboles SMF y PRF, en estos experimentos, resultan más profundos los árboles PRF, mientras que con la base de datos de fonemas eran los árboles SMF. Esto se puede deber al tamaño del conjunto de prototipos con el que se han realizado los experimentos. Mientras que en la base de datos de fonemas el mayor tamaño considerado es 4000, los conjuntos de prototipos para el diccionario de inglés tienen un tamaño mínimo de 10000 prototipos. Por la forma de construir los árboles PRF, su incremento de profundidad frente a los árboles SFF se debe a la pérdida de equilibrio provocada al mantener en una rama el

[3] Publicación derivada de este estudio (Gómez-Ballester et al., 2005)

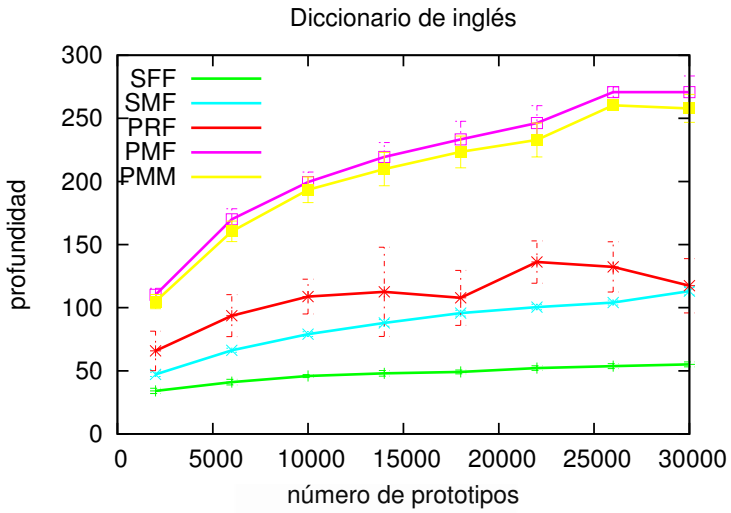


Figura 6.6: Profundidad media de los árboles para distinto tamaño del conjunto de prototipos usando el diccionario de inglés.

mismo representante, mientras en la otra se quedan los prototipos *marginales*. Al ser un conjunto de prototipos reducido y con una inicialización aleatoria del nodo raíz, el desequilibrio de prototipos en la parte superior del árbol puede no ser lo suficientemente grande (figura 6.3) como para superar el desequilibrio que la definición de los árboles SMF conlleva, al basarse en la mediana y su prototipo más alejado.

A continuación se han analizado las distancias calculadas durante la búsqueda del vecino más cercano en cada tipo de árbol. En la figura 6.7 se puede observar cómo las distancias calculadas utilizando los árboles PMF y PMM son muy inferiores en número a las que se calculan con el resto de árboles. La utilización de estos árboles supone un ahorro de un 25 % respecto a los árboles PRF. Y los árboles PRF suponen a su vez un ahorro en el cálculo de distancias de más del 80 % respecto a los árboles SAT y VPT, y de más de un 60 % respecto al árbol SMF. La misma situación se observa en cuanto al número de nodos visitados, aunque en este caso los porcentajes de ahorro de distancias son inferiores.

Los resultados para los árboles SMM no se incluyen en las gráficas ya que, además del coste temporal que implica la construcción del árbol, los resultados obtenidos para algunos tamaños del conjunto de prototipos muestran que calcula un número muy elevado de distancias respecto a los anteriores. En la tabla 6.1 se muestra el número de distancias calculadas para varios tamaños del conjunto de prototipos empleando árboles SMM. Se puede observar cómo

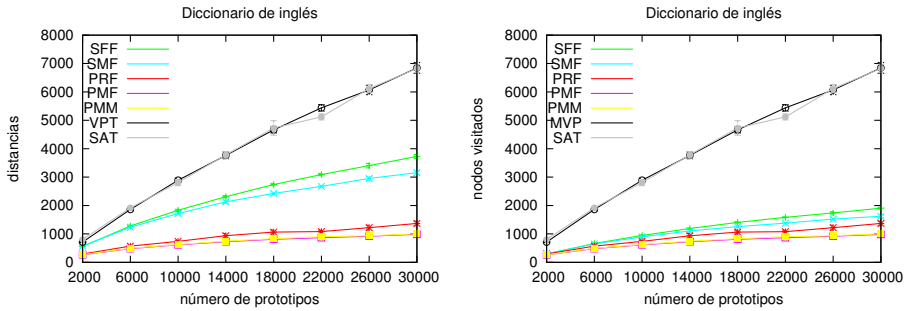


Figura 6.7: Número medio de distancias calculadas y de nodos visitados en la tarea del corrector ortográfico en un diccionario de inglés y usando la regla de eliminación FNR, para las distintas propuestas de construcción del árbol.

Num prototipos	SMM	PRF	SFF
2000	1272	292	556
6000	3800	571	1281
9000	6366	740	1832
14000	9326	937	2311
18000	11654	1065	2739

Tabla 6.1: Número de distancias calculadas empleando los árboles SMM, PRF y SFF para distintos tamaños del conjunto de prototipos en la tarea del corrector ortográfico en un diccionario de inglés.

calculan, en general, el triple de distancias que los árboles SFF, que son los árboles que entre los propuestos calculan el mayor número de distancias. Estas diferencias aumentan considerablemente si se comparan, por ejemplo, con los árboles PRF, respecto a los que llega a calcular 10 veces más distancias.

6.1.1. Experimentos con las reglas de eliminación

Utilizando la construcción del árbol PRF, se ha estudiado la incidencia que tiene el uso de las reglas de eliminación sobre el número de nodos visitados y el número de distancias calculadas.

Experimentos con la distancia de euclídea

Los experimentos de esta sección han sido realizados con las bases de datos de fonemas y de imágenes de satélite. La figuras 6.8 y 6.9 muestran los

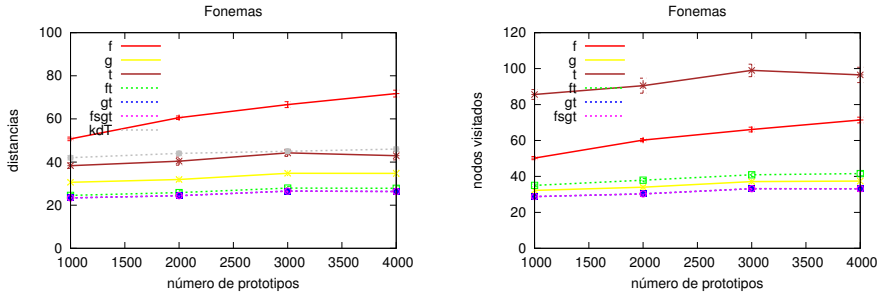


Figura 6.8: Número medio de distancias calculadas y de nodos visitados durante la búsqueda en la base de datos de los fonemas, utilizando un árbol PRF y diferentes combinaciones de reglas de eliminación.

resultados obtenidos. En estos experimentos las reglas de eliminación presentan un comportamiento similar al que han presentado en los experimentos con datos sintéticos para prototipos de dimensión baja. Una de las diferencias más notorias se produce con la regla de la tabla TR. Como se puede observar, la aplicación de la regla TR hace que el número de distancias calculadas durante la búsqueda sea superior al que se calcula con las otras reglas propuestas. No obstante, la regla TR reduce ligeramente las distancia calculadas en la base de datos de imágenes de satélite respecto a la regla FNR y, en casi en un 50 %, en la base de datos de fonemas.

Como ocurre en los experimentos con datos sintéticos, la combinación de reglas FNR y TR o GR y TR (ft o gt en la gráfica) implica calcular el mismo número de distancias que la combinación de todas las reglas de eliminación. Empleando las reglas individualmente, la que produce un mayor ahorro tanto en el número de distancias calculadas como en el de nodos visitados es la regla GR.

La combinación de reglas supone una disminución muy relevante del número de distancias a calcular. Para datos de dimensión alta los experimentos demuestran que las construcciones de árbol propuestas mejoran los resultados obtenidos con las otras técnicas. Para datos de dimensión baja el *kd*-tree calcula menos distancias que los árboles propuestos cuando se emplea la regla de eliminación FNR, sin embargo, la combinación de reglas propuestas evita el cálculo de muchas distancias llegando incluso a calcular menos distancias que el *kd*-tree en la base de datos de los fonemas.

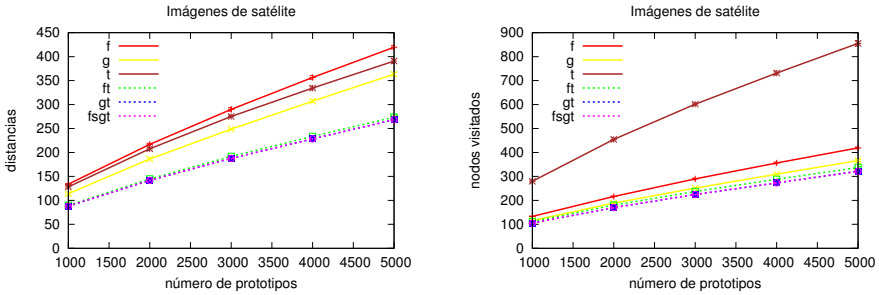


Figura 6.9: Número medio de distancias calculadas y de nodos visitados durante la búsqueda en la base de datos de los imágenes de satélite, utilizando un árbol PRF y diferentes combinaciones de reglas de eliminación.

Experimentos con la distancia de edición

Utilizando la construcción del árbol PRF, se ha estudiado la incidencia que tiene el uso de las reglas de eliminación en la tarea del corrector ortográfico. Al igual que ocurrió con los experimentos realizados con la base de datos de las imágenes de satélite, la regla TR es la que explora más ramas del árbol durante la búsqueda y, por lo tanto, el número de distancias calculadas es mayor que cuando se emplean otras reglas de eliminación (fig. 6.10). En esta figura también se puede observar cómo el número de nodos visitados empleando la regla TR es también superior al que se visita con las otras reglas. La regla de eliminación GR presenta un comportamiento similar al observado con datos sintéticos y su uso supone un ahorro en torno al 50 % respecto a la regla de eliminación FNR, tanto en el número de nodos visitados como en el número de distancias calculadas.

Aunque el uso exclusivo de la regla de eliminación TR no deriva en un ahorro del número de distancias calculadas, la aplicación de esta regla combinada con otras sí supone un ahorro importante en el número de distancias calculadas y de nodos visitados, como se puede observar en la figura 6.10. Se observa que la aplicación de la regla TR combinada con la regla FNR o con la regla GR supone un ahorro de casi un 50 % respecto al número de distancias que se calculan si se aplica sólo la regla FNR o sólo la regla GR. Por otro lado, se puede observar que la regla GR combinada con la regla TR supone el mismo ahorro en el número de distancias calculadas y nodos visitados que si se combinan también con la regla FNR.

En resumen, los experimentos con datos reales también demuestran que los árboles y las reglas de eliminación propuestas reducen notablemente el

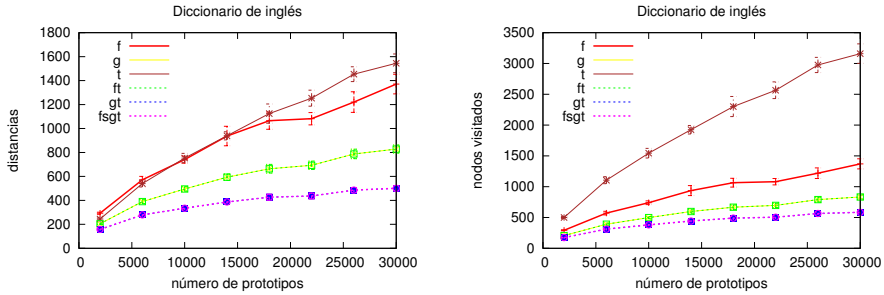


Figura 6.10: Número medio de distancias calculadas y de nodos visitados en la tarea del corrector ortográfico en un diccionario de inglés, utilizando un árbol PRF y diferentes combinaciones de reglas de eliminación.

número de distancias calculadas y de nodos visitados para encontrar el vecino más cercano.

Parte III

Conclusiones y trabajos futuros

Capítulo 7

Conclusiones

En esta tesis se han presentado nuevas formas de construir el índice para la búsqueda por similitud en espacios métricos, así como nuevas reglas de eliminación que evitan la exploración completa del índice. Estas propuestas se han utilizado para la búsqueda del vecino más cercano, dejando pendiente su extensión a otros tipos de búsqueda.

Para los distintos árboles propuestos se han analizado características tales como su profundidad, su solapamiento, y el número de distancias calculadas y nodos visitados en la búsqueda. La idea central en la construcción de los árboles se ha basado en conseguir nodos con poco solapamiento, y se ha proporcionado una definición de solapamiento que se aplica en el estudio de cada uno de los árboles propuestos.

En esta tesis se ha comprobado que el solapamiento de los nodos incide claramente en la búsqueda del vecino más cercano en el árbol y que, en los árboles propuestos que presentan mayor solapamiento, se exploran más ramas del árbol, con el consiguiente coste computacional en número de distancias calculadas y nodos visitados. En general, el uso de los árboles propuestos como índice de la base de datos acelera la búsqueda del vecino más cercano al explorar menos nodos del árbol.

En concreto, en los árboles PRF, PMF y PMM, por su caraterística común de construcción, en los que se reutiliza el representante del nodo padre en el siguiente nivel, permiten, durante la fase de búsqueda, la utilización de distancias ya calculadas y, por lo tanto, no sólo reducen el número de nodos visitados sino que además consiguen un ahorro muy relevante del número de distancias calculadas. Al aplicar estas técnicas en la búsqueda del vecino más cercano en las distintas bases de datos utilizadas, reales y artificiales, los experimentos muestran cómo su uso, empleando únicamente la regla de eliminación de Fukunaga y Narendra (FNR), supone una aceleración con-

siderable del proceso, aún sin utilizar las nuevas reglas de eliminación que hemos propuesto en esta tesis. Por ejemplo, en el caso de que se utilizase en un corrector ortográfico con el diccionario de inglés empleado en esta tesis, los experimentos muestran que mientras técnicas como SAT o VPT consultan entre un 25 % y un 30 % de la base de datos para encontrar el vecino más cercano, el árbol PMM no llega a calcular un 5 % de las distancias a los prototipos de la base de datos.

Los árboles propuestos se pueden utilizar con prototipos de cualquier naturaleza y se puede emplear cualquier función de disimilitud para trabajar con ellos.

Además, se han propuesto algunas reglas de eliminación para que, una vez construido el árbol, se evite la exploración de algunas ramas del árbol durante la fase búsqueda. Los experimentos muestran cómo el uso de estas reglas acelera la búsqueda. A excepción de una de las reglas propuestas, la *regla basada en el nodo hermano* (SBR), cuyo uso por sí sola no sirve para evitar el recorrido por muchos nodos del árbol, la aplicación independiente de las otras dos reglas propuestas repercute en una disminución, en muchos casos drástica, del número de distancias calculadas y de nodos visitados.

La regla de eliminación que presenta un comportamiento más homogéneo con independencia del tipo de árbol y de la base de datos con la que se trabaje es la *regla generalizada* (GR), y su uso supone entre un 30 % y un 40 % menos de nodos a visitar y de distancias a calcular frente a la regla FNR, lo que supone una aceleración notable del proceso de búsqueda. La *regla de la tabla* (TR), usada de manera independiente, en general, también consigue reducir considerablemente el número de distancias a calcular. Pero el uso de esta regla, aunque es muy beneficioso para acelerar la búsqueda, presenta una desventaja importante, su alto coste espacial, que es cuadrático con el número de nodos en el árbol. Para reducir el coste espacial que conlleva el uso de la regla TR, se plantea la estimación de un parámetro en función de la distancia esperada al vecino más cercano, que consigue utilizar la capacidad de eliminación de esta regla reduciendo hasta en un 90 % su ocupación espacial.

Además, se ha propuesto un algoritmo para combinar el uso de las reglas de eliminación propuestas de modo óptimo. Los experimentos también muestran que el uso de este algoritmo combina de manera eficaz el buen funcionamiento individual de las reglas consiguiendo acelerar el proceso.

Por último, se ha desarrollado un conjunto de experimentos con datos reales con los que se ha podido comprobar el buen funcionamiento de las reglas de eliminación propuestas.

Las publicaciones más relevantes derivadas de esta tesis han sido:

- Gómez-Ballester, E.; Micó, L.; Thollard, F.; Oncina, J.; Moreno-Seco, F. Combining Elimination Rules in Tree-Based Nearest Neighbor Search Algorithms. Lecture Notes in Computer Science, pp. 80–89, Cesme, Turkey (S+SSPR 2010).

- Gómez-Ballester, E; Micó, L.; Oncina, J. A pruning Rule Based on a Distance Sparse Table for Hierarchical Similarity Search Algorithms. Lecture Notes in Computer Science, vol. 5342, pp. 936–946 (S+SSPR 2008).

- Oncina J., Thollard F., Gómez-Ballester E., Micó L., Moreno-Seco F. A tabular pruning rule in tree-based pruning rule fast nearest neighbour search algorithms. Lecture Notes in Computer Science, vol. 4478, pp. 306–313 (ibPRIA 2007).

- Gómez-Ballester E.; Micó L.; Oncina J. Some approaches to improve tree-based nearest neighbour search algorithms. Pattern Recognition, vol. 39, pp. 171-179 (2006).

- Gómez-Ballester, E.; Micó, L.; Oncina, J. Testing Some Improvements of the Fukunaga and Narendra's Fast Nearest Neighbour Search Algorithm in a Spelling Task. Lecture Notes in Computer Science, vol. 3523, pp. 3–10 (ibPRIA 2005).

- Gómez-Ballester, E.; Micó, L.; Oncina, J. Some Improvements in Tree Based Nearest Neighbour Search Algorithms. Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence, pp. 456-463 (CIARP 2003).

Otras publicaciones:

- Gómez-Ballester, E.; Micó, L.; Oncina, J. A fast approximated k-median algorithm. Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence, vol. 2396, pp. 684-690 (S+SSPR 2002).

- Gómez-Ballester, E.; Forcada-Zubizarreta, M.L.; Micó-Andrés, M.L. A gradient-descent method to adapt the edit-distance to a classification task. Pattern Recognition and Applications. Frontiers in Artificial Intelligence and Applications (2000).

- Gómez-Ballester, E.; Forcada-Zubizarreta. Mikel L.; Micó, L. A gradient-descent method to adapt the edit distance to a classification task. Pattern Recognition and Image Analysis, Proceedings of the VIII Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes (1999).
- Gómez-Ballester, E.; Micó-Andrés, M.L.; Oncina, J.; Forcada-Zubizarreta, M.L. An empirical method to improve edit-distance parameters for a nearest-neighbor-based classification task'. VII Spanish Symposium on Pattern Recognition and Image Analysis, Barcelona, Spain (1997).
- Gómez-Ballester, E.; Micó-Andrés, M.L.; Oncina, J. Testing the linear approximating and eliminating search algorithm in handwritten character recognition tasks. VI Spanish Symposium on Pattern Recognition and Image Analysis, Córdoba, Spain (1995).

7.1. Trabajo futuro

Los métodos de construcción de árbol y las reglas de eliminación propuestas en esta tesis se pueden aplicar a muchos problemas de búsqueda por similitud. Esto hace que sea interesante continuar profundizando en este estudio, analizando algunos aspectos que han quedado abiertos durante la investigación:

- La principal desventaja de los métodos propuestos es que se han planteado de manera que se obtienen árboles estáticos, en los que la base de datos de los prototipos se utiliza inicialmente para construir el árbol y, en caso de que hubiesen modificaciones en la misma, habría que volver a construir el índice. Un tema de relevancia práctica es cómo poder aplicar estas ideas de construcción del árbol sobre una estructura dinámica, que mantenga las bondades que se han observado en los árboles propuestos.
- Los árboles y reglas propuestos en esta tesis se utilizan para la búsqueda del vecino más cercano y, parece interesante estudiar su uso en otros tipos de consulta.
- Hay que continuar con el estudio del solapamiento en los nodos del árbol. Se podría plantear un modo de particionamiento que divida los prototipos en base a una minimización de su solapamiento, manteniendo otras características con las que se ha comprobado que el resultado es interesante, por ejemplo, la herencia del representante del nodo padre.

- La regla de eliminación basada en el nodo hermano (SBR) es una regla que combinada con otras reglas aumenta la capacidad de eliminación de éstas, sin embargo, utilizada de manera independiente no favorece la eliminación de nodos del árbol y queda pendiente profundizar en el estudio de esta regla.

- Ha quedado pendiente la evaluación de los métodos propuestos en más bases de datos y empleando otros tipos de distancias. Por ejemplo, durante el desarrollo de la tesis se planteó el uso de la distancia de Haversine para medir cercanías entre ciudades representadas por sus coordenadas geográficas. Sin embargo, la distancia de Haversine no cumple la desigualdad triangular, pero se podrían utilizar estas coordenadas para trabajar con la distancia coseno y evaluar si los resultados son similares a los obtenidos con la distancia euclídea u otras métricas. El uso de otras representaciones de objetos como árboles o grafos, también nos darán una percepción de lo importante que es la reducción del número de distancias respecto al propio coste temporal del algoritmo, ya que el coste de la distancia de edición es bastante elevado en estos casos.

- El tipo de información que se utiliza y almacena en la construcción del árbol (radio y representante del nodo) hace que sea fácilmente adaptable a aplicar en otros algoritmos de búsqueda de otros autores. Además, por el mismo motivo, y con independencia o no, también podrán utilizarse las nuevas reglas de eliminación añadiendo a los nodos la información necesaria para su aplicación.

- Por último, en algunos trabajos preliminares con bases de datos reales de muy alta dimensionalidad (normalmente asociadas a imágenes) hemos podido constatar que estas técnicas tienen un futuro prometedor, incluso comparándolas con otras en los que sí se utiliza la representación vectorial de los datos para construir el índice (estructuras tipo *kd*-tree).

Bibliografía

A.V. Aho, Hopcroft J. E., and J. D. Ullman. *Estructuras de Datos y Algoritmos*. Addison-Wesley Iberoamericana, 1988.

G. Amato. *Approximate similarity search in metric spaces*. PhD thesis, Universität Dortmund am Fachbereich Informatik, 2002.

A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/ml`.

V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios. Nearest neighbor retrieval using distance-based hashing. In *Proceedings of the IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 327–336. IEEE Computer Society, 2008.

E. Baydal, G. Andreu, H. Rulot, and E. Vidal. Dimensionalidad intrínseca de un conjunto de palabras aisladas. *Qüestiió: Quaderns d'Estadística, Sistemes, Informatica i Investigació Operativa*, 11(1), 1987.

C.D. Bei and R.M. Gray. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communications*, COM-33:1132–1133, 1985.

R.E. Bellman. *Adaptive control processes:a guided tour*. Princeton University Press, 1961.

R.E. Bellman and Rand Corporation. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957.

J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

J. L. Bentley. K-d trees for semidynamic point sets. In *Proceedings of the sixth annual symposium on computational geometry*, SCG '90, pages 187–197, New York, USA, 1990. ACM.

K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? In *International Conference on Database Theory*, pages 217–235, 1999.

H. Bock. Clustering methods: A history of k-means algorithms. In *Selected Contributions in Data Analysis and Classification*, pages 161–172. Springer Berlin Heidelberg, 2007.

C. Böhm, S. Berchtold, and D. Keim. Searching in high-dimensional spaces: index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33:322–373, 2001.

G.E. Box, G.M. Jenkins, and G.C. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley, 2008.

T. Bozkaya and M. Özsoyoglu. Distance-based indexing for high dimensional metric spaces. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 26, pages 357–368, 1997.

T. Bozkaya and M. Özsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems*, 24:361–404, 1999.

S. Brin. Nearest neighbor search in large metric spaces. *Proceedings of the 21st Very Large Database (VLDB) Conference*, pages 574–584, 1995.

W. A. Burkhand and R.M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, pages 230–236, 1973.

B. Bustos. *Index Structures for Similarity Search in Multimedia Databases*. PhD thesis, Universität Konstanz, 2006.

B. Bustos. Búsqueda por similitud en bases de datos multimedia (material docente)., 2009.

F. Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36:2945–2954, 2003.

N. R. Campbell. *Physics the elements*. University Press, 1920.

E. Chávez, J.L. Marroquin, and R. Baeza-Yates. Spaghettis: an array based algorithm for similarity queries in metric spaces. In *String Processing and Information Retrieval Symposium and International Workshop on Groupware*, pages 38 –46, 1999.

E. Chávez, G. Navarro, R. Baeza-Yates, and J.L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 426–435. Morgan Kaufmann Publishers, Inc., 1997.

K. L. Clarkson. Nearest-neighbor searching and metric space dimensions. In *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006.

E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13:377–387, 1970.

T. M. Covert and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory IT-13*, 12(1):21–27, 1967.

F. K. H. A. Dehne and H. Noltemeier. Voronoi trees and clustering problems. *Information Systems*, 12:171–175, 1987.

A. Duch, V. Estivill-Castro, and C. Martínez. Randomized k-dimensional binary search trees. In *Proceedings of the Ninth Annual International Symposium on Algorithms and Computation*, ISAAC '98, pages 199–209. Springer Verlag, 1998.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, New York, 2000.

M. Dunlop and A. Crossan. Predictive text entry methods for mobile phones. *Personal Technologies*, 4:134–143, 2000.

ELENA. European esprit 5516 project. phoneme dataset, 2004.

K. Figueroa. *Indexación Efectiva de Espacios Métricos usando Permutaciones*. PhD thesis, Universidad de Chile, 2007.

K. Figueroa, G. Navarro, and E. Chávez. Metric spaces library, 2007. Available at `http://www.sisap.org/Metric_Space_Library.html`.

R.F.S. Filho, A. Traina, Jr. Traina, C., and C. Faloutsos. Similarity search without tears: the omni-family of all-purpose access methods. In *Proceedings of the 17th International Conference on Data Engineering*, pages 623 –630, 2001.

R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

E. Fix and J. Hodges. Discriminatory analysis, nonparametric discrimination: consistency properties. *Tech. Rep. 4, USAF School of Aviation Medicine, Randolph Field, Texas*, 1951.

M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884 - 1940)*, 22(1):1–72, 1906.

J.H. Friedman, F. Baskett, and L.J. Shusket. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24:1000–1006, 1975.

K.S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, 1982.

K. Fukunaga and P.M. Narendra. A branch and bound algorithm for computing k-nearest neighbors. *IEEE Transactions on Computers, IEC*, 24: 750–753, 1975.

V. Gaede and O. Gunther. Multidimensional access methods. *ACM Computing Surveys*, 30:170–231, 1998.

A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th Very Large Database (VLDB) Conference*, pages 518–529, 1999.

R. L. Goldstone and J. Son. Similarity. In *Handbook of Thinking and Reasoning*, pages 13–36. Cambridge University Press, 2005.

Eva Gómez-Ballester, Luisa Micó, and José Oncina. Some improvements in tree based nearest neighbour search algorithms. In *Proceedings of the 8th Iberoamerican Congress on Pattern Recognition*, pages 456–463, 2003.

Eva Gómez-Ballester, Luisa Micó, and José Oncina. Testing some improvements of the fukunaga and narendra's fast nearest neighbour search algorithm in a spelling task. In *Proceedings of the Second Iberian Conference on Pattern Recognition and Image Analysis, IbPRIA 2005*, pages 3–10, 2005.

Eva Gómez-Ballester, Luisa Micó, and José Oncina. Some approaches to improve tree-based nearest neighbour search algorithms. *Pattern Recognition*, 39(2):171–179, 2006.

Eva Gómez-Ballester, Luisa Micó, and José Oncina. A pruning rule based on a distance sparse table for hierarchical similarity search algorithms. In *Proceedings of the Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR & SPR 2008*, pages 926–936, 2008.

Eva Gómez-Ballester, Luisa Micó, Franck Thollard, José Oncina, and Francisco Moreno-Seco. Combining elimination rules in tree-based nearest neighbor search algorithms. In *Proceedings of the Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR&SPR*, pages 80–89, 2010.

R. Guigó. Bioinformática: la creciente interconexión entre biología y computación. *Boletín electrónico de la Sociedad Española de Genética, 17*, pages 4 – 8, 2003.

R. Guigó and D. Gusfield. *Algorithms in Bioinformatics*, volume 2452 of *Lecture Notes in Computer Science*. Springer, 2002.

D. Gunopulos and G. Das. Time series similarity measures (tutorial pm-2). In *Tutorial notes of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 243 – 307. ACM New York, USA, 2000.

T.L. Heath and T.L. Heath. *The thirteen books of Euclid's Elements*. The Thirteen Books of Euclid's Elements. Dover Publications, 1956.

G. R. Hjaltason and H. Samet. Index-driven search in metric spaces. *ACM Transactions on Database Systems*, 28(4):517–580, 2003.

M. E. Houle, H. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Can shared-neighbor distances defeat the curse of dimensionality? In *Scientific and Statistical Database Management Conference (SSDBM'10)*, pages 482–500, 2010.

Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, 2010.

K. Janowicz. *Computing Semantic Similarity Among Geographic Feature Types Represented in Expressive Description Logic*. PhD thesis, Institute for Geoinformatics (IFGI), Universitity of Münster, 2008.

K. Janowicz and M. Wilkes. Sim-dla: A novel semantic similarity measure for description logics reducing inter-concept to inter-instance similarity. In *The 6th Annual European Semantic Web Conference (ESWC2009)*, pages 353 – 367. Lecture Notes in Computer Science 5554, Springer, 2009.

K. Janowicz, C. KeBler, M. Schwarz, M. Wilkes, I. Panov, M. Espeter, and B. Baeumer. Algorithm, implementation and application of the sim-dl similarity server. In *Second International Conference on GeoSpatial Seman-*

tics (GeoS 2007), pages 128 – 145. Lecture Notes in Computer Science, Springer, 2007.

K. Janowicz, M. Raubal, A Schwering, and W. Kuhn. Semantic similarity measurement and geospatial applications (editorial). *Special Issue on Semantic Similarity Measurement and Geospatial Applications.*, 12:651–659, 2008.

E. Jurado. *Una aproximación eficiente para la búsqueda de vecinos en espacios multidimensionales a través del árbol Q*. PhD thesis, Universidad de Extremadura, 2003.

I. Kalantari and G. McDonald. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 9: 631–634, 1983.

L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An introduction to cluster analysis*. John Wiley, New York, 1990.

A.M. Kibriya. *Fast Algorithms for Nearest Neighbour Search*. PhD thesis, The University of Waikato, 2007.

D. E. Knuth. *El arte de programar ordenadores: Clasificación y búsqueda*. Editorial Reverté, S.A., 1987.

A. N. Kolmogorov and S. V. Fomin. *Elements of the Theory of Functions and Functional Analysis, vol. I: Metric and Normed Spaces*. Izdact Moscow University, Moscow, 1954. in Russian; (English traslation, Graylock Press, Rochester, N.Y., 1957).

VI. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

K. Lin, H. V. Jagadish, and C. Faloutsos. The tv-tree: an index structure for high-dimensional data. *The Very Large Database Journal*, 3(4):517–542, 1994.

Z. Lin, Y. Huang, H. Wang, and S. McClean. Neighborhood counting for financial time series forecasting. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pages 815–821, 2009.

J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281 – 297. University of California Press, 1967.

L. Micó, J. Oncina, and E. Vidal. An algorithm for finding nearest neighbors in constant average time with a linear space complexity. In *Proceedings of the 11th International Conference on Pattern Recognition (ICPR'92)*, volume II, pages 557–560, 1992.

L. Micó, J. Oncina, and E. Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing-time and memory requirements. *Pattern Recognition Letters*, 15:9–17, 1994.

L. Micó, J. Oncina, and R.C. Carrasco. A fast branch and bound nearest neighbour classifier in metric spaces. *Pattern Recognition Letters*, 17:731–739, 1996.

N. Moënne-Loccoz. High-dimensional access methods for efficient similarity queries. Technical report, Computing Science Center, University of Geneva, 2005.

F. Moreno-Seco. *Clasificadores eficaces basados en algoritmos rápidos de búsqueda del vecino más cercano*. PhD thesis, Universidad de Alicante, 2004.

D. W. Mount. *Bioinformatics. Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2004.

K. Najarian and R. Splinter. *Biomedical Signal and Image Processing*. Biomedical engineering. CRC Press of Taylor & Francis Group, USA, 2012.

G. Navarro. Searching in metric spaces by spatial approximation. In *Proceedings of the 6th International Symposium on String Processing and Information Retrieval*, pages 141–148. IEEE Computer Society, 1999.

G. Navarro. Searching in metric spaces by spatial approximation. *The Very Large Database Journal*, 11(1):28–46, 2002.

S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48:443–453, 1970.

T. Negi and V. Bansal. Time series: similarity search and its applications. In *International Conference on Systemics, Cybernetics and Informatics: ICSCI-04*, pages 528–533, 2005.

H. Noltemeier, T. Roos, and C. Zirkelbach. Partitioning of complex scenes of geometric objects. In *System Modelling and Optimization*, volume 180 of

Lecture Notes in Control and Information Sciences, pages 94–101. Springer Berlin / Heidelberg, 1992.

José Oncina, Franck Thollard, Eva Gómez-Ballester, Luisa Micó, and Francisco Moreno-Seco. A tabular pruning rule in tree-based fast nearest neighbor search algorithms. In *Proceedings of the Third Iberian conference on Pattern Recognition and Image Analysis, IbPRIA 2007, Girona, Spain*, pages 306–313, 2007.

I. Paromtchik. Steering and velocity commands for parking assistance. In *Proceedings of the 10th IASTED Conference on Robotics and Applications*, pages 178–183, 2004.

M. Patella and P. Ciaccia. The many facets of approximate similarity search. In *Proceedings of the First International Workshop on Similarity Search and Applications*, pages 10–21, 2008.

D. Paulus and J. Hornegger. *Applied Pattern Recognition.* Vieweg, 1998.

K. W. Pettis, T. A. Bailey, A. K. Jain, and R. C. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):25–37, 1979.

W. V. Quine. *Ontological Relativity and Other Essays.* University Press, New York, Columbia, 1969.

M. Raubal. Formalizing conceptual spaces. In *Proceedings of the Third International Conference on Formal Ontology in Information Systems (FOIS 2004)*, volume 114, pages 153 – 164. IOS Press, 2004.

A. Rimrott and T. Heift. Evaluating automatic detection of misspellings in german. *Language Learning & Technology*, 12(3):73–92, 2011.

A. Rodríguez and M. Egenhofer. Comparing geospatial entity classes: an asymmetric and context-dependent similarity measure. *International Journal of Geographical Information Science*, 18:229–256, 2004.

H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, USA, 1986.

H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers, Inc., 2006.

C. W. Savage. *The measurement of sensation: a critique of perceptual psychophysics*. University of California Press, 1970.

A. Schwering. Approaches to semantic similarity measurement for geo-spatial data: A survey. *Transactions in Geographical Information Science*, 12:5–29, 2008.

A. Scwering and M. Raubal. Spatial relations for semantic similarity measurement. In *Proceedings of the 24th International Conference on Perspectives in Conceptual Modeling*, pages 259 – 269. Springer-Verlag, 2005.

G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006.

C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR08)*, pages 1–8, 2008.

P. Somboonsak and M.A. Munlin. A new edit distance method for finding similarity in dna sequence. *World Academic of Science Engineering and Technology*, 58:895–899, 2011.

S. S. Stevens. On the theory of scales of measurement. *Science*, 103(2684): 677–680, 1946.

Smith T. and Waterman M. Identification of common molecular subsequences. *Journal of molecular biology*, 147:195–197, 1981.

S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, San Diego, 2009. 4th Edition.

C. Traina, A. Traina, B. Seeger, and C. Faloutsos. Slim-trees: High performance metric trees minimizing overlap between nodes. In *7th International Conference on Extending Database Technology (EDBT)*, pages 51–65. Springer-Verlag, 2000.

J.K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, 1991.

E. Vidal. An algorithm for finding nearest neighbours in (approximately) constant average time complexity. *Pattern Recognition Letters*, 4:145–157, 1986.

E. Vidal. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (aesa). *Pattern Recognition Letters*, 15:1–7, 1994.

B. Waggoner. *Compression for Great Video and Audio: Master Tips and Common Sense*. Elsevier Science, 2009.

R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

A. Webb. *Statistical Pattern Recognition*. Arnold Publishers, 1999.

P.N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.

P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search. The metric Space Approach*. Springer, New York, 2006.

K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18: 1245–1262, 1989.

X. Zhu and I. Davidson. *Knowledge Discovery and Data Mining: Challenges and Realities*. Idea Group Inc IGI, USA, 2007.