Uso eficiente de pivotes aplicado a la búsqueda aproximada en algoritmos rápidos sobre espacios métricos

Raisa Socorro Llanes[1] y Luisa Micó Andrés[2]

[1] Instituto Superior Politécnico José Antonio Echeverría, CUJAE, Cuba

[2] Universidad de Alicante, España

raisa@ceis.cujae.edu.cu mico@dlsi.ua.es

Resumen. El contexto de este trabajo es la búsqueda rápida de vecinos más cercanos en espacios métricos. Uno de los objetivos de estos algoritmos es la reducción del tiempo de respuesta durante la búsqueda. Reducir el tiempo de respuesta consiste muchas veces en reducir el número de distancias a calcular, debido al alto coste computacional que de por sí pueden tener las distancias a utilizar en ciertas aplicaciones. Nosotros proponemos una nueva versión y mejoras de un algoritmo recientemente publicado, PiAESA, variante del algoritmo AESA, usado como referencia en este área por sus buenos resultados desde hace más de 20 años. La nueva versión es más simple y permite una mejor comprensión del algoritmo y sus parámetros. Además, se ha conseguido aumentar la eficiencia definiendo una versión aproximada. Los resultados empíricos obtenidos utilizando datos artificiales y reales confirman una mejora en los resultados de la versión aproximada, con un alto porcentaje de una respuesta correcta (dada por un algoritmo exacto).

Palabras clave. Búsqueda aproximada, espacios métricos, vecino más cercano, distancias.

Efficient use of Pivots for Approximate Search in Metric Spaces

Abstract. This work focuses on pivot-based fast nearest neighbor search algorithms that can work in any metric space. One of the objectives of these algorithms is to reduce the time consumed during search. Reducing time consumption of such algorithms usually consists in reducing the number of distances for computing, due to the high cost that they have in certain applications. We introduce a new version and improvements for a recently proposed algorithm, *PiAESA*, a variant of the *AESA* algorithm, used as baseline for performance measurement for over twenty years. The new version is simpler and allows better understanding of the algorithm and parameters used. Moreover, the efficiency is increased by defining an approximated version. Our empirical results with real and artificial databases confirm a consistent improvement in performance, when retrieving very high percentage of the correct answers (given by the exact algorithm).

Keywords. Approximate search, metric spaces, near neighbor, distances.

1 Introducción

La búsqueda por similitud (o vecindad) es una técnica utilizada en aquellos casos en que el objeto de consulta no se encuentra exactamente en la base de datos, por lo tanto es un fenómeno muy frecuente en aplicaciones de Reconocimiento de Patrones [17], Recuperación de Imágenes [4], Bases de Datos Multimedia [12, 1],o Bioinformática [13] entre otras. Este tipo de búsqueda consiste en recuperar de la base de datos los objetos más similares a la consulta. La regla del vecino más cercano (*vmc*) es utilizada en esta tarea por su simplicidad y eficacia. El algoritmo más sencillo para su implementación es el llamado búsqueda exhaustiva o fuerza bruta, que calcula la distancia de todos los elementos de la base de datos a la consulta, seleccionando el de menor distancia como el *vmc*. Sin embargo, cuando el tamaño de las bases de datos es grande, o la función de distancia es muy costosa, no es recomendable el uso de este algoritmo tan simple.

Computación y Sistemas Vol. 17 No. 4, 2013, pp. 477-488
ISSN 1405-5546
DOI:10.13053/CyS-17-4-2013-002

El desarrollo de algoritmos eficientes para la búsqueda del *vmc* es un tema de investigación vigente desde hace muchos años. El objetivo de estos algoritmos es el de reducir el número de distancias a calcular cuando el coste de la distancias es muy elevado y/o evitar recorrer exhaustivamente toda la base de datos para encontrar la solución. Entre los algoritmos existentes podemos encontrar algunas propuestas cuya solución solo es válida para una representación vectorial de los datos y su eficiencia se basa justamente en aprovechar las características propias de esta representación. Sin embargo, existen otros algoritmos aplicables a espacios métricos en general cuya solución incluye el caso anterior y otros en los que resulta imposible transformar los datos a una representación vectorial.

Un espacio métrico está constituido por un conjunto de datos (por ejemplo: caras, secuencias de audio, secuencias de ADN, etc.) y una función de distancia que evalúa la semejanza entre dos elementos cualesquiera. En los espacios métricos la función de distancia debe cumplir las siguientes propiedades: positividad, reflexividad, simetría y la desigualdad triangular.

Diferentes métodos de indexación basados en las propiedades de los espacios métricos se han desarrollado para resolver de manera eficiente la búsqueda por similitud. En la literatura podemos encontrar algunas compilaciones de estos algoritmos, por ejemplo en [9, 7, 22]. Estos métodos normalmente se clasifican en 1) basados en pivotes (que almacenan distancias precalculadas para descartar objetos), o 2) algoritmos de agrupamiento (basados en la partición del espacio en grupos para descartar algunos de ellos durante la búsqueda).

Vidal propone en [21] el algoritmo de búsqueda por aproximación y eliminación, *AESA* (*Approximating and Eliminating Search Algorithm*), clasificado por Chávez en [7] como un algoritmo de búsqueda basado en pivotes. Los pivotes son un subconjunto de los objetos en la base de datos que se utilizan en la construcción del índice para acelerar la búsqueda. Por lo general, las distancias de cada pivote a algunos (o todos) los objetos en la base de datos se calculan y almacenan en la etapa de preprocesamiento y se utilizan durante la búsqueda. *AESA* ha sido considerado en las últimas dos décadas como referencia para los métodos de búsqueda en espacios métricos en cuanto al número de distancias calculadas [11], que es independiente respecto al tamaño de la base de datos. Para conseguirlo, utiliza una función que representa una cota inferior del valor real de la distancia (cota mínima), de forma que su uso permite reducir el número de distancias a calcular. Recientemente se ha propuesto un nuevo método llamado *PiAESA*, el cual reduce considerablemente el número de distancias calculadas con respecto al *AESA*, sin sobrecargar la búsqueda [3]. La idea de este algoritmo consiste en cambiar el enfoque utilizado por *AESA* para seleccionar los pivotes en las primeras iteraciones durante la fase de aproximación. Este algoritmo utiliza pivotes (previamente ordenados en tiempo de preprocesamiento), que aunque pueden ser malos candidatos a *vcm*, contribuyan a actualizar rápidamente el valor de la cota mínima de la distancia en las primeras iteraciones, y cuando esta es lo suficientemente precisa se cambia a la estrategia de aproximación habitual del *AESA.*

Un problema serio de todos los algoritmos en los espacios métricos, incluso para el *PiAESA*, es que empeoran su rendimiento cuando aumenta la dimensión de los datos, hecho conocido como maldición de la dimensionalidad. En [7] se muestra que el concepto de dimensión de un espacio vectorial se puede extender a espacios métricos arbitrarios, utilizando el concepto de dimensión intrínseca[1] y que la maldición de la dimensionalidad se mantiene en dichos espacios. Esto implica que a medida que aumenta la dimensión intrínseca del espacio métrico, los algoritmos basados en pivotes necesitan aumentar el número de pivotes utilizados para mantener un rendimiento aceptable [7]. En este caso, un algoritmo inexacto (que puede perder algunas de las respuestas pertinentes) es una herramienta práctica. Cualquier algoritmo exacto puede ser convertido en inexacto al relajar las condiciones de la búsqueda. Se trata por tanto de

[1] Expresa que la relación que se establece entre los objetos sólo está basada en sus distancias relativas en vez de la representación de sus coordenadas en el espacio

medir el número de respuestas correctas encontradas.

En este trabajo proponemos una nueva versión del algoritmo *PiAESA* donde se modifica el criterio utilizado para el cambio de la estrategia de selección de pivotes. Esta modificación permite comprender mejor el comportamiento del algoritmo. Además, se propone una variante aproximada del *PiAESA* que permitirá reducir la cantidad de distancias a calcular en bases de datos de dimensionalidad elevada.

2 El algoritmo PiAESA

El algoritmo *AESA* propuesto por Vidal originalmente en [19] (véase Algoritmo 1), construye durante la etapa de preprocesamiento una matriz de distancias *D*, en la que almacena la distancia entre todos los objetos de la base de datos *T*. La etapa de búsqueda cuenta con dos fases: aproximación y eliminación. Durante la búsqueda se selecciona un objeto *s* aproximadamente cercano a la consulta *q*, además calcula la distancia entre ambos y se actualiza el *vmc* si esta distancia es menor que la del *vmc* hasta el momento. En la segunda fase (eliminación), usando la desigualdad triangular, se eliminan para posteriores búsquedas todos los objetos que no pueden estar más cercanos a la muestra que el actualmente seleccionado como tal. El procedimiento termina cuando la base de datos se ha recorrido completamente. En ambas fases se utiliza la cota mínima de la distancia *G*: durante la aproximación se selecciona como candidato a *vmc* un objeto *s* con el menor valor de su cota mínima a la consulta (línea 4, Algoritmo 1), y durante la fase de eliminación se eliminan todos los objetos cuya cota mínima sea mayor a la distancia a la consulta del *vmc* hasta el momento (línea 9, Algoritmo 1).

Un aspecto importante es la actualización de la función G. Esta se calcula utilizando la siguiente ecuación:

$$G(t)=\max_{p\in V}\left(\left|d(q,p)-d(p,t)\right|\right) \quad (1)$$

donde *V* es el conjunto de los *vmc* seleccionados en iteraciones anteriores. Esta función se actualiza en cada iteración, de tal manera que la cota es cada vez más precisa. Es necesario señalar que durante las primeras iteraciones, *V* es muy pequeño ya que empieza siendo un conjunto vacío, y en cada iteración va incrementando su tamaño con el objeto seleccionado como mejor candidato.

Algoritmo 1. *Algoritmo AESA*

Entrada: T base de datos
q consulta
$D \in R^{|T|\times|T|}$ matriz de distancias

Salida: $p_{\min} \in T$ vmc de q

1 $d_{\min} = \infty$;
2 para todo $t \in T$ hacer $G(t) = 0$;
3 mientras $T \neq 0$ hacer
4 $s = \text{argmin}_{u\in T} G(u)$;
5 $T = T - \{s\}$; $d = d(s,q)$;
6 si $d < d_{\min}$ entonces $d_{\min} = s$; $d_{\min} = d$;
7 para todo $t \in T$ hacer
8 $G(t)\max\left(G(t), |D(s,t) - d|\right)$;
9 si $G(t) > d_{\min}$ entonces $T = T - \{s\}$;
10 fin para
11 fin mientras

En [3] se propuso recientemente un nuevo algoritmo llamado *PiAESA* (véase Algoritmo 2, de ahora en adelante *PiAESAc*). La idea de este algoritmo es utilizar un criterio de aproximación diferente al empleado por *AESA* que permita durante las primeras iteraciones del algoritmo emplear como pivotes aquellos objetos que más contribuyen en el incremento de la cota mínima *G* (aunque no sean buenos candidatos a *vmc*). Esto se debe a que en las primeras iteraciones de AESA, como se ha comentado anteriormente, el tamaño de *V* es muy pequeño, por lo que la cota calculada no estima adecuadamente la distancia al *vmc* y se calculan más distancias de las

necesarias. Cuando se considera que *G* está satisfactoriamente actualizada, se cambia al criterio de aproximación del *AESA*. Para poder elegir los objetos (pivotes) que más contribuyen al incremento de *G* durante la fase de preprocesamiento, además de calcular la matriz de distancias *D*, se ordenan los objetos de la base de datos en una lista ordenada *P* por su contribución al aumento de la cota mínima *G*. En la etapa de búsqueda, para realizar la fase de aproximación, esta lista ordenada de pivotes *P* es utilizada en las primeras iteraciones.

Algoritmo 2. *Algoritmo PiAESAc*

Entrada: *T*: base de datos
q: consulta
$D \in R^{|T| \times |T|}$ matriz de distancias
$P \subseteq T$ lista ordenada de pivotes
$c \in N$ controla el cambio de criterio de aproximación

Salida: $p_{\min} \in T$ *vmc* de *q*

1 $d_{\min} = \infty$;
2 para todo $t \in T$ hacer $G(t) = 0$;
3 $i = g_{\min} = g_{prev\min} = 0$;
4 mientras $P \neq 0$ and $i < c$ hacer
5 $s = pop(P)$; $i = i + 1$;
6 $T = T - \{s\}$; $d = d(s,q)$;
7 si $d < d_{\min}$ entonces $p_{\min} = s$; $d_{\min} = d$;
8 $g_{prev\min} = g_{\min} = \infty$;
9 para todo $t \in T$ hacer
10 $G(t) = \max(G(t), |D(t,s) - d|)$;
11 $g_{prev\min} = \min(G(t), g_{\min})$;
12 fin para
13 si $g_{\min} \geq g_{prev\min}$ entonces $i = 0$;
14 fin mientras
15 mientras $T \neq 0$ hacer
16 $s = \arg\min_{u \in T} G(u)$;
17 $T = T - \{s\}$; $d = d(s,q)$;
18 si $d < d_{\min}$ entonces $p_{\min} = s$; $d_{\min} = d$;
19 para todo $t \in T$ hacer
20 $G(t) = \max(G(t), |D(t,s) - d|)$;
21 si $G(t) > d_{\min}$ entonces $T = T - \{s\}$;
22 fin para
23 fin mientras

Para cambiar el criterio de aproximación se utiliza un parámetro *c* que calcula el número de iteraciones sucesivas en las que el mejor candidato a *vmc* no ha cambiado (líneas de la 4 a la 14, Algoritmo 2). Cuando esto ocurre, significa que no es necesario realizar la aproximación utilizando la lista ordenada de pivotes *P* porque las cotas ya están suficientemente ajustadas. En este punto, el algoritmo cambia su estrategia y se centra en buscar buenos candidatos para *vmc* pasando al comportamiento habitual del *AESA* (líneas de la 15 a la 23, Algoritmo 2).

Algoritmo 3. *Algoritmo PiAESAn*

Entrada: *T*: base de datos
q: consulta
$D \in R^{|T| \times |T|}$ matriz de distancias
$P \subseteq T$ lista ordenada de pivotes
$n \in N$ número de pivotes a utilizar de *P*

Salida: $p_{\min} \in T$ *vmc* de *q*

1 $d_{\min} = \infty$;
2 para todo $t \in T$ hacer $G(t) = 0$;
3 mientras $P \neq 0$ and $i < n$ hacer
4 $s = pop(P)$; $i = i + 1$;
5 $T = T - \{s\}$; $d = d(s,q)$;
6 si $d < d_{\min}$ entonces $p_{\min} = s$; $d_{\min} = d$;
7 para todo $t \in T$ hacer
8 $G(t) = \max(G(t), |D(t,s) - d|)$;
9 fin para
10 fin mientras
11 mientras $T \neq 0$ hacer
12 $s = \arg\min_{u \in T} G(u)$;
13 $T = T - \{s\}$; $d = d(s,q)$;
14 si $d < d_{\min}$ entonces $p_{\min} = s$; $d_{\min} = d$;
15 para todo $t \in T$ hacer
16 $G(t) = \max(G(t), |D(t,s) - d|)$;
17 si $G(t) > d_{\min}$ entonces $T = T - \{s\}$;
18 fin para
19 fin mientras

Hay que tener en cuenta que cuando el parámetro *c* es igual a 0, este algoritmo se comporta exactamente como el *AESA*, ya que se omite el primer bucle (líneas de la 4 a la 14). Este algoritmo permite utilizar diferentes formas de construir la lista ordenada de pivotes *P*, por lo que se han usado algunas técnicas de selección de pivotes empleados por algoritmos de

búsquedas basados en pivotes con buenos resultados. A continuación se describen brevemente dichas técnicas.

Selección aleatoria de pivotes (RPS): esta técnica es muy simple, los pivotes son seleccionados al azar.

Selección de *outliers*: Este método selecciona los pivotes de forma incremental, de forma que estos se encuentren muy separados entre ellos y también respecto al resto de los prototipos. Se comienza seleccionando el primer pivote al azar (p_1) y se pueden seguir dos estrategias para seleccionar el resto de los pivotes [15]

Máximo del mínimo de la distancia (MMD):

$$p_i = \arg \max_{s \in (T - P_{i-1})} \min_{j=1}^{i-1} d(p_j, s) \quad (2)$$

Máximo de la suma de la distancia (MSD):

$$P_i = \arg \max_{s \in (T - P_i)} \sum_{j=1}^{i-1} d(P_j, S) \quad (3)$$

donde *T* es el conjunto de entrenamiento y $P_{i-1} = \{p_1, p_2, \ldots, p_{i-1}\}$ los pivotes seleccionados hasta el momento. En la lista de pivotes *P* obtenida con esta técnica pueden almacenarse todos los objetos de la base de datos en el mismo orden en que fueron elegidos, al igual que al utilizar el criterio RPS.

Selección espacial de pivotes dispersos (SSS): Este método selecciona dinámicamente un conjunto de pivotes cuya distancia a cualquier pivote ya seleccionado es mayor que un porcentaje de la distancia máxima entre todos los objetos de la base de datos [5]. Si un objeto no cumple con la condición anterior, no es insertado en la lista de pivotes *P* y entonces la lista no puede ser una enumeración de todos los objetos en la base de datos.

Selección dinámica de pivotes (DPS): Esta técnica propuesta en [6] es una extensión del método descrito anteriormente (SSS). En este caso, además de cumplir la condición explicada en SSS, el candidato a pivote no debe ser redundante con respecto a los anteriores pivotes seleccionados, incluso es posible eliminar un pivote ya incluido si se demuestra que es más redundante que el candidato a añadir.

3 Nuestra propuesta del uso de pivotes

El algoritmo *PiAESAc* descrito anteriormente no utiliza un número fijo de pivotes en la primera etapa de aproximación y su eficacia depende del valor del parámetro *c*. En este trabajo, proponemos modificar el criterio utilizado para cambiar el método aproximación. Nuestra propuesta consiste en utilizar una cantidad fija de pivotes en la primera etapa (en lugar de ver las condiciones del criterio anterior, en el que había que calcular el número de iteraciones seguidas en las que el candidato a vmc no se modificaba), lo cual simplifica el proceso de actualización de la cota mínima de la distancia G (líneas 4 a 15 en el Algoritmo 2). En este caso proponemos repetir el proceso de actualización de *G* para los *n* primeros pivotes de la lista ordenada *P*, donde $n \leq |P|$. Este nuevo enfoque simplifica el algoritmo, y permite analizar la influencia de la cantidad de pivotes utilizados en esta etapa en la reducción del número de distancias calculadas para encontrar el *vmc*. El algoritmo resultante de estas modificaciones se presenta en el Algoritmo 3, el cual de ahora en adelante llamaremos *PiAESAn*.

4 Algoritmo PiAESAn aproximado

El algoritmo *PiAESAn* presentado en la sección anterior es un algoritmo exacto, es decir, recupera correctamente el elemento de la base de datos que se encuentra más cercano a la consulta *q* según la función distancia *d* utilizada. Este algoritmo no está exento de la maldición de la dimensionalidad, por lo que una posible solución es convertir el algoritmo exacto en inexacto. Este tipo de algoritmos relaja la precisión de la consulta, lo que permite ganar velocidad en los tiempos de respuesta [18]. Esta aproximación generalmente es razonable en muchas aplicaciones debido a que modelar como espacios métricos ya involucra una aproximación

a la respuesta verdadera y por lo tanto podría ser aceptable una segunda aproximación durante la búsqueda. En este tipo de aproximaciones, adicionalmente a la consulta se especifica un parámetro de precisión que controla cuán lejos (o relajada) se quiere la respuesta de la solución correcta a la consulta. Esta alternativa para la búsqueda por similitud exacta es llamada búsqueda por similitud inexacta, y abarca los algoritmos aproximados y probabilísticos:

- Algoritmos aproximados: en este tipo de búsquedas el usuario define un valor que indica qué tan relajada acepta una solución respecto a la respuesta correcta. Muchos algoritmos exactos usan esta estrategia para conseguir mejorar su comportamiento a cambio de perder precisión [16, 18].
- Algoritmos probabilísticos: otra forma de búsqueda inexacta son los algoritmos probabilísticos donde se especifica la probabilidad de error en la respuesta. La idea es identificar rápidamente los elementos prometedores y a medida que disminuya la calidad de la respuesta dejar de comparar elementos en la base de datos [8].

Figueroa [10] demuestra experimentalmente, al utilizar una variante de búsqueda aproximada basada en el criterio de detención temprana, que al explorar un pequeño porcentaje de la base de datos el algoritmo *iAESA* obtiene un porcentaje de recuperación de respuestas correctas (PR) superior al recuperado por el *AESA* en dimensiones altas.

Nuestra propuesta consiste en obtener una variante aproximada del *PiAESA* empleando un criterio de poda agresiva que nos permita reducir el número de distancias evaluadas manteniendo un alto porcentaje de recuperación. Para esto, se incorpora a los algoritmos propuestos en este trabajo un parámetro llamado *holgura* que permite controlar la precisión de la respuesta [19]. Este parámetro representa la holgura con la que la desigualdad triangular se cumple para el triplete (*x,y,z*) con la medida de distancia utilizada. El valor de la holgura se puede calcular a partir de métodos propuestos en [19] o de forma experimental.

Teniendo en cuenta que el criterio de eliminación utilizado por el *PiAESA* elimina un objeto u de la base de datos cuando se cumple que $G(t) \geq d_{\min}$, siendo $G(t)$ la cota inferior de la distancia de la consulta *q* al objeto *t* y $d_{\min}$ la distancia del objeto más cercano a la consulta hasta el momento, el nuevo criterio de eliminación quedaría como $G(t) \geq d_{\min} - h$, donde *h* es la *holgura* (Algoritmo *PiAESAn* línea 17). Esta modificación es aplicada a todos los algoritmos utilizados en este trabajo.

5 Experimentos

En este trabajo se realizó un conjunto de experimentos con el objetivo de analizar el comportamiento de nuestra propuesta y poder determinar la influencia de la cantidad de pivotes (*n* en el algoritmo 3) utilizados durante la primera etapa del algoritmo *PiAESAn* en la reducción de la cantidad de distancias calculadas en la búsqueda del *vmc*. Además, comprobamos la eficacia del *PiAESAn* aproximado utilizando el criterio descrito en las secciones anteriores, y comparamos los resultados con algunos algoritmos actuales para la búsqueda por similitud basados en pivotes como el *iAESA, AESA y PiAESA*.

En los experimentos hemos utilizado tres tipos de bases de datos:

1. Datos generados con una distribución uniforme en el hipercubo unidad para dimensiones de 2 a 32 obtenidas del sitio http://www.sisap.org[2] . En estos casos se utilizó la distancia de Minkowski L_1 (distancia de Manhattan).
2. Dos bases de datos reales: NASA y COLORS, obtenidas a partir del sitio http://www.sisap.org. Ambas bases de datos son vectores de características extraídos de dos colecciones de imágenes: NASA contiene 40,150 vectores de dimensión 20 obtenidos a partir de archivos de imágenes y vídeos de la NASA, siendo su dimensión intrínseca 8. COLORS contiene 112,682 vectores de dimensión 112 obtenidos a partir del histograma de colores de una base de

[2] Sitio Oficial del International Workshop on Similarity Search and Applications

datos de imágenes, con una dimensionalidad intrínseca de 5. Como en el caso anterior, también se utiliza la distancia de Minkowski L_1 como medida de similitud.

3. Una base de datos de cadenas obtenidas a partir de la extracción de la cadena de contorno a los dígitos en la base de datos MNIST (http://yann.lecun.com/exdb/mnist) y con una dimensionalidad intrínseca de 6. En este caso empleamos la distancia de edición [14] como medida de similitud.

En todos los experimentos realizados se utilizaron 15,000 objetos en el conjunto de entrenamiento y 1,000 objetos para la búsqueda, todos seleccionados aleatoriamente de estas bases de datos. Los resultados presentados son el promedio de estas 1,000 búsquedas.

5.1 Análisis de la influencia del parámetro n y las técnicas de selección de pivotes

En esta sección hemos estudiado el comportamiento del número de pivotes (*n*) utilizados durante la etapa de actualización del la cota inferior de la distancia para las técnicas de selección de pivotes mencionadas en la sección 0. El valor óptimo de *n* se obtiene en tiempo de preproceso.

En las figuras de la 1 a la 3 se puede apreciar la variación del número de pivotes utilizado en la primera etapa del algoritmo para cada una de las bases de datos descritas anteriormente. Se puede observar que para casi todas las bases de datos y para todas las técnicas de selección de pivotes empleadas, existe un *n* óptimo (*n**) con el cual se calcula el menor número de distancias posible. No es este el caso de las bases de datos COLORS y MNIST. Para COLORS, solo cuando se utilizan las técnicas SSS y DPS, es posible disminuir la cantidad de distancias calculadas con respecto al AESA ($n = 0$), existiendo para ellas también un *n**.

En el caso de MNIST, la técnica de selección de pivotes RPS es la de mejores resultados, aunque también MMD reduce ligeramente el número de distancias calculadas. De manera general se puede apreciar que en todos los casos el comportamiento de las técnicas de selección de pivotes es similar al descrito en [3].

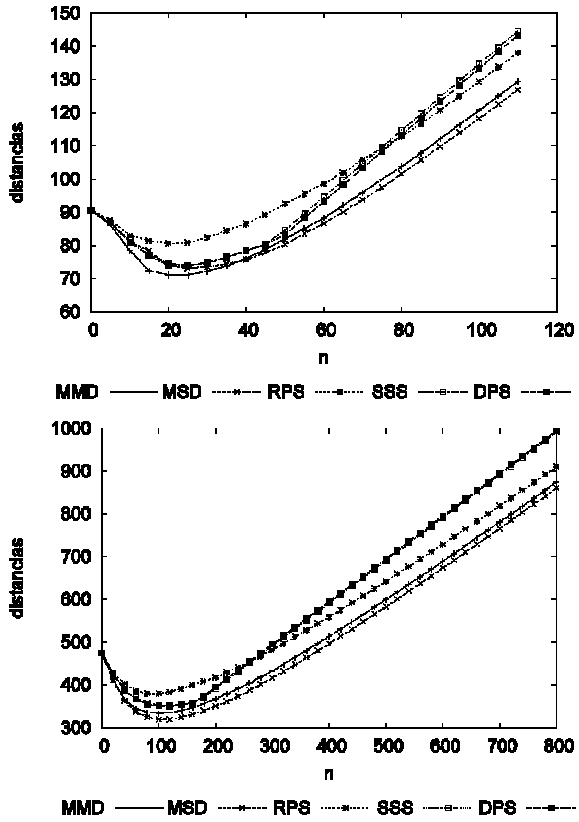

Fig. 1. Número medio de distancias calculadas al aumentar el número de pivotes (n) durante la etapa de aproximación, para un espacio con distribución uniforme en el hipercubo unidad: dimensiones 14 (arriba) y 20 (abajo)

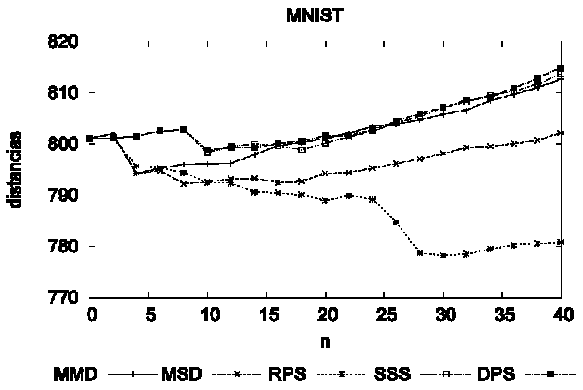

Fig. 2. Número medio de distancias calculadas al aumentar el número de pivotes (n) durante la etapa de aproximación, para las bases de datos MNIST

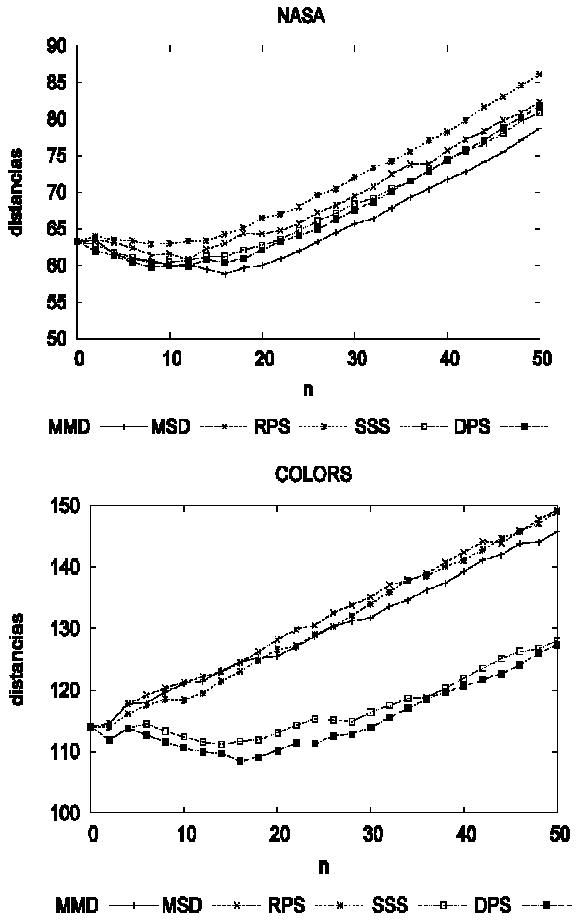

Fig. 3. Número medio de distancias calculadas al aumentar el número de pivotes (n) durante la etapa de aproximación, para las bases de datos NASA (arriba) y COLORS (abajo)

Se ha realizado un experimento adicional para comprobar el comportamiento del parámetro *n** cuando varía el tamaño de la base de datos y cuando aumenta la dimensión del espacio.

En la Fig. 4 (abajo) se presentan los resultados experimentales al incrementar el tamaño del conjunto de entrenamiento desde 500 objetos hasta 15000 en dimensión 20. Podemos ver que para tamaños pequeños del conjunto de entrenamiento (menores que 5000) el valor de n* depende del tamaño del conjunto de datos utilizados sin embargo, al incrementarse la cantidad de objetos utilizados para el entrenamiento, el valor de n* comienza a ser independiente de este tamaño. Estos

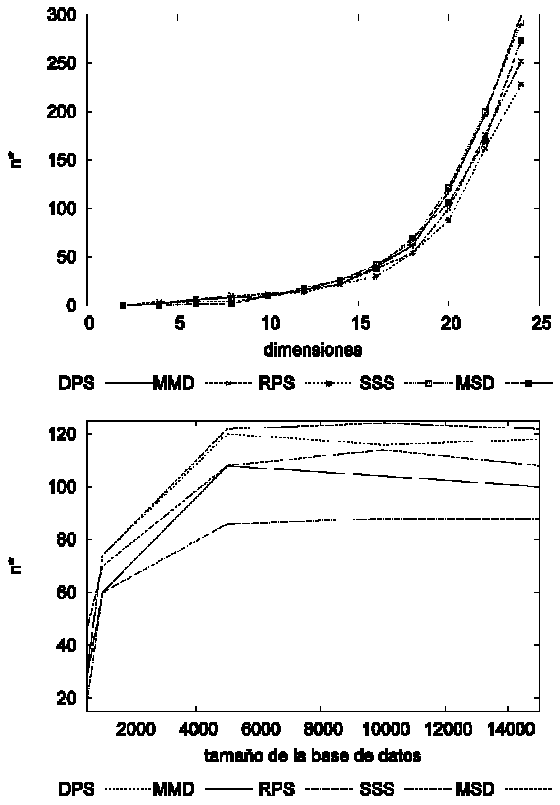

Fig. 4. Comportamiento del parámetro n* a medida que aumenta la dimensión de los datos (arriba) y el tamaño de la base de datos (abajo), en una base de datos con distribución uniforme en el hipercubo unidad

experimentos se repitieron para diferentes tamaños del conjunto de entrenamiento con resultados similares.

En la Fig. 4 (arriba) podemos apreciar que el valor de *n** crece a medida que aumenta la dimensión de los datos.

5.2 Comparación con otros métodos utilizando el criterio de búsqueda aproximada

En esta sección utilizaremos el algoritmo *PiAESAn* para obtener su versión aproximada y compararemos sus resultados en cuanto al número de distancias calculadas y el porcentaje de recuperación del *vmc* con otros algoritmos de búsqueda por similitud basados en pivotes.

Los experimentos de esta sección se realizaron utilizando conjuntos de entrenamiento con 15,000 objetos. No se realizaron pruebas para conjuntos de mayor tamaño, ya que el coste espacial de los algoritmos analizados en esta sección es cuadrático con respecto al tamaño del conjunto de entrenamiento, y el objetivo de nuestra propuesta es disminuir el coste temporal de la búsqueda, a partir de minimizar el número de distancias evaluadas.

En la Tabla 1 se presentan los resultados de los experimentos realizados al utilizar diferentes valores de holgura (*h*) con estos algoritmos para bases de datos con distribución uniforme en el hipercubo unidad y dimensiones 16, 24 y 32. El valor de la holgura no depende del tamaño del conjunto de entrenamiento y se fija antes de comenzar la búsqueda. Para estas bases de datos se puede observar que el *PiAESAn* reduce en todos los casos la cantidad de distancias calculadas con respecto al resto de los algoritmos.

Además, para los valores de *h* analizados para dimensiones 16 y 24, el porcentaje de distancias que se reduce al utilizar *PiAESAn* tiene poca variación al aumentar el valor de *h*. Por ejemplo, para dimensión 24, el *PiAESAn* reduce entre el 32% y 36% las distancias calculadas por el *AESA* y entre 19% y 23% las calculadas por el *iAESA*.

Para dimensión 32, sin embargo se puede observar que el porcentaje de reducción de distancias evaluadas se incrementa entre el algoritmo exacto ($h = 0$) y el inexacto manteniendo, de un 16,7% a un 32.2% con respecto al *AESA* al aumentar el valor de la holgura a 0.3, algo similar ocurre al compararlo con el resto de los algoritmos. Esto nos indica que para dimensiones altas esta variante aproximada del *PiAESA* puede reducir significativamente el número de distancias evaluadas con respecto a los otros algoritmos y que esta reducción aumenta con respecto a la obtenida por el algoritmo exacto.

En cuanto al porcentaje de recuperación (cantidad de *vmc* recuperados correctamente), para estas bases de datos y con estos valores de *h*, todos los algoritmos comparados obtienen un porcentaje de recuperación muy cercano al

Tabla 1. Número medio de distancias calculadas (D) y porcentaje de recuperación (PR) en la búsqueda aproximada para datos con distribución uniforme en el hipercubo unidad. Para PiAESAn se muestran los resultados con el número óptimo de pivotes. Pi/A y Pi/iA representan el porcentaje de mejora del PiAESA con respecto al AESA y al iAESA respectivamente

Dimensión 16

Algoritmo	Criterio	*h*			
		0	**0.1**	**0.2**	**0.3**
AESA	D	165.8	135.0	109.0	88.7
	PR (%)	100	100	100	99.1
iAESA	D	140.0	111.8	90.2	73.5
	PR(%)	100	100	99.9	99.7
$PiAESA_{MSD}$	D	**123.7**	**98.6**	**79.4**	**64.9**
	PR(%)	100	100	99.8	98.5
	Pi/A	25.4	26.8	27.1	26.8
	Pi/iA	11.6	11.8	12.0	11.6

Dimensión 24

Algoritmo	Criterio	*h*			
		0	**0.3**	**0.5**	**0.8**
AESA	D	1277.8	774.3	550.6	327.1
	PR (%)	100	100	100	99.4
iAESA	D	1120.2	655.4	452.9	259.6
	PR(%)	100	100	100	99.7
$PiAESA_{MSD}$	D	**864.5**	**506.0**	**352.5**	**209.8**
	PR(%)	100	100	100	99.1
	Pi/A	32.3	34.6	35.8	35.8
	Pi/iA	22.8	21.9	22.8	19.2

Dimensión 32

Algoritmo	Criterio	*h*			
		0	**0.1**	**0.2**	**0.3**
AESA	D	5518.4	2332.4	1807.5	911.2
	PR (%)	100	100	100	99.6
iAESA	D	5079.8	2032.6	1554.5	754.2
	PR(%)	100	100	100	100
$PiAESA_{MSD}$	D	**4594.6**	**1581.3**	**1175.6**	**513.1**
	PR(%)	100	100	99.9	99.4
	Pi/A	16.7	32.2	34.9	39.3
	Pi/iA	9.5	22.2	24.8	26.7

100%, aunque el *iAESA* mejora ligeramente al resto.

Para las bases de datos NASA, COLORS y MNIST (véase la Tabla 2), PiAESAn mejora ligeramente el comportamiento del resto de los algoritmos en cuanto al número de distancias evaluadas. Esto se debe a que la dimensión intrínseca de las mismas es muy pequeña. El comportamiento del porcentaje de distancias calculadas a medida que aumenta el valor de h tiene muy poca variación. En el caso de NASA, el *PiAESAn* reduce entre el 5% y el 6% de las distancias calculadas por el *AESA*, entre el 13% y el 15% las calculadas por el *iAESA*. En el caso del porcentaje de recuperación, se puede apreciar que para NASA y COLORS valores pequeños de *h* influyen en que el porcentaje de recuperación del *vmc* sea afectado rápidamente. MNIST presenta resultados interesantes, pues aunque la mejora del *PiAESAn* con respecto al *AESA* en cuanto a la cantidad de distancias calculadas es poca (entre un 2% y un 3 %), se puede observar que sí mejora el porcentaje de recuperación.

6 Conclusiones

En este trabajo hemos propuesto una modificación al criterio que utiliza el *PiAESAc* para cambiar de estrategia de aproximación. Esta propuesta llamada *PiAESAn*, simplifica el algoritmo y además permite determinar el número de pivotes utilizados en las primeras iteraciones del algoritmo.

Los resultados experimentales demuestran que la modificación propuesta tiene un comportamiento similar al *PiAESAc* en cuanto a la cantidad de distancias calculadas con las diferentes técnicas de selección de pivotes utilizadas. Además se ha propuesto una variante aproximada del *PiAESAn* al introducir el parámetro de holgura para relajar la regla de eliminación.

Los experimentos realizados para la búsqueda aproximada, demuestran que el número de distancias a calcular que se reduce al utilizar el *PiAESAn* con respecto a los otros algoritmos varia ligeramente al aumentar el valor de h, mientras que el porcentaje de recuperación es muy similar.

Tabla 2. Número medio de distancias calculadas (D) y porcentaje de recuperación (PR) en la búsqueda aproximada para bases de datos reales. Para PiAESAn se muestran los resultados con el número óptimo de pivotes. Pi/A y Pi/iA representan el Porcentaje de mejora del PiAESA con respecto al AESA y al iAESA respectivamente

NASA

Algoritmo	Criterio	*h*			
		0	**0.01**	**0.02**	**0.03**
AESA	D	63.2	161.2	59.4	57.5
	PR (%)	100	99.8	99.8	99.7
iAESA	D	69.3	67.1	65.3	63.4
	PR(%)	100	99.8	99.7	99.7
$PiAESA_{MSD}$	D	**59.5**	**57.8**	**56.0**	**54.3**
	PR(%)	100	100	100	100
	Pi/A	5.9	5.5	5.6	5.6
	Pi/iA	14.1	13.8	14,1	14.4

COLORS

Algoritmo	Criterio	*h*			
		0	**0.001**	**0.002**	**0.003**
AESA	D	114.0	122.7	111.4	110.1
	PR (%)	**100**	99.8	99.8	99.8
iAESA	D	122.3	121.2	119.9	185.5
	PR(%)	**100**	99.9	99.9	99.8
$PiAESA_{MSD}$	D	**111.9**	**110.7**	**109.4**	**108.2**
	PR(%)	**100**	99.9	99.9	99.9
	Pi/A	1.8	1.8	1.8	1.8
	Pi/iA	8.5	8.7	8.7	8.7

MNIST

Algoritmo	Criterio	*h*			
		0	**1**	**2**	**3**
AESA	D	621.0	477.5	364.0	276.5
	PR (%)	100	99.1	97	94.8
iAESA	D	656.2	506.3	386.5	293.1
	PR(%)	100	97.6	94.2	90.6
$PiAESA_{MSD}$	D	**609.6**	**466.9**	**354.2**	**267.6**
	PR(%)	100	99	97.5	96.3
	Pi/A	1.8	2.2	2.7	3.2

Agradecimientos

Los autores agradecen a la Comisión Interministerial de Ciencia y Tecnología del gobierno de España por la ayuda a través del proyecto TIN2009-14205-C04-C1, a la Consellería de Educación de la Comunidad Valenciana a través del proyecto PROMETEO/2012/01 y al Proyecto Habana de la Universidad de Alicante.

Referencias

1. **Lv, Q., Charikar, M., & Li, K. (2004).** Image similarity search with compact data structures. *Thirteenth ACM international conference on Information and knowledge management (CIKM '04),* Washington D.C., USA, 208–217.
2. **Ankerst, M., Kastenmuller, G., Kriegel, H.-P., & Seidl, T. (1999).** 3d shape histograms for similarity search and classification in spatial databases. *6th International Symposium on Advances in Spatial Databases (SSD'99),* Hong Kong, China, 207–226.
3. **Socorro, R., Micó, L., & Oncina, J. (2011).** A fast pivot-based indexing algorithm for metric spaces. *Pattern Recognition Letters*, 32(11), 1511–1516.
4. **Battiato, S., di Blasi, G., & Reforgiato, D. (2007).** Advanced indexing schema for imaging applications: three case studies. *IET Image Processing*, 1(3), 249–268.
5. **Brisaboa, N.R., Farina, A., Pedreira, O., & Reyes, N. (2006).** Similarity search using sparse pivots for efficient multimedia information retrieval. *Eighth IEEE International Symposium on Multimedia (ISM '06)*, San Diego, CA, 881–888.
6. **Bustos, B., Pedreira, O., & Brisaboa, N. (2008).** A dynamic pivot selection technique for similarity search. *1st International Workshop on Similarity Search and Applications (SISAP'08)*, Cancún, Mexico, 105–112.
7. **Chávez, E., Navarro, G., Baeza-Yates, R., & Marroquín, J.L. (2001).** Searching in metric spaces. *ACM Computer Survey*, 33(3), 273–321.
8. **Clarkson, K.L. (1999).** Nearest neighbor queries in metric spaces. *Discrete and Computational Geometry*, 22(1), 63–93.
9. **Dasarathy, B.V. (1990).** *Nearest neighbor (NN) norms: NN pattern classification techniques*. Los Alamitos, Calif.: IEEE Computer Society Press.
10. **Figueroa, K.M. (2007).** *Indexación Efectiva de Espacios Métricos usando Permutaciones*. Tesis de Doctorado, Universidad de Chile, Santiago de Chile, Chile.
11. **Figueroa, K., Chavez, E., Navarro, G., & Paredes, R. (2009).** Speeding up spatial approximation search in metric spaces. *Journal of Experimental Algorithmics*, 14, Article 6.
12. **Huang, T.S., Dagli, C.K., Rajaram, S., Chang, E.Y., Mandel, M.I., Poliner, G.E., & Ellis, D.P.W. (2008).** Active learning for interactive multimedia retrieval. *Proceedings of the IEEE*, 96(4), 648–667.
13. **Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., & Protopapas, Z. (1996).** Fast nearest neighbor search in medical image databases. *22th International Confonference on Very Large Data Bases* (*VLDB '96)*, Bombay, India, 215–226.
14. **Levenshtein, V.I. (1966).** Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8), 707–710.
15. **Micó, M.L. (1996).** *Algoritmos de búsqueda de vecinos más próximos en espacios métricos*. Tesis doctoral, Universidad Politécnica de Valencia, Valencia, España.
16. **Moreno-Seco, F., Micó, L., & Oncina, J. (2003).** A modification of the LAESA algorithm for approximated k-nn classification. *Pattern Recognition Letters*, 24(1-3), 47–53.
17. **Potamias, M. & Athitsos, V. (2008).** Nearest neighbor search methods for handshape recognition. *1st International conference on PErvasive Technologies Related to Assistive Environments (PETRA '08)*, Athens, Greece Article No. 30.
18. **Tokoro, K., Yamaguchi, K., & Masuda, S. (2006).** Improvements of TLAESA nearest neighbour search algorithm and extension to approximation search. *29th Australasian Computer Science Conference* (*ACSC'06)*, Tasmania, Australia, 48, 77–83.
19. **Vida,l E. (1985).** Diversas aportaciones al Reconocimiento Automático del Habla. PhD thesis, Universitat de Valencia, Valencia, España.
20. **Vidal, E. (1986).** An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, 4(3), 145–157.
21. **Vidal, E. (1994).** New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (AESA). *Pattern Recognition Letters*, 15(1), 1–7.
22. **Zezula, P., Amato, G., Dohnal, V., & Batko, M. (2006).** *Similarity Search. The Metric Space Approach*, New York: Springer.

Raisa Socorro Llanes, es ingeniera informática por el Instituto Superior Politécnico José Antonio Echeverría (CUJAE), Cuba. Doctora en Informática por la Universidad de Alicante. Ha participado en varios proyectos nacionales e internacionales. Su interés científico se centra en algoritmos de búsqueda por similitud, minería de datos y reconocimiento de formas.

Luisa Micó Andrés, es Licenciada en Ciencias Físicas y Doctora en Informática por la Universidad Politécnica de Valencia. En estos momentos tiene más de 50 publicaciones, en su mayoría de ámbito internacional. Sus publicaciones han recibido más de 500 citas según el Google Scholar. Además, ha participado en 15 proyectos de investigación, de los cuales ha dirigido 3, y en la red de excelencia PASCAL financiada por la Unión Europea. Es revisora habitual en revistas del campo de Reconocimiento de Formas, miembro del comité científico de varias conferencias relacionadas con el área. Su interés científico se centra en algoritmos de búsqueda por similitud, aprendizaje automático, optimización y definición de métricas y aplicaciones de reconocimiento de formas a la sociedad de la información, el ámbito industrial y el de consumo.

Article received on 13/06/2012, accepted on 20/06/2013.