Information Systems ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Approximate furthest neighbor with application to annulus query[☆]

Rasmus Pagh, Francesco Silvestri, Johan Sivertsen[*], Matthew Skala

IT University of Copenhagen, Denmark

ARTICLE INFO

Article history:
Received 30 November 2015
Received in revised form
24 May 2016
Accepted 12 July 2016
Recommended by: G. Vossen

ABSTRACT

Much recent work has been devoted to approximate nearest neighbor queries. Motivated by applications in recommender systems, we consider *approximate furthest neighbor* (AFN) queries and present a simple, fast, and highly practical data structure for answering AFN queries in high-dimensional Euclidean space. The method builds on the technique of Indyk (SODA 2003), storing random projections to provide sublinear query time for AFN. However, we introduce a different query algorithm, improving on Indyk's approximation factor and reducing the running time by a logarithmic factor. We also present a variation based on a query-independent ordering of the database points; while this does not have the provable approximation factor of the query-dependent data structure, it offers significant improvement in time and space complexity. We give a theoretical analysis and experimental results. As an application, the query-dependent approach is used for deriving a data structure for the approximate annulus query problem, which is defined as follows: given an input set S and two parameters $r > 0$ and $w \geq 1$, construct a data structure that returns for each query point q a point $p \in S$ such that the distance between p and q is at least r/w and at most wr.

1. Introduction

Similarity search is concerned with locating elements from a set S that are close to a given query q. The query can be thought of as describing criteria we would like returned items to satisfy approximately. For example, if a customer has expressed interest in a product q, we may want to recommend other similar products. However, we might not want to recommend products that are *too* similar, since that would not significantly increase the probability of a sale. Among the points that satisfy a near neighbor condition (similar), we would like to return those that also satisfy a furthest-point condition (not too similar), without explicitly computing the set of all near neighbors and then searching it. We refer to this problem as the *annulus query* problem. We claim that an approximate solution to the annulus query problem can be found by suitably combining Locality Sensitive Hashing (LSH), which is an approximation technique commonly used for finding the nearest neighbor of a query, with an approximation technique for furthest neighbor, which is the main topic of this paper.

The *furthest neighbor* problem consists of finding the point in an input set S that maximizes the distance to a query point q. In this paper we investigate the approximate furthest neighbor problem in d-dimensional Euclidean space (i.e., ℓ_2^d), with theoretical and experimental results. We then show how to cast one of our data structures to solve the annulus query problem. As shown in the opening example, the furthest neighbor problem has been used in recommender systems to create more diverse recommendations [23,24]. Moreover, the furthest neighbor is an important primitive in computational geometry

☆ The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement no. 614331.

* Corresponding author.

E-mail addresses: pagh@itu.dk (R. Pagh), fras@itu.dk (F. Silvestri), jovt@itu.dk (J. Sivertsen), mska@itu.dk (M. Skala).

http://dx.doi.org/10.1016/j.is.2016.07.006
0306-4379/© 2016 Elsevier Ltd. All rights reserved.

that has been used for computing the minimum spanning tree and the diameter of a set of points [2,11].

Our focus is on approximate solution because the exact version of the furthest neighbor problem would also solve exact similarity search in d-dimensional Hamming space, and thus is as difficult as that problem [26,3]. The reduction follows from the fact that the complement of every sphere in Hamming space is also a sphere. That limits the hope we may have for an efficient solution to the exact version, so we consider the *c-approximate furthest neighbor* (c-AFN) problem where the task is to return a point x' with $d(q,x') \geq \max_{x \in S} d(q,x)/c$, with $d(x,u)$ denoting the distance between two points.

We will pursue randomized solutions having a small probability of not returning a c-AFN. The success probability can be made arbitrarily close to 1 by repetition.

We describe and analyze our data structures in Section 2. We propose two approaches, both based on random projections but differing in what candidate points are considered at query time. In the main query-dependent version the candidates will vary depending on the given query, while in the query-independent version the candidates will be a fixed set.

The query-dependent data structure is presented in Section 2.1. It returns the c-approximate furthest neighbor, for any $c>1$, with probability at least 0.72. When the number of dimensions is $O(\log n)$, our result requires $\tilde{O}(n^{1/c^2})$ time per query and $\tilde{O}(n^{2/c^2})$ total space, where n denotes the input size.[1] Theorem 7 gives bounds in the general case. This data structure is closely similar to one proposed by Indyk [16], but we use a different approach for the query algorithm.

The query-independent data structure is presented in Section 2.2. When the approximation factor is a constant strictly between 1 and $\sqrt{2}$, this approach requires $2^{O(d)}$ query time and space. This approach is significantly faster than the query dependent approach when the dimensionality is small.

The space requirements of our data structures are quite high: the query-independent data structure requires space exponential in the dimension, while the query-dependent one requires more than linear space when $c<\sqrt{2}$. However, we claim that this bound cannot be significantly improved. In Section 2.3 we show that any data structure that solves the c-AFN by storing a suitable subset of the input points must store at least $\min\{n, 2^{\Omega(d)}\}-1$ data points when $c<\sqrt{2}$.

Section 3 describes experiments on our data structure, and some modified versions, on real and randomly generated data sets. In practice, we can achieve approximation factors significantly below the $\sqrt{2}$ theoretical result, even with the query-independent version of the algorithm. We can also achieve good approximation in practice with significantly fewer projections and points examined than the worst-case bounds suggested by the theory. Our techniques are much simpler to implement than existing methods for $\sqrt{2}$-AFN, which generally require convex programming [9,21]. Our techniques can also be extended to general metric spaces.

[1] The $\tilde{O}()$ notation omits polylog terms.

Having developed an improved AFN technique we return to the annulus query problem in Section 4. We present a sublinear time solution to the approximate annulus query problem based on combining our AFN data structure with LSH techniques [14].

A preliminary version of our data structures for c-AFN appeared in the proceedings of the 8th International Conference on Similarity Search and Applications (SISAP) [22].

1.1. Related work

Exact furthest neighbor: In two dimensions the furthest neighbor problem can be solved in linear space and logarithmic query time using point location in a furthest point Voronoi diagram (see, for example, [5]). However, the space usage of Voronoi diagrams grows exponentially with the number of dimensions, making this approach impractical in high dimensions. More generally, an efficient data structure for the *exact* furthest neighbor problem in high dimension would lead to surprising algorithms for satisfiability [26], so barring a breakthrough in satisfiability algorithms we must assume that such data structures are not feasible. Further evidence of the difficulty of exact furthest neighbor is the following reduction: Given a set $S \subseteq \{-1,1\}^d$ and a query vector $q \in \{-1,1\}^d$, a furthest neighbor (in Euclidean space) from $-q$ is a vector in S of minimum Hamming distance to q. That is, exact furthest neighbor is at least as hard as exact nearest neighbor in d-dimensional Hamming space, which is also believed to be hard for large d and worst-case [26].

Approximate furthest neighbor: Agarwal et al. [2] propose an algorithm for computing the c-AFN for *all* points in a set S in time $O\left(n/(c-1)^{(d-1)/2}\right)$ where $n=|S|$ and $1<c<2$. Bespamyatnikh [6] gives a dynamic data structure for c-AFN. This data structure relies on fair split trees and requires $O\left(1/(c-1)^{d-1}\right)$ time per query and $O(dn)$ space, with $1<c<2$. The query times of both results exhibit an exponential dependency on the dimension. Indyk [16] proposes the first approach avoiding this exponential dependency, by means of multiple random projections of the data and query points to one dimension. More precisely, Indyk shows how to solve a *fixed radius* version of the problem where given a parameter r the task is to return a point at distance at least r/c given that there exist one or more points at distance at least r. Then, he gives a solution to the furthest neighbor problem with approximation factor $c+\delta$, where $\delta>0$ is a sufficiently small constant, by reducing it to queries on many copies of that data structure. The overall result is space $\tilde{O}(dn^{1+1/c^2})$ and query time $\tilde{O}(dn^{1/c^2})$, which improved the previous lower bound when $d=\Omega(\log n)$. The data structure presented in this paper shows that the same basic method, multiple random projections to one dimension, can be used for solving c-AFN directly, avoiding the intermediate data structures for the fixed radius version. Our result is then a simpler data structure that works for all radii and, being interested in static queries, we are able to reduce the space to $\tilde{O}(dn^{2/c^2})$.

Methods based on an enclosing ball: Goel et al. [13] show that a $\sqrt{2}$-approximate furthest neighbor can always be found on the surface of the minimum enclosing ball of S.