T-79.514 Special Course on Cryptology / Privacy-Preserving Data Mining: Vulnerabilities in Similarity Search Systems

Sami Vaarala

November 24, 2003

Abstract

In this survey, we describe some fundamental vulnerabilities in similarity search systems, as described by Tosun and Ferhatosmanoglu in their paper "Vulnerabilities in Similarity Search Based Systems".

1 Definitions

The database system under discussion is modeled as a set of n-dimensional vectors, representing elements of data in a space S. Unless noted otherwise, we assume that the coordinates are real numbers and that the database is finite. Similarity (or difference) is measured using a *distance function*, $d(x, y)$, using the l_2 norm.

Two query models, illustrated in Figures 1 and 2 are used. In both models the query is represented as a single vector, q. In the *reply model*, the database returns the element q' which minimizes the distance $d(q, q')$. In the *score model* the distance $d(q, q')$, or some function $f(x)$ of the distance x, is returned instead of the element q'. (The authors only consider the case where the score directly equals the distance, i.e. $f(x) = x$.)

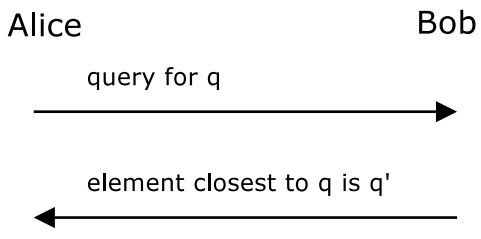

Figure 1: Reply model

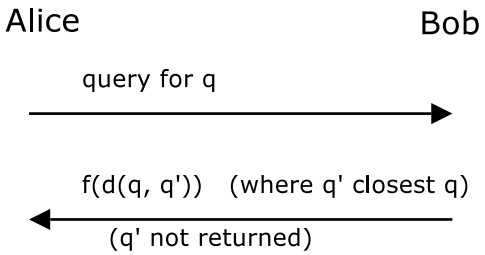

Figure 2: Score model

Given a set of query points $q_1, ..., q_m$, a Voronoi region of a certain query point q_i is the set $V = \{y \in S : d(q_i, y) \leq d(q_j, y), j \in 1, ..., m, j \neq i\}$, i.e. the set of points which are closer to query point q_i than any other query point q_j. The authors do not define their Voronoi region exactly, so it is a bit unclear how the edge of the Voronoi regions is to be handled (i.e. as an open or a closed set). This has no real effect on the results.

Query points are also referred to as probe points in the text.

2 Problem Considered

The problem described by Tosun and Ferhatosmanoglu relates only indirectly to privacy-preserving data mining. In [2], Du and Atallah describe protocols for secure approximate matching; Alice wants to know "is q approximately in the database?" without letting Bob know q or the response to the query. This problem is "inverse" to the problem discussed by Tosun and Ferhatosmanoglu, where Alice is willing to let Bob know q and the response to the query, but Bob wants to prevent Alice from reconstructing the database.

It is clear that the latter problem is, in general, insolvable. If the coordinate space is discrete, the attacker can simply iterate over all possible queries; if coordinate space is continuous, the attacker can iterate over a grid suitably dense to be considered a replica of the database. Thus, the only useful intuitive measure is the difficulty (for instance, number of queries) in doing so.

The two questions discussed by the authors are:

1. What is the best strategy to reconstruct the entire database (using queries in either query model)?
2. What can be done prevent (or alleviate) such attacks?

The vulnerabilities in score and reply models are quite different; the authors consider each separately.

3 Reply Model Vulnerabilities

Here we will examine what follows when we assume a distance function $d(x, y) = (\sum (x_i - y_i)^2)^{1/2}$ (as the authors also do). Furthermore, we assume that vector coordinates have a limited range.

3.1 Basic approach

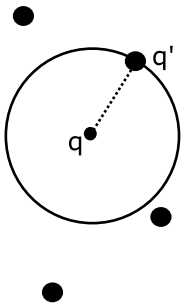

Figure 3: Closest match in reply model

When a query for q is sent in the reply model, we obtain the vector q' which minimizes the distance $d(q, q')$ (Figure 3). Also, we then know that points x for which $d(q, x) < d(q, q')$ (points inside the circle) are not in the database.

The authors describe a basic method for reconstructing a database where the minimum distance between any two vectors is c. The idea is to cover the space of possible queries with an equally spaced grid of query points. The grid spacing (distance between two query points) is chosen so that the maximum distance between any two points in the Voronoi region of a query point (i.e. the set of points for which the closest grid point is the query point in question) is at most c (see Figure 4). For n dimensions (and the assumed distance function), the grid spacing is $c/\sqrt{n}$.

If c was assumed correctly, each Voronoi region of a query point contains at most one database element. If this were not the case, the assumption regarding c would be violated (since the maximum distance between any two points in the Voronoi region is c).

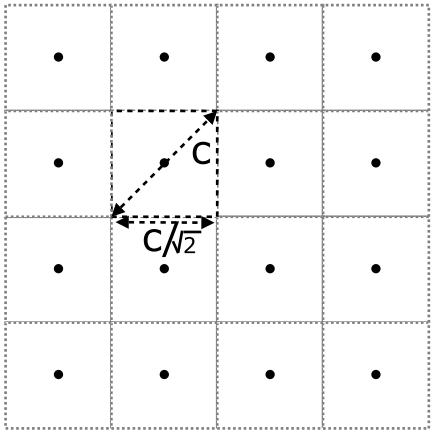

Figure 4: Query points and their Voronoi regions

The authors assume that the coordinate space (or the database) is bounded. However, this is not necessary as long as the database is finite. One can simply extend the search from the origin until all database points have been recovered. (Of course, it is impossible for the attacker to know when to stop; still, the reconstruction will be complete at some point.)

When two queries are exactly equidistant from a certain database element we have a rare "corner case". Depending on the implementation of the database the query may go one way or the other. The attacker cannot easily ensure, by an appropriate choice of the set of query points, that such corner cases are not encountered.

However, by choosing c slightly smaller than the

actual minimum distance between database elements, the corner case resolves itself automatically; the element itself is recovered in any case, and neither Voronoi region can contain elements other than the database element in question.

3.2 Schemes

The authors describe variants of the basic scheme described above:

- Progressive scheme – start with a sparse grid, and halve the grid spacing each round. This makes intermediate results more useful.
- Adaptive progressive scheme – start with a sparse grid and start refining the grid in areas where the density of database elements is high.
- Random scheme – use random probes.

The authors also suggest that multiple protocol clients could be used in conjunction with any other scheme (which they refer to as a "distributed scheme").

4 Score Model

In the score model, the authors assume that the distance between the query vector, q, and the vector closest to the query vector, q', is directly the score returned by the database in the query result. Further, they assume that the attacker knows the distance function. The basic idea is then to perform two or more queries which return scores related to the same database element q', and then use the definition of the distance function and the scores to determine coordinates of q'.

Suppose the distance function (in three dimensions) is:

$$d(a,b) = \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2 + (a_3-b_3)^2}$$

Further suppose the database element is $x = (x_1, x_2, x_3)$, and the two query points are $a = (a_1, a_2, a_3)$ and $b = (b_1, b_2, b_3)$. Then the distances $d_a = d(a,x)$ and $d_b = d(b,x)$ are as follows:

$$d_a = \sqrt{(a_1-x_1)^2 + (a_2-x_2)^2 + (a_3-x_3)^2}$$
$$d_b = \sqrt{(b_1-x_1)^2 + (b_2-x_2)^2 + (b_3-x_3)^2}.$$

The attacker chooses $a_2 = b_2$ and $a_3 = b_3$, giving:

$$d_a^2 = (a_1-x_1)^2 + (a_2-x_2)^2 + (a_3-x_3)^2$$
$$d_b^2 = (b_1-x_1)^2 + (a_2-x_2)^2 + (a_3-x_3)^2;$$

thus

$$\begin{aligned} d_a^2 - d_b^2 &= (a_1-x_1)^2 - (b_1-x_1)^2 \\ &= a_1^2 - 2a_1x_1 + x_1^2 - b_1^2 + 2b_1x_1 - x_1^2. \end{aligned}$$

Solving for x_1 we get:

$$x_1 = \frac{b_1^2 - a_1^2 + d_a^2 - d_b^2}{2b_1 - 2a_1}.$$

The authors illustrate this approach for a database containing a single element (but n dimensions). Following the authors' approach in our three dimensions, we choose $a = (0,0,0)$ and $b = (1,0,0)$ in the first set of queries. The final equation for x_1 is then:

$$x_1 = \frac{1 + d_a^2 - d_b^2}{2}.$$

The method is then iterated for every component in turn to determine all components of the unknown vector q'. Note, however, that the method only works if all scores obtained by the various queries refer to the same database vector q'. The example used by the authors avoids this by assuming there is only one element in the database.

In practice one could overcome this limitation by first doing random probes until a suitably small score was obtained for some q', and then perturbing the query very slightly and apply the more generic version of the method (described above). A potential database element can be easily verified using it as a query element; if correct, distance should be zero.

5 Attack Detection

How could these kinds of attacks be prevented or at least detected? It seems that either we need to take a complexity approach or we need some definition for what constitutes a "illegal" sequence of queries (for detection). The latter approach is implicitly used by the authors, although they do not provide any such concrete definition.

One observation is that in both score and reply models, the query results provide more information of what is *not* in the database than what is. Figure 5 illustrates the case where four queries (in either the reply or the score model, assuming that score equals distance) effectively provide a "bounding box" for the elements in the database. Each shaded circle is the set of points that are known not to be in the database.

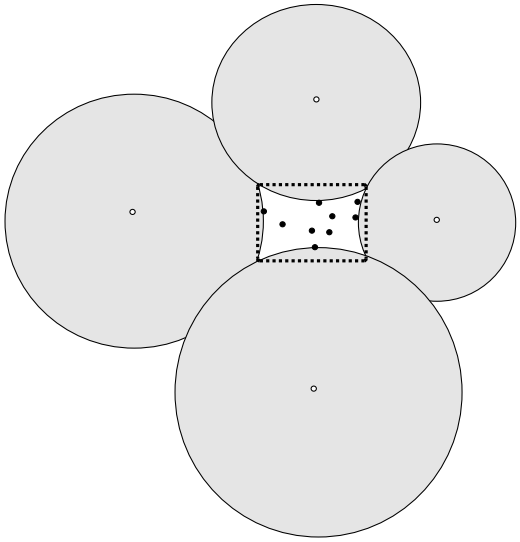

Figure 5: Four queries and "bounding box"

5.1 Reply model

For the reply model not much can be done. Although there will be some patterns in trivial query sequences, there is nothing stopping an attacker from avoiding trivial sequences. One could use e.g. an encryption function as a permutation function to generate a cryptographically strong (unpredictable) sequence of queries; alternatively, an attacker may use true random numbers to generate queries. In both cases, a suspicious feature is that the distribution is indeed random, while actual queries would probably never have such a distribution.

In some sense the problem resembles network intrusion detection systems – the definition of an "intrusion" is arbitrary, and only weak security can be achieved. False positives will also almost certainly result.

5.2 Score model

For the score model there seem to be more possibilities to protect the database.

The authors assume that the distance function is known, and the attack described in Section 4 is possible. They propose that the attack can be detected by noticing that multiple queries near a certain query vector are being made. They define a function which maps a set of coordinates and a query source IP address into a single real number as follows [1]:

$$f(x_1, ..., x_n, a, b, c, d) = 10^{12}a + 10^9 b + 10^6 c + 10^3 d + \sum x_i$$

where the source IP address is $a.b.c.d$.

The idea is to compare the difference of two queries, measured as a difference of the function values; if the difference is very small, the queries are possibly related to an attack. The authors used a "three strike policy", apparently meaning that two very similar queries are allowed but a third one is not.

It is a bit unclear why such a function f is required in the first place. One could simply define a distance function for two vectors (each consisting of query coordinates and the source IP address) and use that to determine similarity of queries. This would eliminate the false positives which result from the model used (although the probability of such false positives is rather small). Perhaps the motivation is simply to conserve space in databases with a high number of dimensions.

To combat distributed attacks, the authors suggest that the IP address classes (A, B, C) should be used to determine whether queries come from the same subnet. This solution clearly does not work; the Internet nowadays uses a flexible subnetting scheme and such strict classes no longer exist [3]. Further, network address translation (NAT) [4] of private addresses mean that thousands of queries could come from the same source IP address. (And even if there was still a way to detect the client's identity, this measure is rather weak and network specific.)

If the definition of the distance function is not public, the score model attack described by the au-

thors will not work directly (although it should be possible to match parameters for a large class of functions).

Suppose the distance function is chosen such that $d(x, y) = 0$ iff $x = y$ and $d(x, y) = 1$ otherwise (which makes sense only if the coordinate space is discrete). Clearly, using this distance function, the score model is equivalent to an exact match search, and consequently the database is as secure from reconstruction attacks as using an exact match search. Thus the security depends very much on the distance function chosen.

If $d(x, y) = (\sum (x_i - y_i)^2)^{1/2}$ and we define $d'(x, y) = \min(d(x, y), \epsilon)$ for some ϵ, d' returns a uniform result for all queries except those within ϵ from the database elements (as measured using d). The smaller the ϵ, the more difficult the attack (intuitively): a smaller ϵ means that a larger portion of the search space returns a uniform score, and at the same time each query eliminates a smaller portion of the search space. However, the smaller ϵ is, the less useful the similarity search is for the user. (Figures 6 and 7 illustrate: the database consists of six points and the distance to the nearest point is plotted for each query point. The vertical axis is the value of the distance function (to the closest database element), while the two other axes are the two coordinates.)

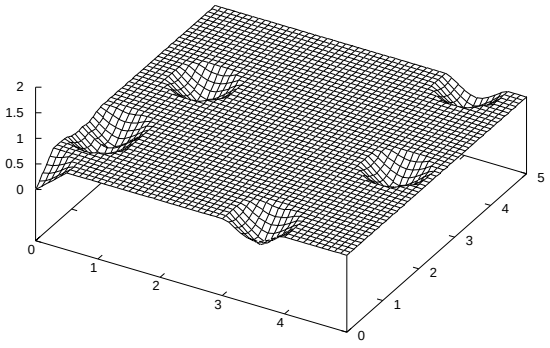

Figure 6: "Capped" distance function, $\epsilon = 0.5$

One possibility is to discretize the distance function, e.g. $d'(x, y) = \text{ceil}(d(x, y))$ (Figure 8). This might also increase the difficulty of the attack (as no "gradient information" is usually revealed). However, the distance function still returns arbitrarily high values, which can be used to dismiss large portions of the search space quickly (as illustrated in Figure 5).

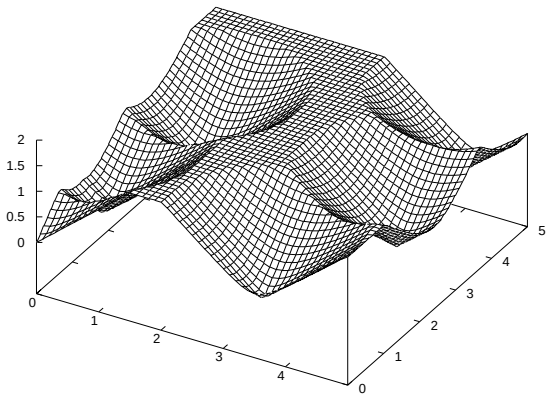

Figure 7: "Capped" distance function, $\epsilon = 1.5$

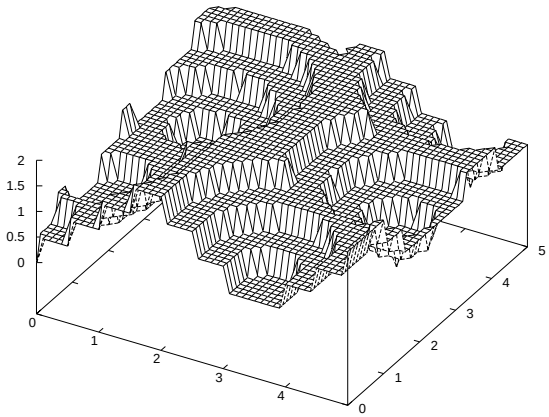

Figure 8: Discrete distance function

6 Experiments Performed by Authors

The experiments described in the paper provide quite ambiguous results, and do not confirm whether the attacks are realistic. Attacks in the reply model are clearly possible, however, the authors use databases with only a few dimensions which do not represent real databases well (all three experiments use a two-dimensional database).

The authors did not try an attack on the score model. Instead, they analyzed the distribution of the sums of element coordinates (the function f in the discussion above). This provides no useful information about how realistic attacks against the score model are.

7 Thoughts

The authors do not give an explicit measure of "performance" of a reconstruction algorithm. This makes comparison of relative advantages of each algorithm difficult. The practical experiments do not shed light on the practical effectiveness of the algorithms, as described previously.

An oversimplified example of reconstruction effort: with 10 dimensions and coordinates in the range $[1, 100]$, 100^{10} queries are required in the reply model (which is easier) to reconstruct the database completely when grid spacing is 1 (and thus $c = \sqrt{10}$). An attacker capable of a billion queries per second would require ca. 3170 years to complete the database reconstruction. Of course, database elements representing the rough distribution of elements in the database can be found much more rapidly using the (adaptive) progressive scheme.

Whether this is a problem depends on what constitutes a "reconstruction". Consider the situation in Figure 9. In the figure, circled database elements are recovered using queries while others are not. Note that no assumption about a minimum distance c is made in the Figure. On the contrary, it is assumed that the attacker's choice for c is too large (which leads to multiple database elements populating a given Voronoi region).

Although the query grid properly characterizes the areas where some points are located, it provides little information about density of points which may be an important factor. The density is difficult to recover if the minimum distance between points in dense clusters is very small.

In the example given, an arbitrary number of points could be added to cluster of points in the database without changing the results of the query sequence.

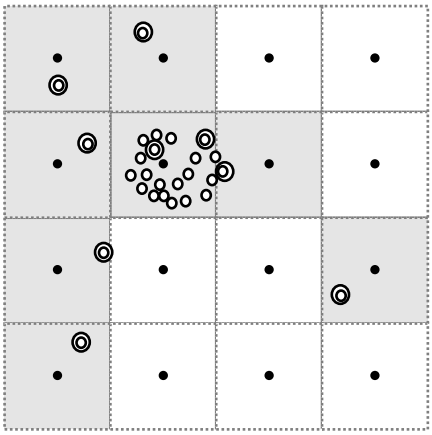

Figure 9: Recovery density is difficult

The importance of such "density" information depends on the application; if one is interested in knowing whether any database element has (approximately) some attributes, density is not important. On the other hand, if one is interested in the number of database elements sharing (approximately) some attributes, density information makes a difference.

References

[1] A. Tosun and H. Ferhatosmanoglu. Vulnerabilities in Similarity Search Based Systems. Conference on Information and Knowledge Management (CIKM 2002).

[2] W. Du and M. Atallah. Protocols for Secure Remote Database Access with Approximate Matching. 7th ACM Conference on Computer and Communications Security (ACM-CCS 2000).

[3] V. Fuller, T. Li, J. Yu and K. Varadhan. Classless Inter-Domain Routing (CIDR): an

Address Assigment and Aggregation Strategy. RFC 1519. Internet Engineering Task Force, 1993.

[4] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022. Internet Engineering Task Force, 2001.