Seminar on Algorithms for Database Systems - Student Report

Dimensionality Reduction Techniques for Proximity Problems [1]

Dhivyabharathi Ramasamy

1 Introduction

In the paper "Dimensionality Reduction Techniques for Proximity Problems" [1], Dr. Piotr Indyk* has proposed approximation algorithms for several proximity problems in high dimensional spaces. Here, we will discuss two proximity problems. First, I will explain what a proximity problem is and then, the need for them. Then, we will look at the technique of dimensionality reduction using hashing in Hamming spaces as proposed by Indyk et. al. and proceed to discuss the theorem for reduction of *Furthest Neighbour Search* which improves in time complexity when reduced to *Nearest Neighbour Search* problem along with the proof.

Hereafter, the term "the author" is used to refer to Dr. Piotr Indyk.

Proximity Problems:

Proximity problems is a class of computational geometric problems which involve estimation of distances between geometric objects(in our case, points in a d-dimensional space). Closest pair problems, furthest neighbour search(*FNS*) and nearest neighbour search(*NNS*) problems are all a subset of these problems. They are used in similarity search or clustering in high dimensional spaces. These have high relevance in our work today for applications in areas like Information retrieval, image and video databases, data mining and pattern recognition.

Focus of the seminar:

The topics that we will discuss here are: Nearest Neighbour Search(NNS) – Approximate Nearest Neighbour Search(c-NNS) – Locality Sensitive Hashing(LSH) – Furthest Neighbour Search(FNS)

2 Nearest neighbor problem:

First, let's think of a simple case of nearest neighbour application.

Application:

One of the most important real-world application of nearest neighbour search is recommender systems. In online-commerce, it is widely used for user convenience and to bring the store, benefits. The aim of recommender systems is to find the most accurate item the user would show higher preference to.

Let's consider a simple example containing few users and items. In the Figure 0[4], we see that user1-user3 have similar likes. This means, those items rated highly by user3 could be the interesting item that can be recommended to user1. The strategy is "*The friend of my friend is my friend*". This brings multiple benefits for both the user and the store.

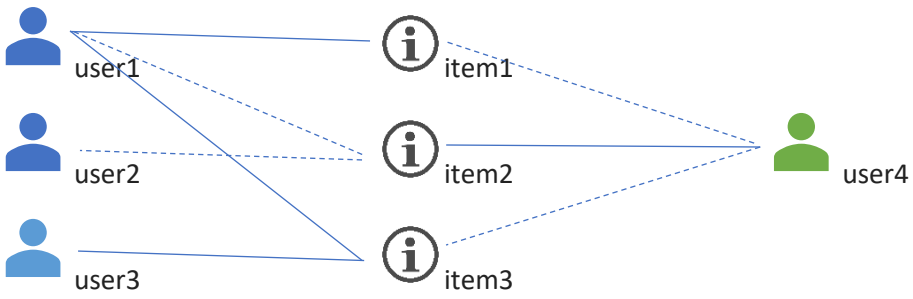

Figure 0: Recommender Universe

Let's look at the definitions of NNS, c-NNS and FNS.

Definition 1 (Nearest Neighbour Search Problem(NNS))

Given a set of points $P = \{p_1, \dots p_n\}$ from some metric space (X, d), we have to construct a data structure, which given a query point $q \in X$, finds the nearest neighbour of $q \in X$ in $p \in P$ based on some distance metric (here, we follow Hamming metric).

The quality and usefulness of the algorithms are determined by the time complexity of queries as well as the space complexity of any search data structures that must be maintained. As the dimensions grow, it is difficult to achieve such an efficient data structure and search time. This is in general expressed as Curse of dimensionality. We can deal with this in an alternative way, by solving an approximate problem. It is always not necessary to solve *exact NNS.* It is sometimes enough to find an *approximate nearest neighbour* if the notion of distance is somehow accurately captured.

Definition 2 (c-Nearest Neighbour Search Problem(c-NNS))

Given a set $P = \{p_1, \dots p_n\}$ of points from some metric space (X, d), devise a data structure which, given any query $q \in X$, produces a point $p \in P$ such that $d(q, p) \leq c\ min_{p' \in P} d(q, p')$.

Definition 3 (c-Furthest Neighbour Search Problem(c-FNS))

Given a set $P = \{p_1, \dots p_n\}$ of points from some metric space (X, d), devise a data structure which, given any $q \in X$, produces a point $p \in P$ such that $d(q, p) \geq 1/c\ max_{p' \in P} d(q, p')$.

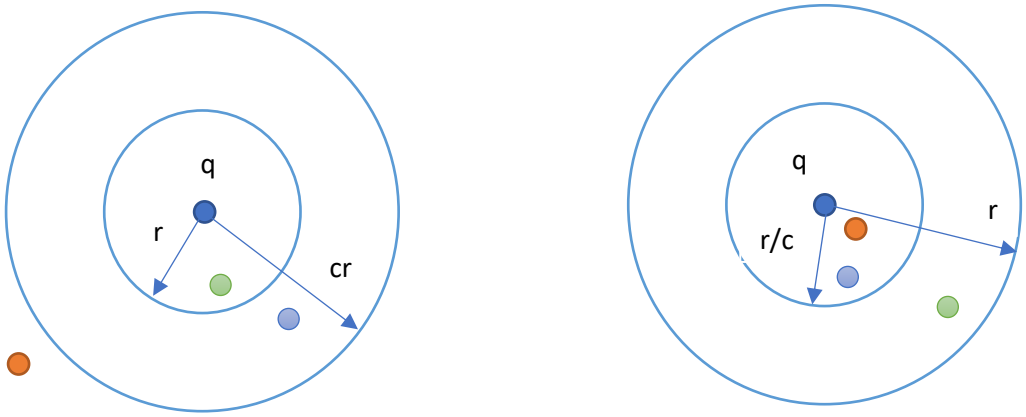

Figure 1: Approximate Neighbour Search Visualization

(Left: *c-Approximate NN*, Right: *1/c-Approximate FN*)

3 Locality Sensitive Hashing(LSH)

One of the main technique used in approximation methods is LSH. The main idea here is to use hash functions such that probability of collision is much higher for points close to each other than the points that are far apart. Algorithms that support the approximate nearest neighbor search include locality-sensitive hashing. Next, we will see the definition and a small proof of LSH with a focus on time and space complexity.

Definition 4([2]) **(Locality Sensitive Hashing)**

A family $\mathcal{H}$ of d hash functions of the form $h: P \to U$ is called $\{r_1, r_2, p_1, p_2\}$-sensitive for H^d if for any $q, p \in S$

- *If $d_H(p,q) \leq r_1$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \geq p_1$.*
- *If $d_H(p,q) > r_2$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \leq p_2$.*

Where d_H is Hamming Distance, r_1 is r and r_2 is cr.

Let's define $c = (1+\varepsilon)$ for a c-*NNS* problem, where ε is the approximation factor.

Fact 1: *Let $P = H^d$ and $d(p,q)$ be the Hamming metric for $p, q \in H$. Then for any $r, \varepsilon > 0$, the family $\mathcal{H} = \{h_i : h_i((b_1, b_2, \dots b_d)) = b_i, i = 1,2, \dots n\}$ is $(r, r(1+\varepsilon), 1 - \frac{r}{d}, 1 - \frac{r(1+\varepsilon)}{d})$-sensitive.*

Let's define a family of d hash function $\mathcal{H}_d$. Each hash function is of the form $h^{(i)}: P \to U$

Lemma: *For $p, q \in H^d$, $\Pr_{\mathcal{H}_d}[h(p) = h(q)] = 1 - \frac{d_H(p,q)}{d}$.*

Proof: Let $d_H(p,q) = \delta$. δ gives the distance between point p and q as defined by Hamming metric. That is, the positions where p and q differ. This means, for an i uniformly chosen at random, probability of p_i being not equal to q_i is $\frac{\delta}{d}$.
This implies $\Pr_{\mathcal{H}_d}[h_i(p) = h_i(q)] = 1 - \frac{\delta}{d}$. So, we get $\Pr_{\mathcal{H}_d}[h(p) = h(q)] = 1 - \frac{d_H(p,q)}{d}$.

Corollary: Since we know $1 > p_1 > p_2 > 0$, $\rho: \frac{log\, p_1}{\log p_2} < 1$.

Hashing by Dimension Reduction:

Using $\mathcal{H}_d$, we build a family of hash function $\mathcal{G}_K$ of the form $g: H^d \to \{0,1\}^K$. g is picked uniformly at random from $\mathcal{G}_K$. That is, $g := (h_k)_{k \in [K]}$ be the concatenation of $h_k's$, where h_k is picked uniformly at random from $\mathcal{H}_d$.

Thus, for every $p \in H^d$, $g(p) := (h_0(p), h_1(p), \dots h_{K-1}(p))$

Hash Table:

For every $p \in H^d$, a pointer to the point p is stored in the hash table T with key $g(p) \in \{0,1\}^K$.

Set $K := \log_{1/p_2} n$, $L := n^\rho$, $r > 0$ and $(1+\varepsilon) > 1$. We will now construct a data structure for *approximate range search*. For above value of K and L, the algorithm succeeds with constant probability [2].

Constructing data structure:

Now, we will construct a data structure such that a hash table T_l is constructed for every $l \in [L]$ which stores pointers to points in P.

- First, we pick a function g_l from $\mathcal{G}_K$ uniformly at random.
- Then, for each $p \in H^d$, a pointer to the point p is stored in the hash table T_l with key $g_l(p) \in \{0,1\}^K$

Time and Space Complexity([3])**:**

Since we have L hash tables now, time to construct the hashing table is $\mathrm{O}(LnK)$ and space used for the L hash tables is given by $\mathrm{O}(Ln)$ while space used for original P is given by $\mathrm{O}(nd)$. Total storage is given by $\mathrm{O}(Ln + nd)$.

Substituting values for L, $\mathrm{O}(Ln + nd) = \mathbf{O}\left(\boldsymbol{dn} + \boldsymbol{n^{1+\rho}}\right)$ is the total storage for the data structure.

For construction time, $\mathrm{O}(LnK) = \mathbf{O}\left(\boldsymbol{n^{1+\rho}}\,\mathbf{log}_{\mathbf{1}/\boldsymbol{p_2}}\,\boldsymbol{n}\right)$ is the construction time for the data structure.

For a given query point q, computation time for $g_l(q)$ is $\mathrm{O}(K)$ and distance computation ($d_H(p,q)$) is given by $\mathrm{O}(d)$ giving total computation time as $\mathrm{O}(K + d)$ for one hash table. So, the total time of algorithm for a given query is $\mathrm{O}(L(K + d))$. By substituting values, the running time for a query is $\mathbf{O}\left(\boldsymbol{n^{\rho}}(\mathbf{log}_{\frac{\mathbf{1}}{\boldsymbol{p_2}}}\,\boldsymbol{n} + \boldsymbol{d})\right)$. Now that we derived the time and space complexity for approximate nearest neighbour search using hashing, let's see how furthest neighbour search can be solved efficiently.

4 Furthest Neighbour Search:

Application:

While the problem of recommender systems is to find the most accurate item the user would show higher preference to, there are also other aspects that are important. One such aspect is diversity. In recommender systems, the recommended items are familiar to the user, but simply have not been rated by them yet. In such case, there is no diversity of recommendations which would provide a value-addition or unexpected-ness.

Let's consider the same example we had for nearest neighbour recommender systems. In our Figure 0, we see that user1-user2 and user1-user3 have similar tastes. We also see that user4 has different taste. This mismatch can also be used to suggest interesting and diverse items. For example, instead of suggesting albums of the artist user4 likes(*NNS*), one could also suggest albums of artists user1-user2 dislikes(*FNS*). The strategy is "*The enemy of my enemy is my friend*".

Furthest to nearest neighbour reduction:

Consider ε' and ε being the approximation factors of *FNS* and *NNS* problem. Now, can we solve $(1 + \varepsilon')$-*FNS* as $(1 + \varepsilon)$-*NNS*? Yes. Dimensional hashing can be applied to reduce $(1 + \varepsilon')$-*FNS* to $(1 + \varepsilon)$-*NNS* in Hamming spaces. This is interesting, because, using the same computational resources, it is possible to solve $(1 + \varepsilon)$-*NNS* efficiently than $(1 + \varepsilon')$-*FNS* for some limit of approximation. We will see how. First, let's state the theorem.

Theorem 1: *There is a (randomized Monte Carlo) reduction of* $(1 + \varepsilon')$*-FNS problem for n points in* $\{0,1\}^d$ *to* $(1 + \varepsilon/6)$*-NNS problem for n points in* $\{0,1\}^{poly(D,\log n,1/\varepsilon)}$ *for* $\varepsilon \in [0,2]$.

Proof:

For reducing from *exact-FNS* to *exact-NNS*, author use a simple idea. Let $p, q \in \{0,1\}^d$; then $d(p,q) = d - d(p,\bar{q})$, where $\bar{q}$ denotes the complement of q. This means, if P is a set of points in $\{0,1\}^d, q \in \{0,1\}^d$ and $p \in P$ is a nearest neighbour of q in P, then p is also a furthest neighbour of $\bar{q}$ in P.

This shows a furthest neighbour of a point is a nearest neighbour of its complement. So, the exact versions of furthest and nearest neighbour are essentially equivalent.

Since we are interested in solving approximate problem of FNS, we would like to reduce *approximate-FNS* to *approximate-NNS.* We can write this problem as reduction of $(1+\varepsilon')$-*FNS* to $(1+\varepsilon)$-*NNS* where $(1+\varepsilon')$ and $(1+\varepsilon)$ are constant approximation factors for *FNS* and *NNS* respectively.

It is easy to see this reduction may not preserve approximation. So, to reduce the *approximate-FNS* to *approximate-NNS*, we must find out the relationship between ε and ε'.

Let's say, the furthest neighbour of our query point q in P is p and the furthest distance is R ($d(q,p) = R$) and p' is E'-*approximate FN* of q ($d(q,p') \geq \left(\frac{1}{E'}\right) d(q,p)$). Using the notion $d(q,p') = d - d(\bar{q},p')$. So, E'-*approximate FNS* to *ANS* gives us

$$E' \geq \frac{d(q,p)}{d - d(\bar{q},p')} \qquad \text{--- (1)}$$

We find an E-*approximate NN* of $\bar{q}$ in P and we call that p'. Implying $d(\bar{q},p') \leq E d(\bar{q},p)$. Substituting known values, we get $d(\bar{q},p') \leq E(d-R)$. Now, equation (1) becomes

$$E' \geq \frac{R}{(d - E(d-R))}$$

We can write this as,

$$E \leq \frac{d-R'}{d-R} = \frac{1-\rho'}{1-\rho} \qquad \text{--- (2)}$$

where, $R = E'R', R = \rho d, R' = \rho' d$.

(The paper has a typo that $R' = E'R$, but this doesn't lead the proof further. It must be $R = E'R'$ to derive further steps and derivations correctly.)

Equation (2) shows, the relation between E and E' depends on the ratio of d to R. Reduction is more efficient when the ratio is smaller.

To reduce this ratio, we apply dimensional hashing technique:

Hashing technique: for any $p \in \{0,1\}^D$ and dimension $i = 1, \ldots D$ the i^{th} coordinate of $f(p)$ is obtained by:

$$f_i(p) \coloneqq p_{i_1} p_{i_2} \ldots p_{i_k}$$

where $i_1 \ldots i_k$ are chosen independently and uniformly at random with replacement.

Using the hashing technique, it is easy to see that for given points p, q if $d(p,q) = \rho d$, the expected value

$$E[d(f(p), f(q))] = \left(1 - (1-\rho)^k\right) D \qquad \text{--- (3)}$$

By Chernoff bound, the expected value and the actual value differs only by $(1 \pm \alpha)$ when $D = \Omega(\log n/\alpha^2)$. For simplicity, we assume $\alpha = 0$. For small ρ, without loss of generality, we can write $1 - (1-\rho)^k = (1 - e^{-\rho k})$. So, we get, $d\big(f(p), f(q)\big) = \big(1 - e^{-\rho k}\big)D$. By applying error correcting codes, the value can be reduced to $\frac{(1-e^{-\rho k})}{2} D'$.

Let $\frac{(1-e^{-\rho k})}{2} = \gamma_k(\rho)$. Function γ defines the probability in dimension D. From equation (2), we can deduce that $\rho' = \rho/E'$. $R = \rho d$ in the space H^d is equivalent to R = $\gamma_k(\rho)D$ in the new space. Hence, ρ in Eq. 2 can be replaced with the value of $\gamma_k(\rho)$. Now, we get

$$E \leq \frac{1+e^{-\rho' k}}{1+e^{-\rho' E' k}} \qquad \text{--- (4)}$$

The above equation defines the upper bound for E.

Let $\delta = \rho' k$. To derive the approximation factor of *NNS* with respect to given *FNS* problem, we pick values of δ as a function of E'. That is, $\delta(E')$. For $\varepsilon' = 2$, the value of δ can be solved for the upper bound of E.

Differentiating for the upper bound using chain rule and solving the resulting equation $E \leq \frac{1+e^{-\delta}}{1+e^{-3\delta}}$, we get $e^{\delta} = 2$, $\delta = ln2 \approx 0.693$. Substituting the value of δ in (4), $E = 1.3333$ and $\varepsilon = 0.3333$. This gives, $\left(\frac{\varepsilon}{\varepsilon'}\right) = \left(\frac{1}{6}\right) \approx 0.167$. Thus, we prove the Theorem 1. The problem now can be efficiently solved as approximate nearest neighbour search using Locality Sensitive Hashing method we discussed before. But we should check, does the reduction really improve efficiency? Now, note that the ratio ε/ε' decreases with ε'.

Time and Space Complexity:

Using above reduction, we can get our storage, computation time and query time as a function of approximation factor ε'. Here, as probability γ_k is a function of ε', we get the following:
For $(1+\varepsilon')$-*FNS* problem, the running time for a query is $\widetilde{\mathrm{O}}\big(Dn^{\gamma(\varepsilon')}\big)$ and data structure construction time is $\widetilde{\mathrm{O}}\big(n^{1+\gamma(\varepsilon')}\big)$ while $\mathrm{O}\big(Dn + n^{1+\gamma(\varepsilon')}\big)$ is the total storage for the data structure for dimension D.

Based on our reduction, in *NNS*, $c = (1 + \varepsilon/6)$

$$p_2 = 1 - \frac{r(1+\varepsilon')}{d} < 1 - \frac{r(1+\varepsilon/6)}{d}$$

The probability p_2 increases while remaining lower than p_1. Implying, $K \coloneqq \log_{1/p_2} n$ decreases for $(1+\varepsilon/6)$-*NNS* problem. So, the running time for a query, $\mathbf{O}\left(\boldsymbol{n^{\rho}}(\mathbf{log}_{\frac{1}{\boldsymbol{p_2}}}\boldsymbol{n} + \boldsymbol{d})\right)$ and data structure construction time, $\mathbf{O}\big(\boldsymbol{n^{1+\rho}}\,\mathbf{log}_{\boldsymbol{1/p_2}}\,\boldsymbol{n}\big)$ becomes more efficient for solving $(1+\varepsilon/6)$-NNS problem than $(1+\varepsilon')$-*FNS* problem. $\mathrm{O}\big(Dn + n^{1+n^{\rho}}\big)$ is the total storage for the data structure for dimension D. With decreasing ratio of ε/ε', it is very clear that with this reduction, we can solve the problem more efficiently.

5 Conclusion:

The paper presents Dimensionality Reduction Techniques for several proximity problems. While approximation problems for nearest neighbour search has been discussed extensively in many papers, this paper provides a new way of approaching furthest neighbour problem using a simple reduction of the problem to nearest neighbour search. With improvement in time complexity, this is clearly a better algorithm to use but only when $\varepsilon \in [0,2]$ and the metric is hamming distance. For $\varepsilon > 2$, there are other general metric space that provide efficient algorithms.

To conclude, the author has provided an efficient and fascinating derivation of the theorem that shows the relation between approximation factors in furthest neighbour problem and nearest neighbour problem when $\varepsilon \in [0,2]$.

6 References

[1] Piotr Indyk. *Dimensionality Reduction Techniques for Proximity Problems*. In SODA '00 Proceedings of the eleventh annual ACM-SIAM symposium on Discrete Algorithms Pages 371-378.

[2] Piotr Indyk and Rajeev Motwani. *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality.* In STOC '98 Proceedings of the thirtieth annual ACM symposium on Theory of computing Pages 604-613.

[3] A. Gionis, Piotr Indyk and Rajeev Motwani. *Similarity Search in High Dimensions via Hashing.* In VLDB '99 Proceedings of the 25th International Conference on Very Large Data Bases Pages 518-529.

[4] Alan Said, Benjamin Kille, Brijnesh J. Jain, Sahin Albayra. *Increasing Diversity Through Furthest Neighbour-Based Recommendation.* http://www.dcs.gla.ac.uk/workshops/ddr2012/papers/p3said.pdf