Supplemental Material for "On the Difficulty of Nearest Neighbor Search"

1. Notation Table

Table 1. Notations

Symbol	Meaning
x	a random vector x
x^j	the j dimension of x
x_i	a sample in database, each x_i a i.i.d. sample of x
n	the number of samples in database
q	query
d	number of dimensions
p	parameter for L_p norm
s	fraction of non-zero dimensions
$D_d(,)$	distance for d-dimensional data, abbreviated as $D(,)$ if no ambiguity
D^q_{max}	$\max_{i=1,\ldots n} D_d(x_i, q)$, maximum distance between q and database samples
D^q_{min}	$\min_{i=1,\ldots n} D_d(x_i, q)$, minimum distance between q and database samples
D^q_{mean}	$meanD_d(x_i, q)$, mean distance between the query and database samples
D_{min}	$E_q(D^q_{min})$, expected minimum distance between queries and database samples
D_{mean}	$E_q(D^q_{mean})$, expected mean distance between the queries and database samples

2. Proofs

Proof of Theorem 2.2:
The probability for both x^j and q^j to be non-zero is s_j^2, and the probability for one of them to be non-zero is $2(1-s_j)s_j$. Hence, the mean

$$\mu_j = E[R_j] = E[|x^j - q^j|^p]$$

for sparse vectors can be computed as,

$$\mu_j = s_j^2 m'_{j,p} + 2(1-s_j)s_j m_{j,p}$$

Similarly, the variance

$$\sigma_j^2 = Var[R_j] = E[R_j^2] - E[R_j]^2 = E[|x^j - q^j|^{2p}) - \mu_j^2$$

for sparse vectors can be given as,

$$\sigma_j^2 = s_j^2 m'_{j,2p} + 2(1-s_j)s_j m_{j,2p} - \mu_j^2,$$

Thus, the normalized variance for sparse vectors is:

$$\sigma'^2 = \frac{\sum_{j=1}^d \sigma_j^2}{(\sum_{j=1}^d \mu_j)^2}. \tag{1}$$

If we assume each dimension to be i.i.d, i.e., all V_j have the same distribution with $E[V_j] = \mu_d$, $var[V_j] = \sigma_d^2$, and also assume $s_j = s$, $m_{j,p} = m_p$ and $m'_{j,p} = m'_p$, then

$$\sigma' = \frac{1}{d^{1/2}}\frac{\sigma_d}{\mu_d} = \frac{1}{d^{1/2}}\sqrt{\frac{s[(m'_{2p} - 2m_{2p})s + 2m_{2p}]}{s^2[(m'_p - 2m_p)s + 2m_p]^2} - 1} \tag{2}$$

Proof of Theorem 3.1:
With the hash functions of

$$h(x) = \lfloor \frac{w^T x + b}{t} \rfloor$$

it can be shown that(Datar et al., 2004),

$$P(h(x_i) = h(q)) = f_h(||x_i - q||_p) \tag{3}$$

where function $f_h(a) = \int_0^t \frac{1}{a} f_p(\frac{z}{a})(1 - \frac{z}{t})dz$ is monotonically decreasing with a. Here f_p is the p.d.f. of the absolute value of a p-stable variable.

Suppose the data are normalized by a scale factor such that $D_{mean} = 1$. Note that such a normalization will not change the nearest neighbor search results at all. In this case, $D_{min} = 1/C_r$. Denote p_1 (p_2) as the probability for one random query q and its nearest neighbor (q and a random database point) to have the same code with one hash function. According to equation (3),

$$p_1 = f_h(1/C_r)$$

and

$$p_2 = f_h(1),$$

since the expected distance between q and its nearest neighbor is $D_{min} = 1/C_r$, and the expected distance between q and a random database point is $D_{mean} = 1$.

Suppose there are k hash bits in one table and l hash tables in LSH. The probability that the true nearest neighbor will have the same code of the query in one hash table is p_1^k. So The probability that the true nearest neighbor will be missed in one hash table is $(1-p_1^k)$ and will be missed in all l hash tables is $(1-p_1^k)^l$. We want to make sure

$$(1-p_1^k)^l = \delta.$$

So

$$l = \frac{\log\delta}{\log(1-p_1^k)} \le \frac{-\log\delta}{p_1^k} = \log\frac{1}{\delta}p_1^{-k}$$

The number of all hash bits to compute are

$$O(kl) = O(k\log\frac{1}{\delta}p_1^{-k}).$$

The number of all points falling into the query bucket in one table are $O(np_2^k)$. In total there are l hash tables, the number of points to be check will be

$$O(lnp_2^k).$$

As discussed in (Gionis et al., 1999), we can choose

$$np_2^k = O(1),$$

i.e., $k = O(\frac{\log n}{\log p_2^{-1}})$. Note that

$$p_1 = p_2^{\frac{\log p_1}{\log p_2}},$$

so

$$p_1^k = (p_2^{\frac{\log p_1}{\log p_2}})^k = (p_2^k)^{\frac{\log p_1}{\log p_2}} = O((\frac{1}{n})^{\frac{\log p_1}{\log p_2}}) = O(n^{-g(C_r)})$$

where

$$g(C_r) = \frac{\log p_1}{\log p_2} = \frac{\log f_h(1/C_r)}{\log f_h(1)}.$$

And hence

$$l \le \log\frac{1}{\delta}p_1^{-k} = O(\log\frac{1}{\delta}n^{g(C_r)}).$$

And the number of all points to check, or in other words, the number of returned candidate points, is

$$O(lnp_2^k) = O(\log\frac{1}{\delta}n^{g(C_r)}).$$

Since $f_h(\cdot)$ is a monotonically decreasing function, when C_r is larger, $g(C_r)$ will be smaller[1] . This completes the proof.

[1] Note that both $\log f_h(1/C_r)$ and $\log f_h(1)$ are negative, since $f_h(\cdot)$ is always ≤ 1.

Proof of Corollary 3.2:
From the proof above, we know $l = \log\frac{1}{\delta}p_1^{-k}$ and $p_1^{-k} = O(n^{g(C_r)})$, so

$$l = O(\log\frac{1}{\delta}n^{g(C_r)}).$$

The number of returned candidate points, is

$$O(lnp_2^k) = O(\log\frac{1}{\delta}n^{g(C_r)}).$$

The number of all hash bits to compute is

$$O(\frac{\log n}{\log p_2^{-1}}\log\frac{1}{\delta}p_1^{-k}) = O(\frac{\log n}{\log p_2^{-1}}\log\frac{1}{\delta}n^{g(C_r)}).$$

Since computing one hash bit and check one point both take $O(d)$, the time complexity of LSH will be

$$O(d\frac{\log n}{\log p_2^{-1}}\log\frac{1}{\delta}n^{g(C_r)}),$$

and moreover, we need l tables, each table will have space complexity $O(n)$, so the total space complexity for LSH will be

$$O(nd+nl) = O(nd+\log\frac{1}{\delta}n^{(1+g(C_r))}).$$

This completes the proof.

Proof of Theorem 3.3:
After projecting the data on vector w, the squared L_2 distance between a query q and its nearest neighbor $x_{q,NN}$ is

$$(w^Tq - w^Tx_{q,NN})^2.$$

Moreover, the expected distance between q and a random point x is

$$E_x(w^Tq - w^Tx)^2,$$

which can be approximated as

$$\frac{1}{n}\sum_i(w^Tq - w^Tx_i)^2.$$

To find projections that maximize the (squared) relative contrast, we will have

$$\hat{w} = \arg\max_w \frac{E_q[\sum_i(w^Tq - w^Tx_i)^2]}{E_q[(w^Tq - w^Tx_{q,NN})^2]} \quad (4)$$

$$= \arg\max_w \frac{w^TE_q[\sum_i(q-x_i)(q-x_i)^T]w}{w^TE_q[(q-x_{q,NN})(q-x_{q,NN})^T]w} \quad (5)$$

$$= \arg\max_w \frac{w^TS_Xw}{w^TS_{NN}w} \quad (6)$$

where $S_X = E_q[\sum_i (q - x_i)(q - x_i)^T]$.

Since $\sum_i x_i = 0$,

$$S_X = E_q[nqq^T + \sum_i x_i x_i^T].$$

Moreover, q has the same distribution as x, so

$$E_q[qq^T] \approx \Sigma_X,$$

where $\Sigma_X = (1/n)\sum_i x_i x_i^T$. Hence S_X can be approximated as $2n\Sigma_X$, and

$$\hat{w} = \arg\max_w \frac{w^T S_X w}{w^T S_{NN} w} = \arg\max_w \frac{w^T \Sigma_X w}{w^T S_{NN} w}$$

If we further assume that the nearest neighbors are isotropic, i.e.,

$$S_{NN} = \alpha I,$$

we get

$$\hat{w} = \arg\max_w w^T \Sigma_X w,$$

which leads to picking high-variance PCA directions.

Proof of Theorem 4.3:
If σ' is very small, for example,

$$\phi(\frac{-1}{\sigma'}) \ll \frac{1}{n},$$

then in Theorem 2.1, we can omit $\phi(\frac{-1}{\sigma'})$ and then we can get

$$C_r = \frac{D_{mean}}{D_{min}} \approx \frac{1}{(1 + \phi^{-1}(\frac{1}{n})\sigma')^{\frac{1}{p}}}.$$

Moreover, note that

$$\phi^{-1}(\frac{1}{n})\sigma' \gg \phi^{-1}(\phi(\frac{-1}{\sigma'}))\sigma' = -1$$

In other words, $\phi^{-1}(\frac{1}{n})\sigma'$ is a negative number with very small absolute value, so we can further apporximate the result as

$$(1 + \phi^{-1}(\frac{1}{n})\sigma')^{\frac{1}{p}} \approx 1 + \frac{1}{p}\phi^{-1}(\frac{1}{n})\sigma'.$$

If we have i.i.d assumption for each dimension, then

$$\sigma' = \frac{1}{d^{1/2}}\frac{\sigma_j}{\mu_j}.$$

And hence

$$\frac{D_{mean}}{D_{min}} = \frac{1}{1 + \phi^{-1}(\frac{1}{n})\frac{1}{p}\frac{1}{d^{1/2}}\frac{\sigma_j}{\mu_j}}.$$

References

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V.S. Locality-sensitive hashing scheme based on p-stable distributions. In *SOGC*, 2004.

Gionis, A., Indyk, P., and Motwani, R. Similarity search in high dimensions via hashing. In *VLDB*, 1999.