

Locality Sensitive Hashing
and
Approximate Nearest Neighbors

Lecture 1

Probability

Events, probabilities,
conditional probabilities,
union bound

Random variables,
expected value,
linearity of expectation
Markov's inequality

Algorithms

Approx. 3-SAT,

Big idea: we only want the average

$$\begin{array}{l} \text{Linearity of Expectation} \\ E[X+Y] = E[X] + E[Y] \quad (\text{stochastic variables}) \\ \text{"average sum = sum of averages"} \end{array}$$

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases} \quad (i=1, \dots, m)$

$$E[Y_i] = 7/8 \quad (1 \text{ clause case})$$

$$E[\text{\# clauses satisfied}] = E\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m E[Y_i] = 7/8 m$$

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$

$$Z = \sum_{i=1}^m Y_i = \text{\# clauses satisfied}$$

$$E[Z] = \frac{1}{2} E[Z | x_1 = \text{true}] + \frac{1}{2} E[Z | x_1 = \text{false}]$$

$\Rightarrow$ either $E[Z | x_1 = \text{true}]$ or $E[Z | x_1 = \text{false}]$ is $\geq E[Z]$!

QuickSort,

Random divide and conquer

① pick random 'pivot'

② Divide A into two parts



Expected running time of QuickSort

For each $i, j \in [n]$ w/ $i < j$,

$X_{ij} = \begin{cases} 1 & \text{if } i\text{th smallest element is compared to } j\text{th smallest} \\ 0 & \text{otherwise} \end{cases}$

$\sum_{i,j} X_{ij}$ = total # comparisons made by the algorithm
 $\approx$ total running time

$E[\sum_{i,j} X_{ij}]$ = average running time

$E[\sum_{i,j} X_{ij}] = \sum_{i,j} E[X_{ij}]$

now analyze $E[X_{ij}]$ in isolation

$$E[X_{ij}] = \Pr[X_{ij} = 1] = \Pr[i\text{th element is compared to } j\text{th}]$$

i th and j th elem selected before any element in between happens w/ probability $\frac{2}{j-i+1}$

$$E[X_{ij}] = \frac{2}{j-i+1}$$

$$E[\sum_{i,j} X_{ij}] = \sum_{i,j} E[X_{ij}] = \sum_{i,j} \frac{2}{j-i+1}$$

$$\leq 2n \left(\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} \right) \leq O(n \log n) \quad \text{with harmonic series}$$

Quickselect (HW)

Lecture 2

Probability

Hash Functions,

Universal Hash Functions

$h: [n] \rightarrow k$ s.t. for all pairs $i, j \in [n]$,
 $\Pr[h(i) = h(j)] \leq \frac{1}{k}$ (much weaker than ideal hash)

e.g. $h(x) = (ax + b) \bmod p \bmod k$

where • p prime, $> k$

• a in $\{1, \dots, p-1\}$ unif. random

• b in $\{0, \dots, p-1\}$ unif. random



Markov's Inequality

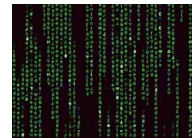
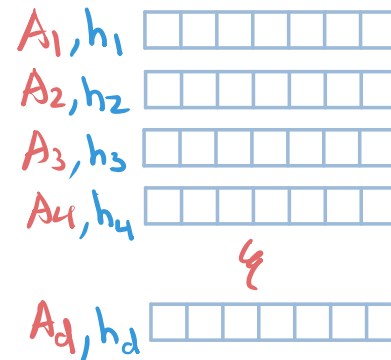
Markov's Inequality (special case)

Let $X \geq 0$ be nonnegative RV

$$\Pr[X \geq 2E[X]] \leq \frac{1}{2}$$

Algorithms

Frequency Estimation, Heavy Hitters (in streams)



estimate f_i w/
 $\min_{j=1, \dots, d} A_j[h_j(i)]$

Lecture 3 (last Tuesday)

Probability

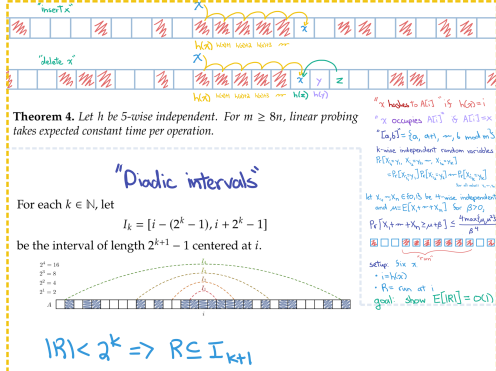
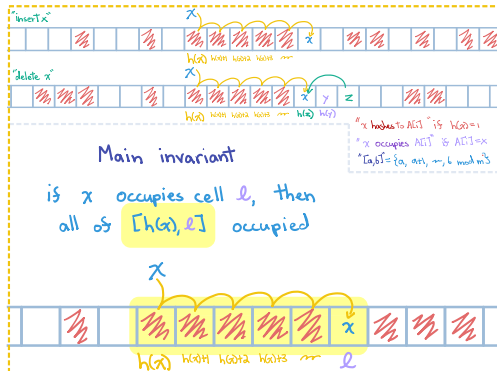
k-wise independence,
higher order moments
(generalizing universal hash
functions and Markov's inequality)

Algorithms

Hash tables w/
chaining +
linear probing

k-wise independent hash functions
 $h: [U] \rightarrow [m]$
s.t. for any k keys $x_1, x_2, \dots, x_k \in [U]$,
 k values $v_1, v_2, \dots, v_k \in [m]$,
 $P[h(x_1)=v_1, h(x_2)=v_2, \dots, h(x_k)=v_k] = \frac{1}{m^k}$
eg. $h(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1} \pmod p$
where $p \geq U$ prime,
 $a_0, a_1, \dots, a_{k-1} \in [p]$ indep, uniformly random

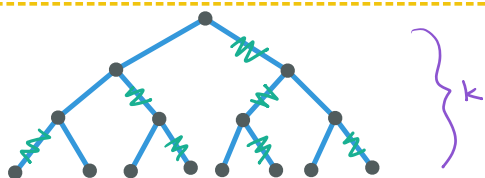
Important lemma
k-wise independent random variables
 $P[X_1=y_1, X_2=y_2, \dots, X_k=y_k]$
 $= P[X_1=y_1]P[X_2=y_2] \dots P[X_k=y_k]$
for all values $y_1, \dots, y_k$
let $X_1, \dots, X_n \in \{0,1\}$ be 4-wise independent
and $\mu = E[X_1 + \dots + X_n]$. then for $\beta > 0$,
 $P[X_1 + \dots + X_n \geq \mu + \beta] \leq \frac{4 \max\{\mu, \mu^3\}}{\beta^4}$



Lecture 4

Probability

Galton-Watson processes



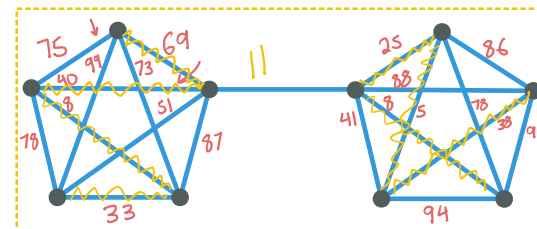
let T = complete binary tree of height k ,
w/ every edge deleted w/ prob $\leq 1/2$

Fact: $P[\text{some leaf connected to root}] \geq \frac{1}{k}$

"Galton-Watson process"

Algorithms

Random Contractions for min cut



Randomized contractions

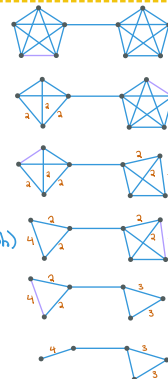
Input: $G=(V,E)$, weights $c \in \mathbb{R}_{>0}^E$
(expected)

① while $|E| > 1$

 Ⓐ sample $e \in E$ in proportion to c

 Ⓑ "contract" e

② return set of edges (in original graph)
contracted to remaining edge



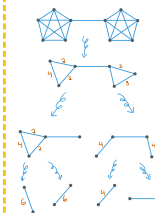
Amplification by branching

Instead of restarting at the beginning, restart partway through

High-level algorithm

① randomly contract until prob.
of ruining the min-cut is $\sim 1/2$

② Branch:
make two copies of current graph
recurse on both copies



Lecture 5

Probability

Multiplicative Chernoff Bound

Multiplicative Chernoff bound

let $X_1, \dots, X_n \in [0, 1]$ be independent

$$\mu = E[X_1 + \dots + X_n]$$

$$\epsilon \in (0, 1)$$

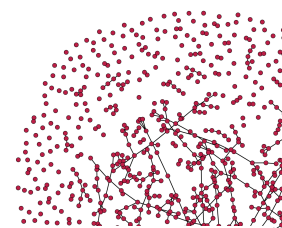
- $P[X_1 + \dots + X_n \geq (1 + \epsilon)\mu] \leq e^{-\epsilon^2 \mu / 3}$
- $P[X_1 + \dots + X_n \leq (1 - \epsilon)\mu] \leq e^{-\epsilon^2 \mu / 2}$

Algorithms (/ results)

Random graphs & the giant component

Erdős-Rényi graphs $G(n, p)$

- n vertices
- each edge sampled independently w/ probability p



Theorem 5.3. Consider a random graph $G \sim \mathcal{G}(n, p)$ for $p = c/n$, where c is a constant.

- If $c > 1$, then with high probability, there is exactly one connected component of G with $\Omega(n)$ vertices, and all other components have size $\leq O(\log n)$.
- For $c < 1$, then with high probability, all connected components of G has size $< O(\log n)$.

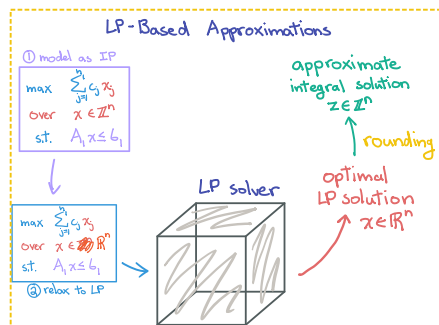
Lecture 6

Probability (& techniques)

Approximations via linear programming & randomized rounding

Linear programs (LPs)
Similar to IPs, except no integrality constraints

min/max $\sum_{j=1}^n c_j x_j$ linear objective
over $x \in \mathbb{R}^n$ linear integral variables
st. $A_1 x \leq b_1$ linear constraints
 $A_2 x = b_2$
 $A_3 x \geq b_3$



Continuous optimization vs Discrete optimization

randomization

Algorithms

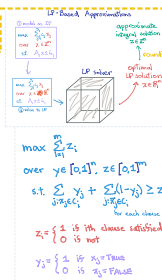
Max Sat Set Cover Covering Integer Programs

For each variable x_j , independently, randomly set $x_j = 1$ with probability p_j , and $x_j = 0$ otherwise.

$$E[\# \text{ sat. clauses}] \geq (1 - 1/e) \text{OPT}$$

k	oblivious	LP	average
1	1/2	$z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
2	3/4	$(3/4)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
3	7/8	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
4	15/16	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
5	31/32	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
...	...	...	...
k	$1 - 1/2^k$	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$

Can we get $\frac{3}{4}$ -APX?



Rounding LP solution
 $x \in [0, 1]^n \rightsquigarrow F \subseteq \{S_1, \dots, S_n\}$

For each set S_j , independently, take S_j with probability x_j .

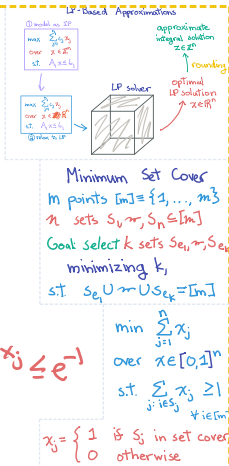
$$E[|F|] = \sum x_j \leq \text{OPT}$$

Fix a point i .

$$E[\# \text{ sets in } F \text{ covering } i] = \sum_{j \in S_i} x_j \geq z_i$$

$$P[F \text{ not cover } i] = \prod_{j \in S_i} (1 - x_j) \leq e^{-\sum_{j \in S_i} x_j} \leq e^{-z_i}$$

$$P[F \text{ cover } i] \geq 1 - e^{-1}$$



Lecture 7

Probability

Continuous Random Variables

Variance, Chebyshev's Variance Reduction

Median trick

Variance

$$\text{Var}(X) = E[(X - E[X])^2] = E[X^2] - (E[X])^2$$

Chebyshev's inequality

$$P[|X - E[X]| \geq d] \leq \frac{\text{Var}(X)}{d^2}$$

"median trick", "median of means"

- ① take mean to reduce variance, get constant prob. of error
- ② take median to amplify success prob. exponentially

Algorithms

Distinct Elements



Goal: estimate # of distinct elements in the stream

Suppose you had an ideal hash function

$$h: [m] \rightarrow [0, 1]$$

and the following facts:

let $Y_1, \dots, Y_k \in [0, 1]$ independent and uniformly random

$$X = \min\{Y_1, \dots, Y_k\}$$

$$\text{then } E[X] = \frac{1}{k+1}$$

$$\textcircled{1} X \leftarrow 1$$

② for each element e

$$\textcircled{a} X \leftarrow \min\{X, h(e)\}$$

$$/* E[X] = \frac{1}{k+1}, k = (\text{\# distinct elements}) */$$

$$\textcircled{3} \text{ return } \frac{1}{X} - 1$$

$$\text{recall } E[X] = \frac{1}{k+1}$$

$$\text{goal: show } |X - E[X]| \leq \frac{\epsilon}{k+1}$$

w/ good prob.

$$P[|X - E[X]| \geq \frac{\epsilon}{k+1}]$$

$$\leq \text{Var}(X) \cdot \left(\frac{k+1}{\epsilon}\right)^2 \leftarrow \text{Chebyshev's inequality}$$

where

$$\text{Var}(X) = \frac{k}{(k+1)^2(k+2)} \approx \frac{1}{k^2}$$

Q: how to reduce $\text{Var}(X)$?

we can get $(1 \pm \epsilon)$ -APX

w/ 10% error prob. in $O(\frac{1}{\epsilon^2})$ space

what about high probability?

① make q copies $Y_1, \dots, Y_q$ of

② return median of $Y_1, \dots, Y_q$

Algo correct $\Leftrightarrow \geq \frac{1}{2}$ of Y_i 's correct

let $Z = \# \text{correct } Y_i$'s $E[Z] = \frac{q}{2}$

$$P[Z \leq (1 - \frac{\epsilon}{4})E[Z]] \leq e^{-\left(\frac{\epsilon}{4}\right)^2 E[Z]/2} \leq \frac{1}{n^{O(n)}} \text{ for } q = O(\log n)$$

ideal hash $h: [m] \rightarrow [0, 1]$

let $Y_1, \dots, Y_k \in [0, 1]$

indep. and unifs. rand.

$X = \min\{Y_1, \dots, Y_k\}$

then $E[X] = \frac{1}{k+1}$

and $E[X^2] = \frac{2}{(k+1)(k+2)}$

① $X \leftarrow 1$

② for each element e

③ $X \leftarrow \min\{X, h(e)\}$

$/* E[X] = \frac{1}{k+1}, k = (\text{\# distinct elements}) */$

④ return $\frac{1}{X} - 1$

$\text{Var}(X) = E[(X - E[X])^2]$

$X_1, \dots, X_k$ indep. copies of X

$\bar{X} = \frac{1}{k}(X_1 + \dots + X_k)$

$\text{Var}(\bar{X}) = \frac{1}{k} \text{Var}(X)$

$P[X_1 + \dots + X_k \leq (1 - \epsilon)k\mu] \leq e^{-\epsilon^2 k \mu / 4}$

Lecture 8

Probability

Gaussian random variables

Gaussian random variables

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

mean variance

Def: the nicest, most pleasant, convenient random variable ever

Fact let $x \sim \mathcal{N}(\mu, \sigma^2)$, $d \in \mathbb{R}$.

$$\text{then } x+d \sim \mathcal{N}(\mu+d, \sigma^2)$$

$$dx \sim \mathcal{N}(d\mu, d^2\sigma^2)$$

Fact let $x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

$$\text{then } x_1 + x_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

Fact let $x \sim \mathcal{N}(0, 1)$, $t < \frac{1}{2}$

$$\text{then } E[e^{tx^2}] = \frac{1}{\sqrt{1-2t}}$$

Algorithms

Dimensionality reduction

Bag-of-words

literally any sentence is fine

any ↑ is literally sentence English
five encodes text in high dimensions

Lemma 8.8. Let $A \sim \mathcal{N}^{k \times d}$ be a random matrix where each coordinate A_{ij} is an independently drawn sample from $\mathcal{N}$. Let $\epsilon > 0$ be sufficiently small. For any vector x ,

$$\mathbb{P}\left[(1-\epsilon)\|x\|^2 \leq \frac{1}{k}\|Ax\|^2 \leq (1+\epsilon)\|x\|^2\right] \geq 1 - 2e^{-k/8}$$

$$k \left\{ \left[\frac{1}{k} \|Ax\|^2 \right] \leftarrow \alpha_i \right.$$

$$E[\|Ax\|^2] = \sum_{i=1}^k E[(Ax)_i^2] = k\|x\|^2$$

$$E[(\sum_{i=1}^k A_{ij} x_j)^2] = \|x\|^2$$

$$A_{ij} x_j \sim \mathcal{N}(0, x_j^2)$$

$$\sum_{j=1}^n A_{ij} x_j \sim \mathcal{N}(0, \|x\|^2)$$

$$(Ax) \sim \mathcal{N}^k(0, \|x\|^2)$$

$$\Pr[\|Ax\|^2 \geq k\|x\|^2 \cdot (1+\epsilon)] \leq e^{-\epsilon^2 k/8}$$

Dimensionality Reduction - high level

$O(\log n)$ dimensions suffice

where $n = \# \text{ points}$

$$\mathbb{R}^{\text{very large}} \xrightarrow{f(x)} \mathbb{R}^{O(\log n)}$$

One can map n points into $\mathbb{R}^{O(\log n/\epsilon^2)}$ while preserving all distances up to $(1 \pm \epsilon)$ -APX

Lecture 9

Probability

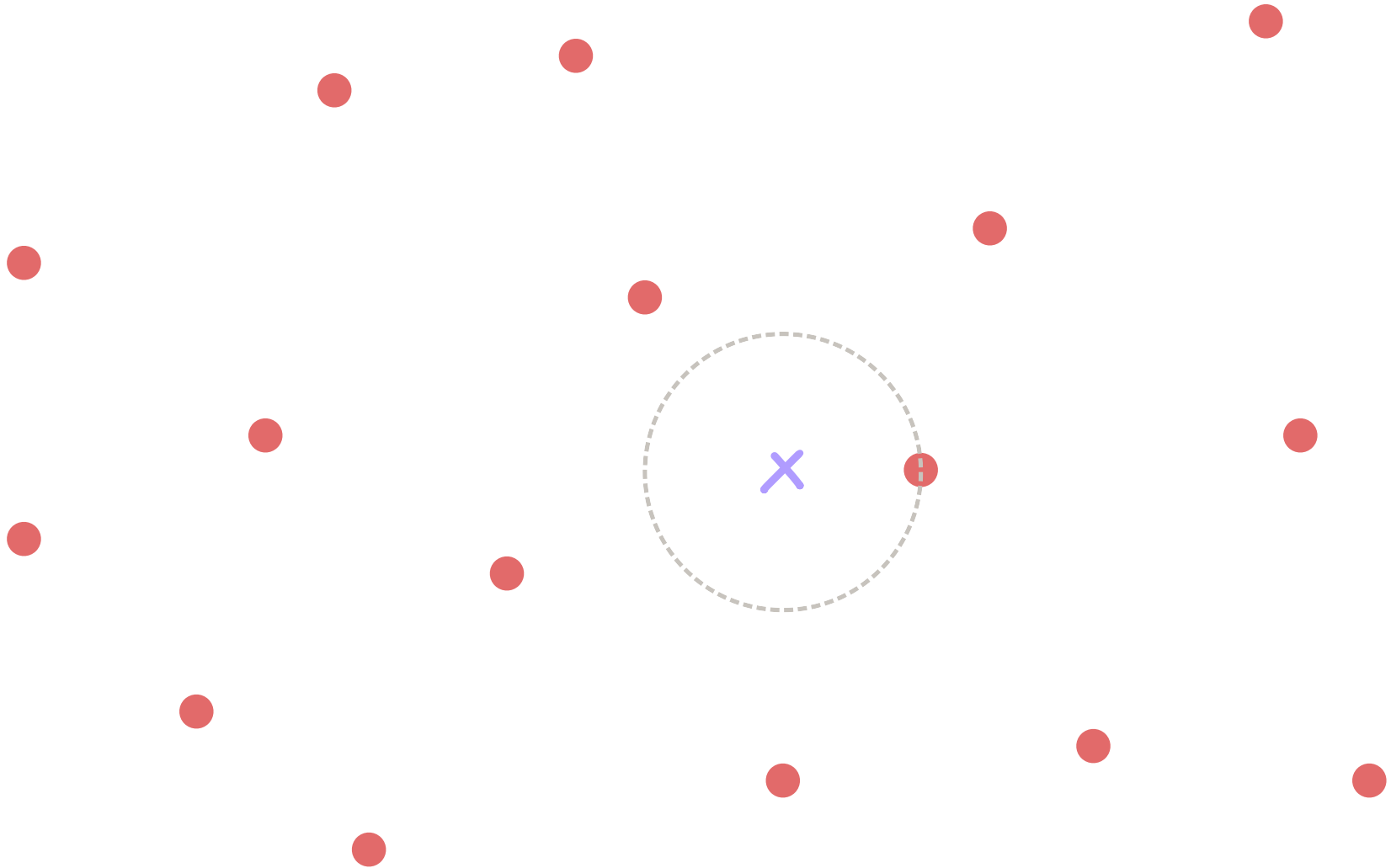
Locality-sensitive
hashing

Algorithms

Approximate
nearest neighbors

Nearest neighbor search

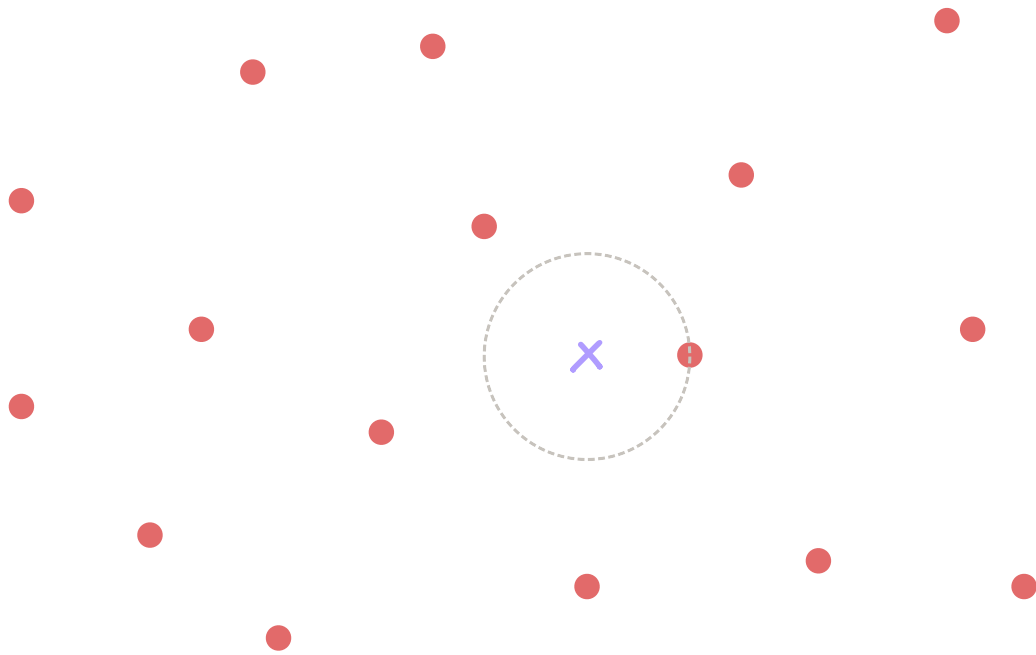
Input: n points $P \subset \mathbb{R}^d$



Query: given $q \in \mathbb{R}^d$, find closest point in P to q

Nearest neighbor search

Input: n points $P \subset \mathbb{R}^d$

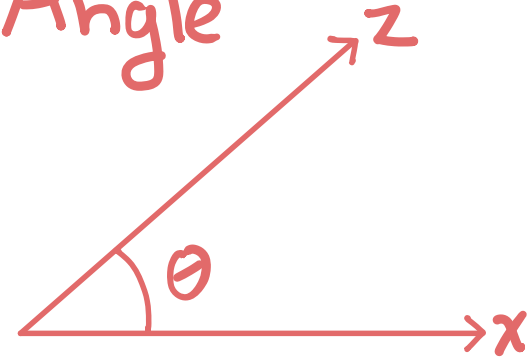


Types of distances

- Euclidean distance

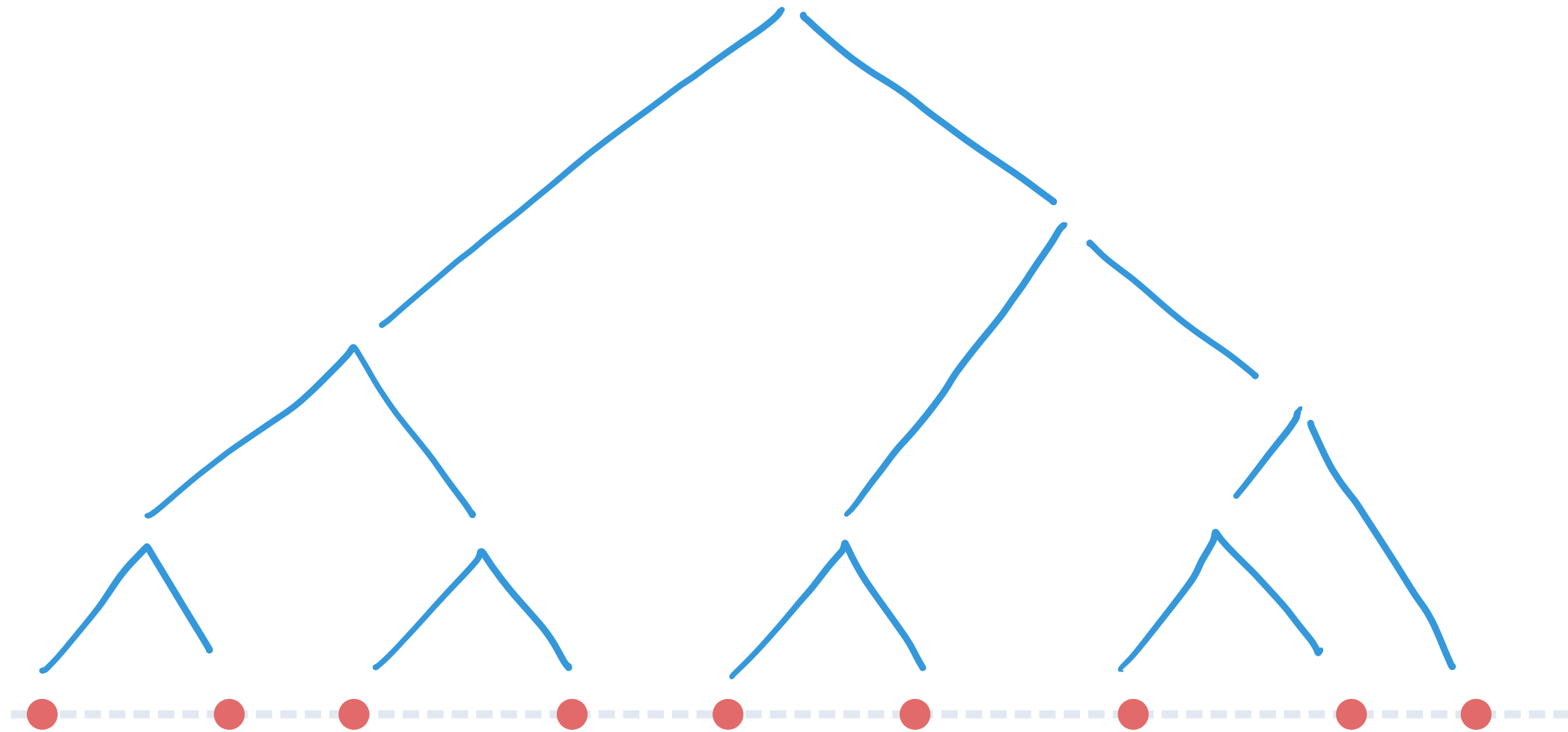
$$\|x - z\|_2 = \sqrt{\sum_i (x_i - z_i)^2}$$

- Angle

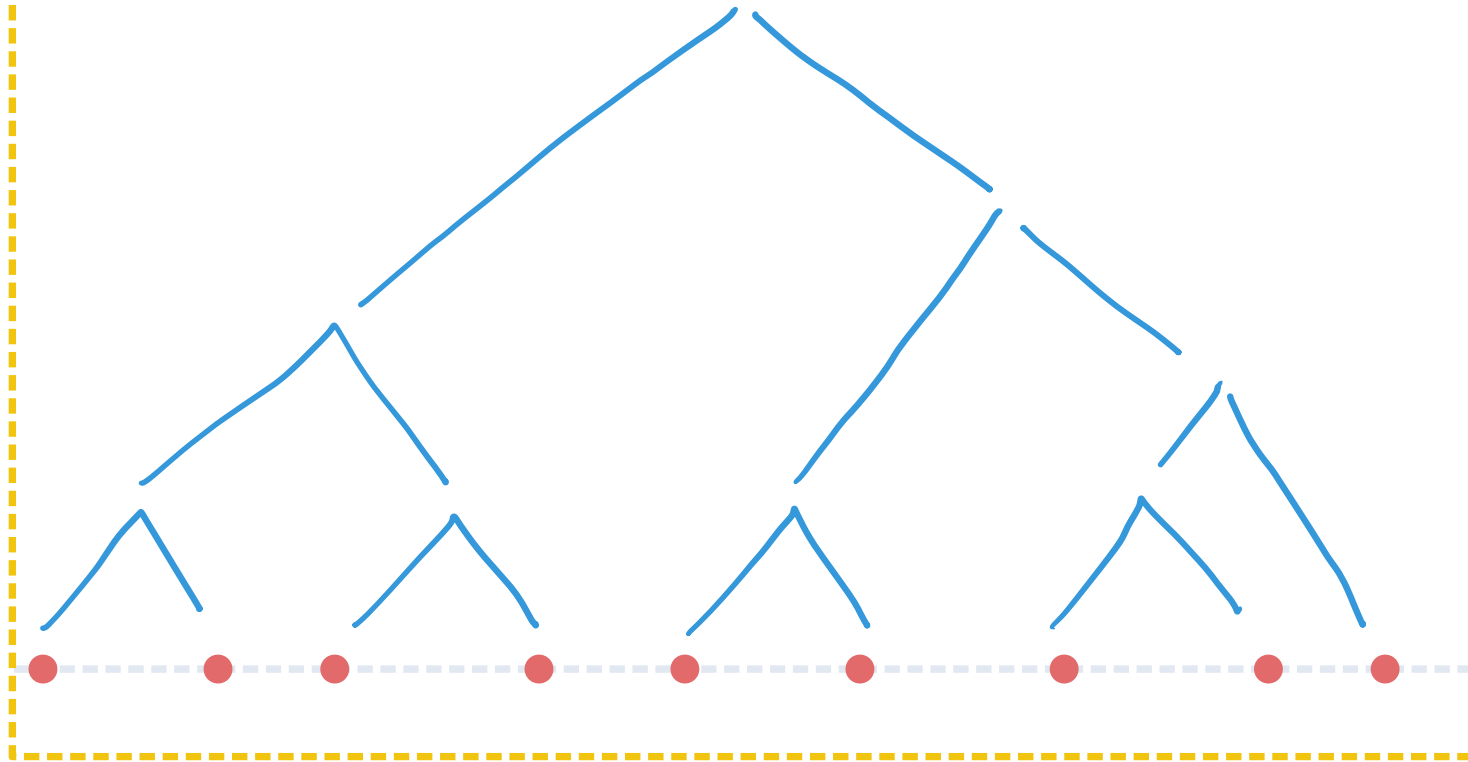


Query: given $q \in \mathbb{R}^d$, find closest point in P to q

Nearest neighbor search in 1D

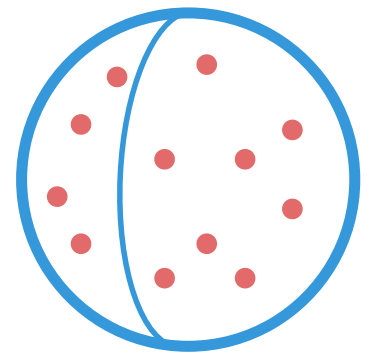
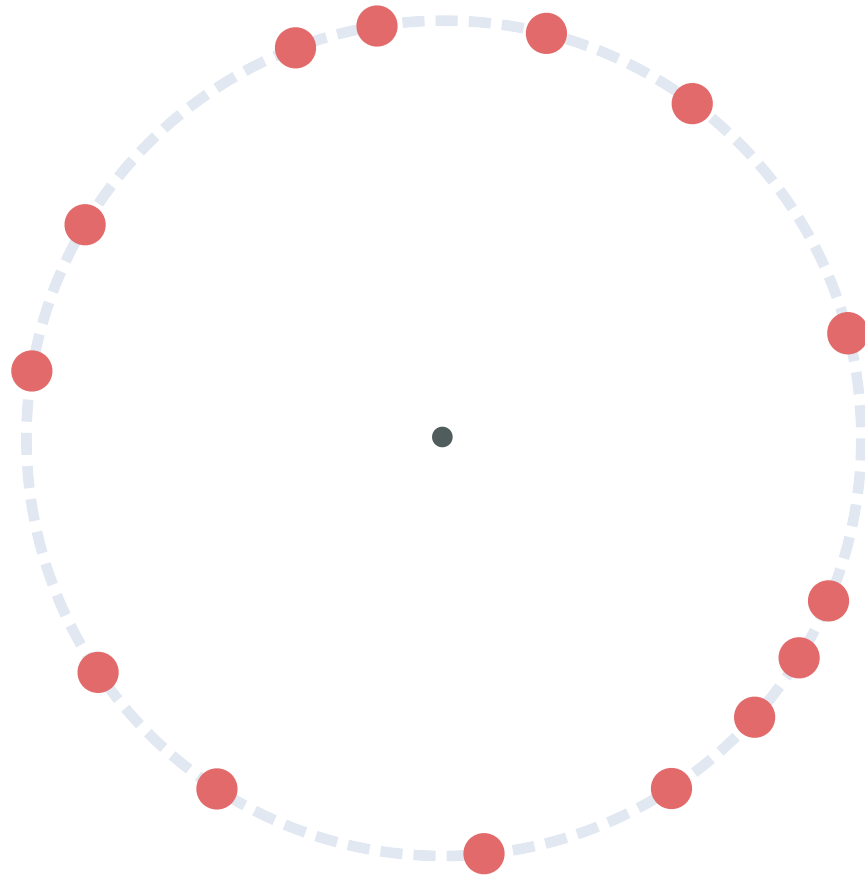


Nearest neighbor search in 1D

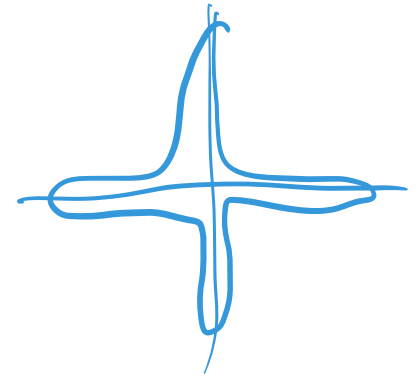


generalizes to k dimensions w/
kd-trees, quadtrees; $O(\log^k n)$ time

Angular LSH

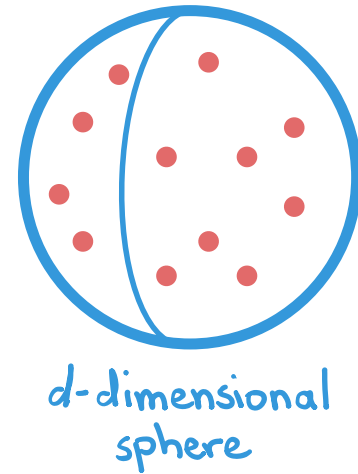
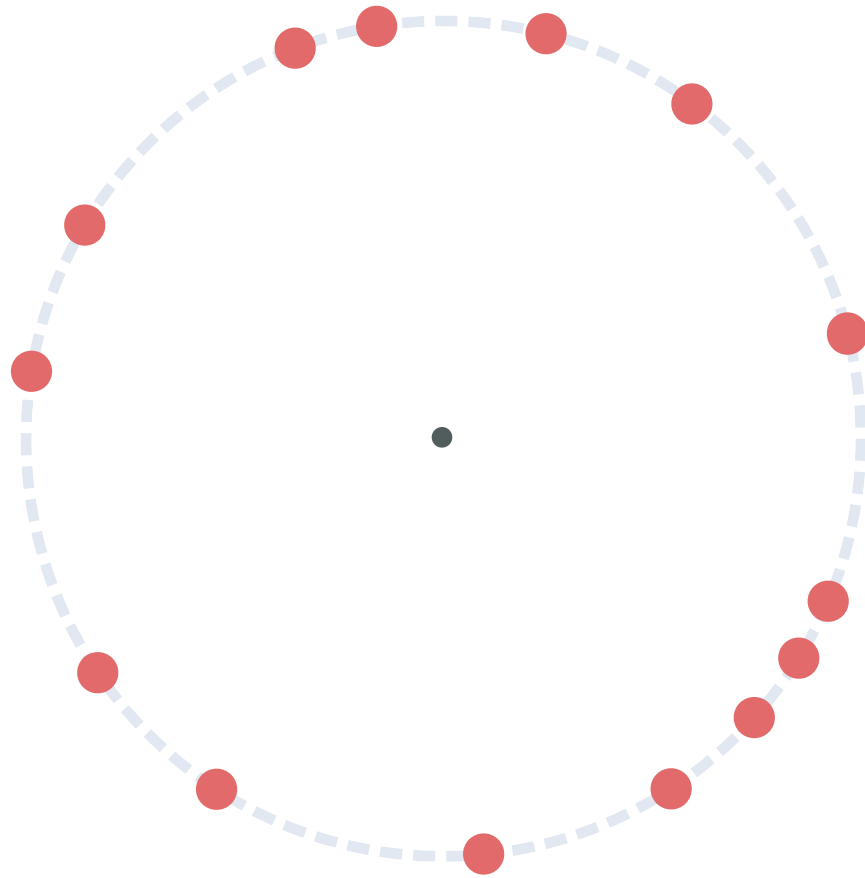


d-dimensional
sphere



Query for point w/ smallest angle

Angular LSH - parametrized and approximate



d-dimensional
sphere

Fix $\theta \in [0, \pi]$. given q , either:

(a) output $p \in P$ w/ $\angle(p, q) \leq 4\theta$

(b) declare no $p \in P$ w/ $\angle(p, q) \leq \theta$

Lemma let $x, y \in S^{d-1}$

$g \sim \mathcal{N}^d$

then

$$P[\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)] = \frac{\angle(x, y)}{\pi}$$

proof

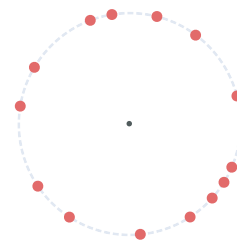
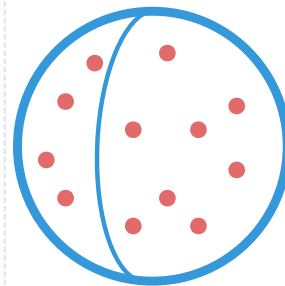
Angular LSH

parametrized and approximate

Fix $\theta \in [0, \pi]$. given q , either:

(a) output $p \in P$ w/ $\angle(p, q) \leq 4\theta$

(b) declare no $p \in P$ w/ $\angle(p, q) \leq \theta$



Fix $\theta \in [0, \pi]$. given q , either:

(a) output $p \in P$ w/ $\angle(p, q) \leq 4\theta$

(b) declare no $p \in P$ w/ $\angle(p, q) \leq \theta$

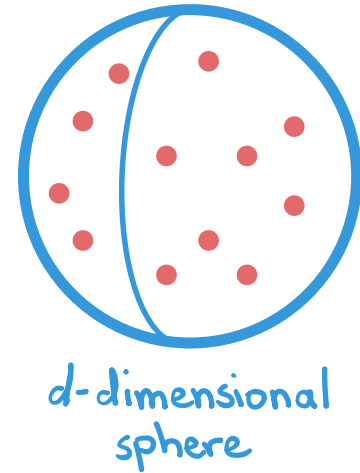
Lemma let $\bullet x, y \in S^{d-1}$
 $\bullet g \sim \mathcal{N}^d$

then

$$P[\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)] = \frac{\angle(x, y)}{\pi}$$

define $h: S^{d-1} \rightarrow \{-1, 1\}^k$ by

$$h_i(x) = \text{sign}(\langle g_i, x \rangle) \quad \text{w/ } g_i \sim \mathcal{N}^d$$



Algo: group P by hash code $h(x)$

to query y , scan all points $x \in P$ w/ $h(x) = h(y)$

Fix $\theta \in [0, \pi]$. given q , either:

(a) output $p \in P$ w/ $\angle(p, q) \leq 4\theta$

(b) declare no $p \in P$ w/ $\angle(p, q) \leq \theta$

Lemma let $\bullet x, y \in S^{d-1}$
 $\bullet g \sim \mathcal{N}^d$

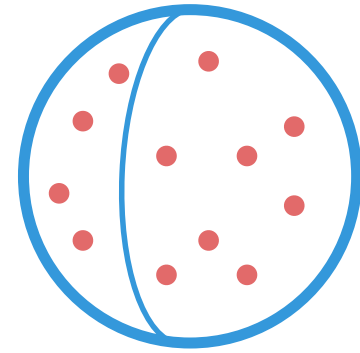
then

$$P[\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)] = \frac{\angle(x, y)}{\pi}$$

define $h: S^{d-1} \rightarrow \{-1, 1\}^k$ by

$$h_i(x) = \text{sign}(\langle g_i, x \rangle) \quad \text{w/ } g_i \sim \mathcal{N}^d$$

$\angle(x, y) < \theta$:



d-dimensional
sphere

Fix $\theta \in [0, \pi]$. given q , either:

(a) output $p \in P$ w/ $\angle(p, q) \leq 4\theta$

(b) declare no $p \in P$ w/ $\angle(p, q) \leq \theta$

Lemma let $\bullet x, y \in S^{d-1}$
 $\bullet g \sim \mathcal{N}^d$

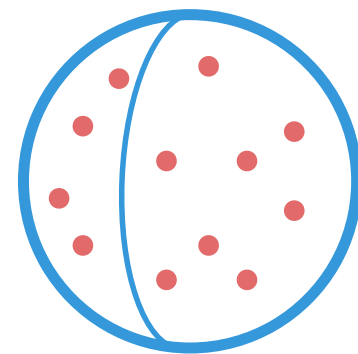
then

$$P[\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)] = \frac{\angle(x, y)}{\pi}$$

define $h: S^{d-1} \rightarrow \{-1, 1\}^k$ by

$$h_i(x) = \text{sign}(\langle g_i, x \rangle) \quad \text{w/ } g_i \sim \mathcal{N}^d$$

$$\angle(x, y) > 2\theta$$



d-dimensional
sphere

Fix $\theta \in [0, \pi]$. given q , either:

(a) output $p \in P$ w/ $\angle(p, q) \leq 4\theta$

(b) declare no $p \in P$ w/ $\angle(p, q) \leq \theta$

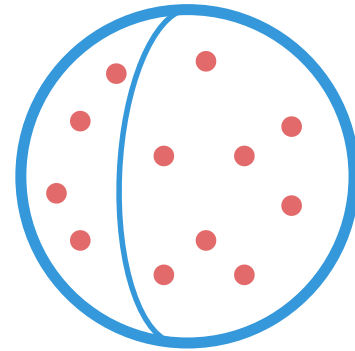
Lemma let $\bullet x, y \in S^{d-1}$
 $\bullet g \sim \mathcal{N}^d$

then

$$P[\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)] = \frac{\angle(x, y)}{\pi}$$

define $h: S^{d-1} \rightarrow \{-1, 1\}^k$ by

$$h_i(x) = \text{sign}(\langle g_i, x \rangle) \quad \text{w/ } g_i \sim \mathcal{N}^d$$



$$k = \pi \log(n) / 2\theta$$

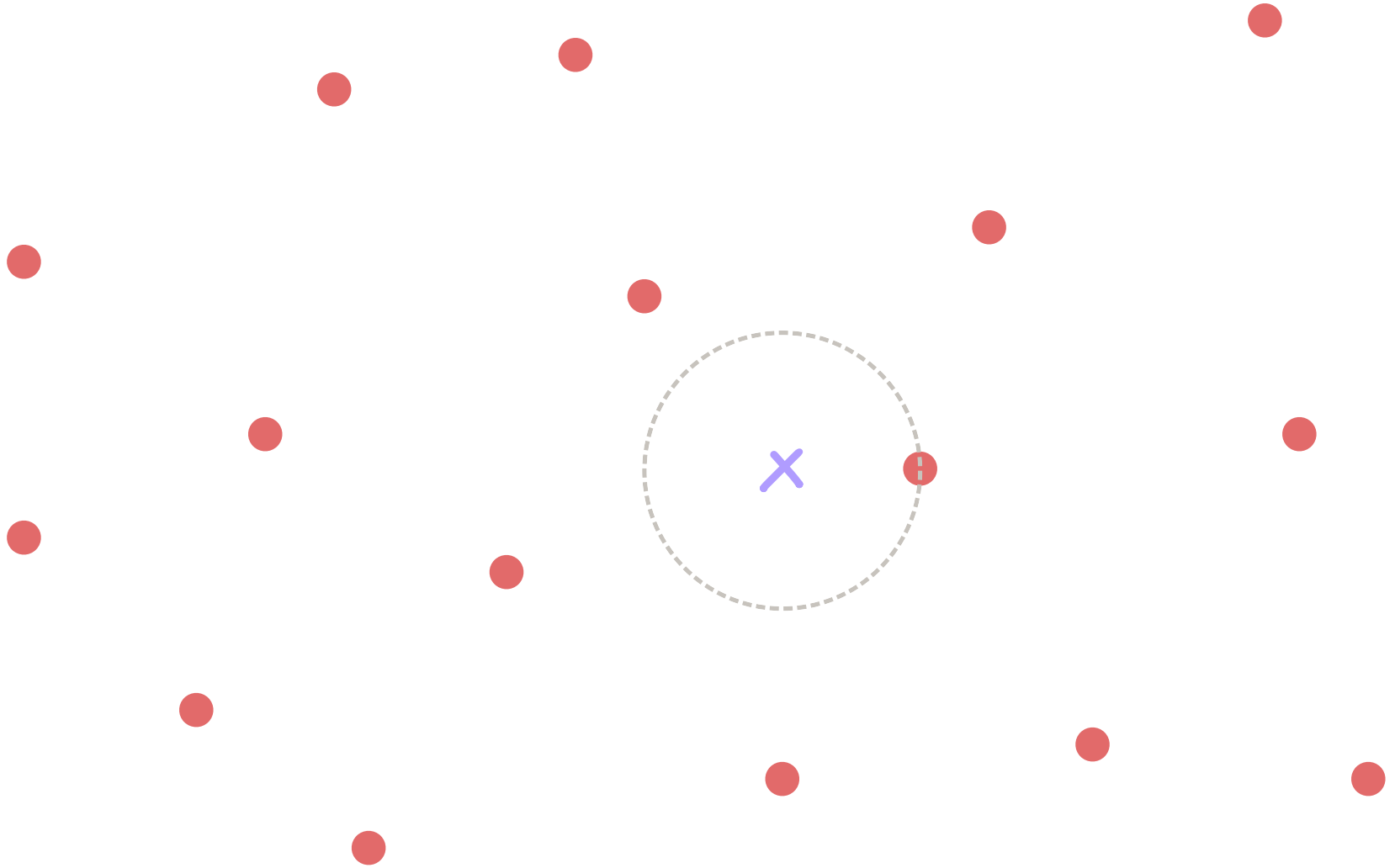
d-dimensional
sphere

when querying y :

- if there is $x \in P$ w/ $\angle(x, y) \leq \theta$, then it collides
w/ probability $\gtrsim 1/\sqrt{n}$

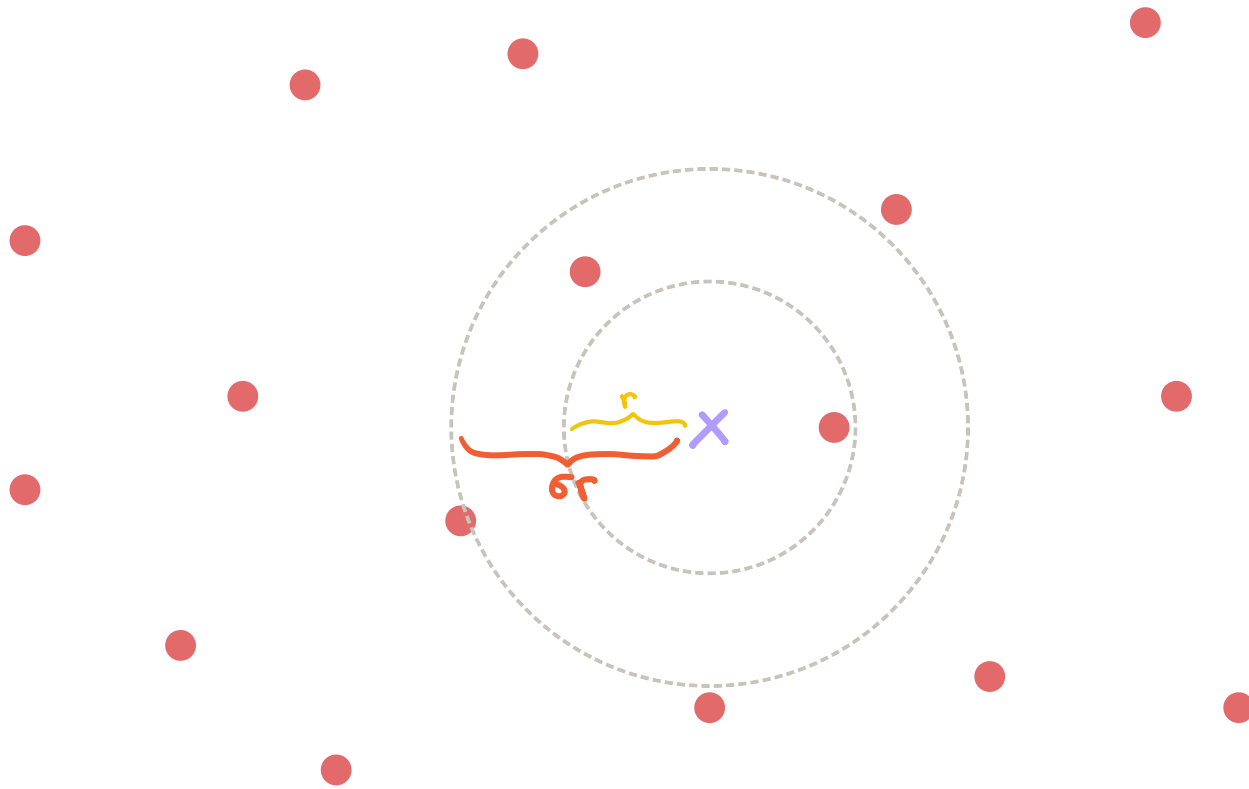
- expect to collide w/ at most one point $x \in P$
w/ $\angle(x, y) > 2\theta$

Euclidean LSH



Query: given $q \in \mathbb{R}^d$, find closest point in P to q

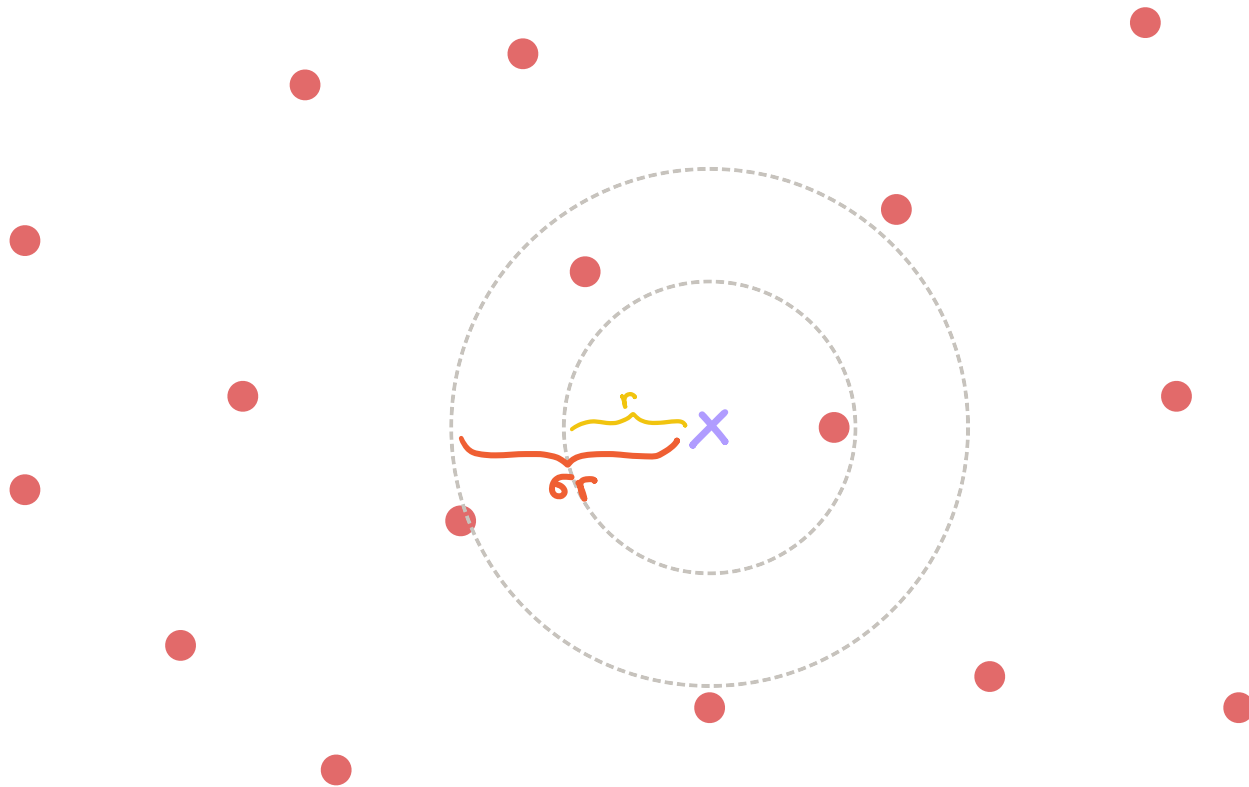
Euclidean LSH - parametrized and approximate



Fix $r > 0$, $\sigma > 1$. given query $y \in \mathbb{R}^d$ either:

- return $x \in P$ w/ $\|x - y\| \leq \sigma r$
- declare no $x \in P$ w/ $\|x - y\| \leq r$

Euclidean LSH - parametrized and approximate

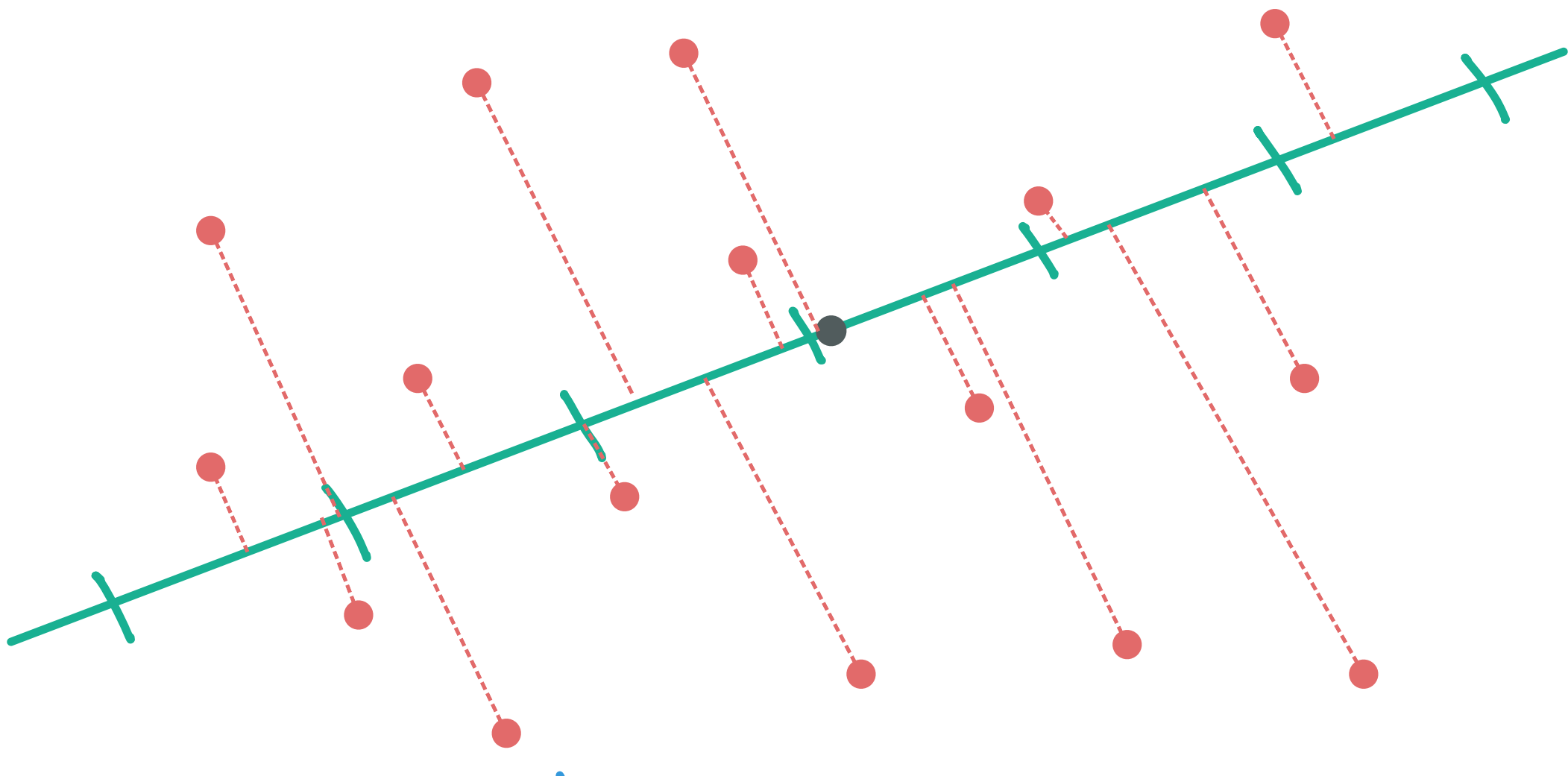


Fix $r=1$, $\sigma > 1$. given query $y \in \mathbb{R}^d$ either:

- return $x \in P$ w/ $\|x - y\| \leq \sigma$
- declare no $x \in P$ w/ $\|x - y\| \leq 1$

Random line embedding & Buckets

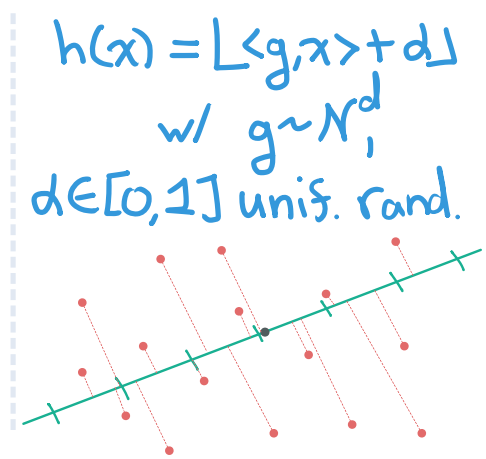
$$h(x) = \lfloor \langle g, x \rangle + d \rfloor \quad \text{w/ } g \sim \mathcal{N}^d, \\ d \in [0, 1] \text{ unif. rand.}$$



Lemma let $x, y \in \mathbb{R}^d$

$$P[h(x) = h(y) | g] = \max \{0, 1 - |\langle x - y, g \rangle|\}$$

Proof

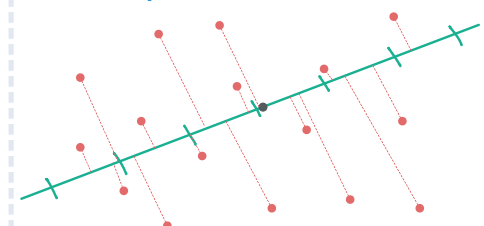


Lemma let $x, y \in \mathbb{R}^d$, $f(t)$ be $N(0,1)$ -density function

$$P[h(x)=h(y)] = 2 \int_0^1 (1-t) f(t\|x-y\|) dt$$

proof

$h(x) = \lfloor \langle g, x \rangle + d \rfloor$
 w/ $g \sim N^d$,
 $d \in [0,1]$ unif. rand.



Lemma let $x, y \in \mathbb{R}^d$
 $P[h(x)=h(y) | g] =$
 $\max \{0, 1 - |\langle x-y, g \rangle|\}$

Lemma let $x, y \in \mathbb{R}^d$, $f(t)$ be $N(0,1)$ -density function

$$P[h(x)=h(y)] = 2 \int_0^1 (1-t) f(t\|x-y\|) dt$$

let

$$p = 2 \int_0^1 (1-t) f(t) dt$$

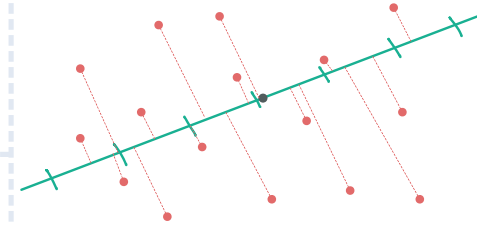
($\leq$ collision probability when $\|x-y\| \leq 1$)

$\approx .368746$

$$q = 2 \int_0^1 (1-t) f(\sigma t) dt$$

($\geq$ collision probability when $\|x-y\| \geq \sigma$)

$h(x) = \lfloor \langle g, x \rangle + d \rfloor$
w/ $g \sim N^d$,
 $d \in [0,1]$ unif. rand.

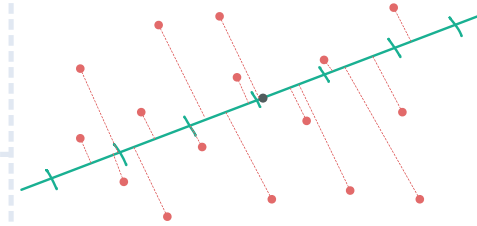


Lemma let $x, y \in \mathbb{R}^d$
 $P[h(x)=h(y) | g] =$
 $\max \{0, 1 - \langle x-y, g \rangle\}$

Lemma let $x, y \in \mathbb{R}^d$, $f(t)$ be $N(0,1)$ -density function

$$P[h(x)=h(y)] = 2 \int_0^1 (1-t) f(t\|x-y\|) dt$$

$h(x) = \lfloor \langle g, x \rangle + d \rfloor$
 w/ $g \sim N^d$,
 $d \in [0,1]$ unif. rand.



$$p = 2 \int_0^1 (1-t) f(t) dt$$

($\leq$ collision probability when $\|x-y\| \leq 1$)

$$q = 2 \int_0^1 (1-t) f(\sigma t) dt$$

($\geq$ collision probability when $\|x-y\| \geq \sigma$)

Lemma let $x, y \in \mathbb{R}^d$
 $P[h(x)=h(y) | g] =$
 $\max\{0, 1 - \langle x-y, g \rangle\}$

Algo: group P by $h(x)$

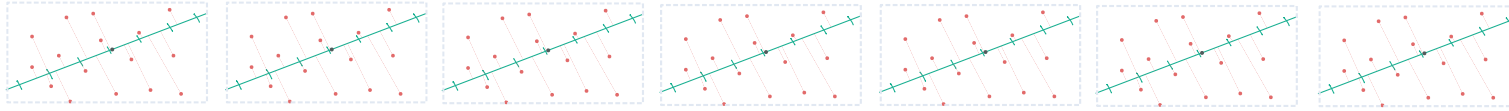
to query y , scan all $x \in P$ w/ $h(x)=h(y)$

$$\text{Prob. success} = p \quad E[\text{time}] = 1 + qn$$

For constant prob. success:

Amplification

$$h: \mathbb{R}^d \rightarrow \mathbb{R}^k \quad \text{w/} \quad h_i(x) = \lfloor \langle g_i, x \rangle + d_i \rfloor \quad i=1, \dots, k$$



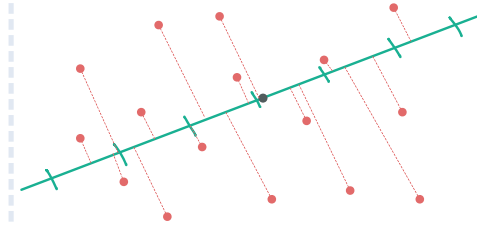
$$P[h(x)=h(y)] = \left(2 \int_0^1 (1-t) f(t \|x-y\|) dt \right)^k$$

$$P[h(x)=h(y)] \geq p^k \quad \text{when} \quad \|x-y\| \leq 1$$

$$P[h(x)=h(y)] \leq q^k \quad \text{when} \quad \|x-y\| \geq 6$$

$$h(x) = \lfloor \langle g, x \rangle + d \rfloor$$

w/ $g \sim \mathcal{N}^d$,
 $d \in [0,1]$ unif. rand.



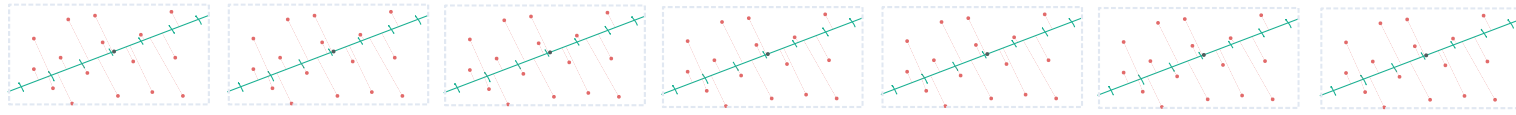
$$p = 2 \int_0^1 (1-t) f(t) dt$$

($\leq$ collision probability when $\|x-y\| \leq 1$)

$$q = 2 \int_0^1 (1-t) f(6t) dt$$

($\geq$ collision probability when $\|x-y\| \geq 6$)

$$h: \mathbb{R}^d \rightarrow \mathbb{R}^k \quad \text{w/} \quad h_i(x) = \lfloor \langle g_i, x \rangle + d_i \rfloor \quad i=1, \dots, k$$



$$P[h(x) = h(y)] \geq p^k \quad \text{when} \quad \|x - y\| \leq 1$$

$$P[h(x) = h(y)] \leq q^k \quad \text{when} \quad \|x - y\| \geq \sigma$$

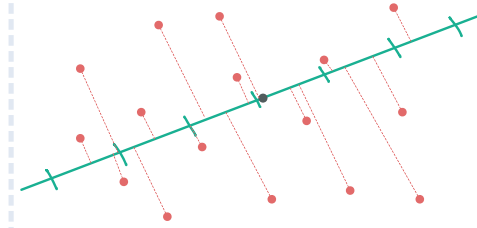
Algo: group P by $h(x)$
 to query y , scan all $x \in P$ w/ $h(x) = h(y)$

$$\text{Prob. success} = \quad E[\text{time}] =$$

For constant prob. success:

$$h(x) = \lfloor \langle g, x \rangle + d \rfloor$$

w/ $g \sim \mathcal{N}^d$,
 $d \in [0, 1]$ unif. rand.



$$p = 2 \int_0^1 (1-t) f(t) dt$$

($\leq$ collision probability when $\|x - y\| \leq 1$)

$$q = 2 \int_0^1 (1-t) f(\sigma t) dt$$

($\geq$ collision probability when $\|x - y\| \geq \sigma$)

$$P[h(x)=h(y)] \geq p^k \text{ when } \|x-y\| \leq 1$$

$$P[h(x)=h(y)] \leq q^k \text{ when } \|x-y\| \geq \sigma$$

Algo: group P by $h(x)$

to query y , scan all $x \in P$ w/ $h(x) = h(y)$

$$E[\text{time}] = k\left(\frac{1}{p}\right)^k + \left(\frac{q}{p}\right)^k n$$

$$= \left(\frac{1}{p}\right)^k (k + q^k n)$$

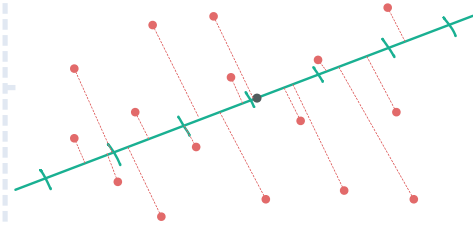
$$k = \frac{\log(n)}{\log(1/q)} \Rightarrow q^k n = 1,$$

$$k\left(\frac{1}{p}\right)^k = \frac{\log(n)}{\log(1/q)} n^{p(\sigma)}$$

$$\text{w/ } p(\sigma) = \frac{\log(1/p)}{\log(1/q)}$$

$$h(x) = \lfloor \langle g, x \rangle + d \rfloor$$

w/ $g \sim \mathcal{N}^d$,
 $d \in [0, 1]$ unif. rand.

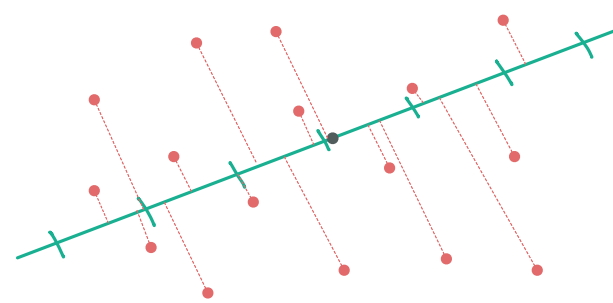


$$p = 2 \int_0^1 (1-t) f(t) dt$$

($\leq$ collision probability when $\|x-y\| \leq 1$)

$$q = 2 \int_0^1 (1-t) f(\sigma t) dt$$

($\geq$ collision probability when $\|x-y\| \geq \sigma$)



Theorem 9.6. One can compute a ρ -approximate nearest neighbor w/r/t Euclidean distance with high probability in $\tilde{O}(n^{\rho(\sigma)})$ randomized time, where

$$\rho(\sigma) = \frac{\log(1/p)}{\log(1/q)}, \quad p = 2 \int_0^1 (1-t)f(t) dt \approx .368746, \quad q = 2 \int_0^1 (1-t)f(\sigma t) dt,$$

and $f(t) = e^{-t^2/2}/\sqrt{2\pi}$ is the density function of the standard Gaussian.

For $\sigma \leq 1.2649$, $\rho(\sigma) \leq \frac{1}{\sigma^2}$

$\Rightarrow (1+\epsilon)$ -APX NN in $\approx \tilde{O}(n^{1-2\epsilon+\epsilon^2})$ time

for $\epsilon > 0$ suff. small

$\nearrow$ sublinear!

Chapter 9

Locality sensitive hashing and approximate nearest neighbors

9.1 Nearest neighbor search

The previous discussion on dimensionality reduction introduced many natural settings where data can be represented as high-dimensional vectors. Surprisingly, by simply projecting onto randomized Gaussian vectors, one can embed an n -point data set into $O(\log(n)/\epsilon^2)$ dimensions while preserving all pairwise distances up to an $(1 \pm \epsilon)$ -multiplicative factor.

A natural query in many domains is *similarity search* where we want to preprocess a large collection of items such that, given a query in the form of an item, we can quickly retrieve the “most similar” item in our collection by some metric. The “most similar” item is sometimes called the *nearest neighbor*.

A simple example might be where we have a collection of n real numbers. Given another number as a query, we want to retrieve the closest number in our collection to the query number. For this problem, we would sort our collection of numbers and store them in an array. Given a query $q \in \mathbb{R}$, we run binary search on our array to find the first number smaller and bigger than q . We return the closer of the two. This approach would take $O(\log(n))$ time. The above approaches extends to low-dimensional data sets, via *quadtrees* and multi-dimensional *range trees*. Both of these approaches scale exponentially in the d . If the data lives in very few dimensions - like 3 dimensions, as in many physical situations - this is still very good.

Continuing the previous discussion on dimensionality reduction, here we are interested in nearest neighbor search with *high dimensional data*. We have a collection of n points $P \subset \mathbb{R}^d$ that we can preprocess to build some kind of data structure. Each query is in the form of another point $z \in \mathbb{R}^d$, and the goal is to output the point $x \in P$ closest to z by some metric. Here we will consider two settings.

1. The *Euclidean distance*, $\|x - z\|$.
2. The *angle* between x and z (as a real value between 0 and π).

If all the vectors in P are normalized to have the same length, then the nearest neighbor in Euclidean distance and angle are the same. However, we will consider *approximations* for this problem, in which case there is a difference between the two metrics. We consider angular nearest neighbors in section 9.2 and Euclidean nearest neighbors in section 9.3.

A natural idea, at least for Euclidean distance, is to use the dimensionality reduction techniques from the previous discussion to reduce to $O(\log(n))$ dimensions, and apply the low-dimensional data structures. But the exponential dependence in the dimension means that even $O(\log(n))$ dimensions – which is fairly small by our standards – will still require $n^{O(1)}$ time and space.

From nearest neighbor to close enough. Fix $\sigma > 1$. Enumerating distances to the nearest neighbor by powers of σ allows us to reduce to the following problem with only logarithmic overhead:

Let a target distance $r > 0$ be fixed. Given a query point z , either output a point at distance $\leq \sigma r$, or declare that all points have distance $\geq r$.

It also suffices, up to logarithmic factors, to succeed with constant probability. We can amplify to high probability with repetition.

When hash collisions are good. The algorithms we develop are based on hashing, although the intuition is the opposite of previous discussions on hashing. Previously, in applications such as heavy hitters and hash tables, hashing was used to randomly *spread out* the elements. Here, we will design hash functions where closer points are *more likely to collide*. This technique is called *locality sensitive hashing* [IM98].¹ The high level strategy is to build a hash table over all the input set P using locality sensitive hash functions. Given a query point z , we hash z and hope to find the nearest neighbor in the same hash bucket. In the algorithms we discuss, this algorithm will only succeed with limited probability. We amplify by building many such hash tables independently. On a query point z , we hash z into all of the hash tables, and return the first point we find that is close enough.

9.2 Locality Sensitive Hashing for Angles

We consider the setting of *angular distance*, which is commonly used when the point set P lies on the hypersphere $\mathbb{S}^{d-1} = \{x \in \mathbb{R}^d : \|x\| = 1\}$. In this case a common measure of distance is the *angle* between points. For two points x and y , let $\angle(x, y) \in [0, \pi]$ denote the angle between the points x and y between 0 and π . We may assume that

¹Here the author would like to propose the term *clashing*.

all the points in our data set P , as well as any query point, is normalized to lie on $\mathbb{S}^{d-1}$. We describe an LSH scheme due to [Cha02].

Let $\theta \in [0, \pi]$ be a fixed angle. In this discussion, given a query point y , our goal is to either output a point $x \in P$ with $\angle(x, y) \leq 4\theta$, or decide that there are no points $x \in P$ with $\angle(x, y) \leq \theta$.

Let $k \in \mathbb{N}$ be a parameter TBD. We will generate a k -ary hash function $h : \mathbb{S}^{d-1} \rightarrow \{-1, 1\}^k$ where each coordinate is generated by splitting $\mathbb{S}^{d-1}$ in half along a random hyperplane. For each coordinate $i \in [k]$, let $g_i \sim \mathcal{N}^d$ be a random Gaussian vector. We define a hash function $h : \mathbb{S}^{d-1} \rightarrow \{-1, 1\}^k$ by

$$h_i(x) = \text{sign}(\langle g_i, x \rangle)$$

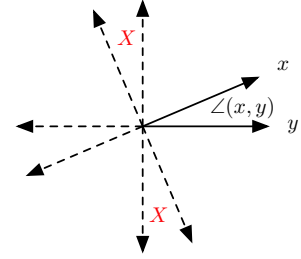
for each coordinate i .

We will leverage the following fact that is entirely based on the rotational invariance of $\mathcal{N}^d$.

Lemma 9.1. *Let $x, y \in \mathbb{S}^{d-1}$, and let $g \sim \mathcal{N}^d$. Then*

$$\mathbf{P}[\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)] = \frac{\angle(x, y)}{\pi}.$$

Proof. Recall that $\mathcal{N}^d$ is *rotationally symmetric*. Rotating, we may assume that x and y are spanned by the first two coordinates, and consider the simpler (to visualize) setting where $x, y \in \mathbb{R}^2$ and $g \sim \mathcal{N}^2$, as on the right.



We are particularly interested in the angle of the random vector g . Since $\mathcal{N}^2$ is rotationally symmetric, it is equally likely to take any particular angle with equal probability. Out of a total of 2π , there are two regions of angle $\angle(x, y)$ in which $\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)$, here marked with a red X . $\square$

Now, let $k = \frac{\pi \log(n)}{2\theta}$. If $\angle(x, y) < \theta$, then

$$\mathbf{P}[h(x) = h(y)] = \left(1 - \frac{\angle(x, y)}{\pi}\right)^k \approx e^{-\frac{\angle(x, y)}{\pi}k} = e^{-\log(n)/2} = \frac{1}{\sqrt{n}}.$$

On the other hand, if $\angle(x, y) > 2\theta$, then

$$\mathbf{P}[h(x) = h(y)] = \left(1 - \frac{\angle(x, y)}{\pi}\right)^k \leq e^{-\angle(x, y)k/\pi} \leq e^{-\log(n)} = \frac{1}{n}.$$

Thus, when querying a point y :

1. If there is a point $x \in P$ with $\angle(x, y) \leq \theta$, then it will collide with probability (approximately) $\geq 1/\sqrt{n}$.
2. We expect to collide with at most 1 point $x \in P$ such that $\angle(x, y) \geq 2\theta$.

If we repeat the experiment $O(\sqrt{n} \log(n))$ times, then with high probability we will find a close neighbor if one exists. The query time is $O(d\sqrt{n} \log(n))$ in expectation, because each time we expect 1 “junk” neighbor on average.

Theorem 9.2. *With $O(dn^{3/2} \text{polylog}(n))$ preprocessing time and space, one can query for 2-approximate nearest neighbors w/r/t angular distance in $O(d\sqrt{n} \text{polylog}(n))$ expected randomized time and with high probability.*

9.3 Locality Sensitive Hashing for Euclidean Distance

We now consider σ -approximate nearest neighbors in Euclidean metrics, for fixed $\sigma > 1$. As mentioned above, it suffices to consider the simpler setting where there is a fixed target distance r . We want to preprocess a set of n points $P \subseteq \mathbb{R}^d$ as to quickly answer the following query with constant probability of success:

Given a query point $z \in \mathbb{R}^d$, either find a point $x \in P$ with $\|x - z\| \leq \sigma r$,
or declare that there are no points $x \in P$ with $\|x - z\| \leq r$.

Our goal is to develop a locality-sensitive hash function² $h : \mathbb{R}^d \rightarrow \mathbb{Z}^k$ for Euclidean distance. We will design h such that the collision probabilities of two points $x, y \in \mathbb{R}^d$ depend only on the ratio $\|x - y\|$ to r . Therefore, we can also rescale and assume that $r = 1$.

Given a query point $z \in \mathbb{R}^d$, either find a point $x \in P$ with $\|x - z\| \leq \sigma$,
or declare that there are no points $x \in P$ with $\|x - z\| \leq 1$.

9.3.1 Random line embeddings and buckets

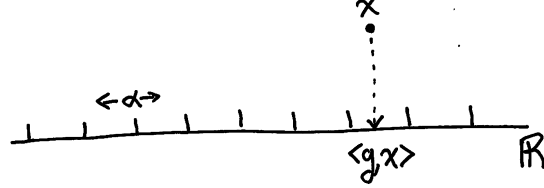
In this section, we first define a hash function $h : \mathbb{R}^d \rightarrow \mathbb{Z}$ that outputs a single hash code. Later we will consider a k -coordinate hash where each coordinate is an independent copy of the single coordinate hash we consider here.

We define $h : \mathbb{R}^d \rightarrow \mathbb{Z}$ by the function

$$h(x) = \lfloor \langle g, x \rangle + \alpha \rfloor,$$

²a *clash function*?

where $g \sim \mathcal{N}^d$ and $\alpha \in [0, 1]$ uniformly at random. Geometrically, we can interpret $h(x)$ as randomly projecting x onto a line, and then bucketing the points in intervals of length 1. The random value $\alpha \in [0, 1]$ translates the buckets randomly.



Lemma 9.3. Let $x, y \in \mathbb{R}^d$.

$$\mathbf{P}[h(x) = h(y) \mid g] = \max\{0, 1 - |\langle x - y, g \rangle|\}.$$

Proof. Once the Gaussian is fixed, so are the coordinates $\langle g, x \rangle$ and $\langle g, y \rangle$ on the line. We have $h(x) \neq h(y)$ iff a randomly shifted divider (determined by α) falls between $\langle g, x \rangle$ and $\langle g, y \rangle$. If $|\langle g, x \rangle - \langle g, y \rangle| \geq 1$, this always happens. Otherwise, it happens with probability $|\langle g, x \rangle - \langle g, y \rangle|$. $\square$

Lemma 9.4. Let $x, y \in \mathbb{R}^d$, and let $f(t)$ be the density function of the standard Gaussian $\mathcal{N}$.

$$\mathbf{P}[h(x) = h(y)] = 2 \int_0^1 f(\|x - y\|t)(1 - t) dt$$

Proof. Recall that $\langle x - y, g \rangle \sim \mathcal{N}(0, \sigma^2)$. In particular, $\langle x - y, g \rangle$ has density function $f(\sigma t)$, and $|\langle x - y, g \rangle|$ has density function $2f(\sigma t)$. We have

$$\begin{aligned} \mathbf{P}[h(x) = h(y)] &= 2 \int_0^1 \mathbf{P}[h(x) = h(y) \mid |\langle g, x - y \rangle| = t] f(\|x - y\|t) dt \\ &= 2 \int_0^1 (1 - t) f(\|x - y\|t) dt, \end{aligned}$$

as desired. $\square$

Remark 9.5. The collision probability obtained in lemma 9.4 is a function of σ , which is the ratio between $\|x - y\|$ and the target distance (here normalized to 1). This justifies our normalization. Note also that $\mathbf{P}[h(x) = h(y)]$ is decreasing in $\|x - y\|$.

Lemma 9.4 derives the exact probability of a collision of two points x and y as a function of the distance between $\|x - y\|$. Let us compare the probabilities of a “close” pair of points, with $\|x - y\| \leq 1$, and a far pair of points, with $\|x - y\| \geq \sigma$. Let

$$p = 2 \int_0^1 (1 - t) f(t) dt$$

be a lower bound on the collision probability when $\|x - y\| \leq 1$. Let

$$q = 2 \int_0^1 (1 - t) f(\sigma t) dt$$

be an upper bound on the collision probability when $\|x - y\| \geq \sigma$. We note that p is a fixed constant, about .368746....

Since $f(\sigma t)$ is decreasing in σ , $p > q$. That is, close points are more likely to collide than far points. To what extent? It turns out that the gap is not enough to simply use h as a locality sensitive hash function. But we can remedy this by amplification, as follows.

9.3.2 Amplifying the gap

Let $k \in \mathbb{N}$ be a parameter TBD. We define a hash function

$$h : \mathbb{R}^d \rightarrow \mathbb{R}^k$$

by defining each coordinate $h_i(x)$ according to the single-coordinate hash function in section 9.3.1; namely, as

$$h_i(x) = \lfloor \langle g_i, x \rangle + \alpha_i \rfloor$$

where $g_i \sim \mathcal{N}$ and $\alpha_i \in [0, 1]$ uniformly at random.

For any two points x and y , by lemma 9.4, we have

$$\mathbf{P}[h(x) = h(y)] = \left(2 \int_0^1 (1 - t) f(\|x - y\|t) dt \right)^k.$$

In particular, recalling the values of p and q as above, we have

$$\mathbf{P}[h(x) = h(y)] \geq p^k \text{ when } \|x - y\| \leq 1$$

and

$$\mathbf{P}[h(x) = h(y)] \leq q^k \text{ when } \|x - y\| \geq \sigma.$$

Now, increasing k decreases both p^k and q^k , which is both good and bad. As p^k decreases, so do the odds of finding a near neighbor when we hash. We will need to rebuild the data structure $\ell = 1/p^k$ times to be able to find a good neighbor with

constant probability, which is expensive.³ On the other hand, as q^k decreases, the number of hash collisions with bad points also reduces. Ultimately, the algorithm pays a running time proportional to the total number of bad collisions as it scans the lists in the hash bucket as it searches for a good quantity. Overall, the ratio $(q/p)^k$ decreases, so to at least some extent k is useful.

Ultimately, the quantity we want to minimize is

$$k \left(\frac{1}{p} \right)^k + \left(\frac{q}{p} \right)^k n.$$

The $(1/p)^k$ term represents having to compute $\ell = (1/p)^k$ hash codes. The $\left(\frac{q}{p}\right)^k n$ represents the expected number of hash collisions across all ℓ instances with bad elements. The above quantity models the running time up to an additional factor of d , incurred from either hashing the query point or comparing the distance between the query point and a point in the same hash bucket. We can rewrite the above as

$$\left(\frac{1}{p} \right)^k (k + q^k n).$$

Choose $k = \log(1/n)/\log(q) = \log(n)/\log(1/q)$. Then $q^k n = 1$, and

$$k \left(\frac{1}{p} \right)^k = \frac{\log(n)}{\log(1/q)} n^{\log(1/p)/\log(1/q)}.$$

Consider the exponent $\log(1/p)/\log(1/q)$: since $p > q$, this quantity is less than 1. In fact, one can show that it is about $1/1 + \epsilon$ for $\sigma \approx 1 + \epsilon$.

Theorem 9.6. *One can compute a ρ -approximate nearest neighbor w/r/t Euclidean distance with high probability in $\tilde{O}(n^{\rho(\sigma)})$ randomized time, where*

$$\rho(\sigma) = \frac{\log(1/p)}{\log(1/q)}, \quad p = 2 \int_0^1 (1-t)f(t) dt \approx .368746, \quad q = 2 \int_0^1 (1-t)f(\sigma t) dt,$$

and $f(t) = e^{-t^2/2}/\sqrt{2\pi}$ is the density function of the standard Gaussian.

Below, we show that for $\sigma^2 \leq 1.6$, we have

$$\rho(\sigma) \leq 1/\sigma^2.$$

³Indeed, the probability of failing to find the good neighbor in each of ℓ constructions, each of which has k hash coordinates, is

$$(1 - p^k)^\ell \approx e^{-p^k \ell}.$$

9.3.3 Analyzing $\rho(\sigma)$

For any fixed value of $\sigma > 1$, one can run a numerical computation to get an accurate estimate of $\rho(\sigma)$. See for example [DIIM04]. Below, we analyze $\rho(\sigma)$ and obtain some upper bounds on ρ . While these bounds are of course not as tight as a computer simulation, they reveal some intuition for how the value of $\rho(\sigma)$ varies with σ .

Lemma 9.7. *Suppose $\sigma^2 = 1 + \epsilon$ for $\epsilon \in (0, 1)$. Then*

$$q \geq (1 - \epsilon)^2 p.$$

In particular,

$$\rho(\sigma) \geq \frac{\log(1/p)}{\log(1/p) - 2\log(1 - \epsilon)}.$$

Proof. We have

$$\sqrt{2\pi}(p - q) = \int_0^1 (1 - t) (e^{-t^2/2} - e^{-\sigma^2 t^2/2})$$

Say $\sigma^2 = (1 + \epsilon)$ with ϵ sufficiently small. Then for all $t \in [0, 1]$, we have

$$e^{-t^2/2} - e^{-(1+\epsilon)t^2/2} = e^{-t^2/2} (1 - e^{-\epsilon t^2/2}) = e^{-t^2/2} \left(\epsilon t^2/2 - (\epsilon t^2/2)^2/2 \right) \leq \epsilon(1 - \epsilon)e^{-t^2/2}.$$

Thus

$$\begin{aligned} \int_0^1 (1 - t) (e^{-t^2/2} - e^{-t^2/2\sigma^2}) dt &\leq \epsilon(1 - \epsilon) \int_0^1 (1 - t) e^{-t^2/2} dt \\ &= \epsilon(1 - \epsilon) \sqrt{2\pi} p. \end{aligned}$$

Thus

$$\sqrt{2\pi}(p - q) \leq \epsilon(1 - \epsilon) \sqrt{2\pi} p;$$

in turn;

$$(1 - \epsilon)^2 p \leq q.$$

□

Lemma 9.8. *Let $\sigma^2 \leq 1.6$. Then*

$$\rho(\sigma) \leq \frac{1}{\sigma^2}.$$

Proof. We continue from the conclusion of lemma 9.7 with $\epsilon = \sigma^2 - 1$. One can directly compute that $\log(1/p)$ is a constant, about (and slightly greater than) .77. On the other hand, $\log(1 - \epsilon) \approx -\epsilon$ for small ϵ ; in particular, $-2 \log(1 - \epsilon) \geq -\log(1/p)\epsilon$ for $\epsilon \leq .6$. Thus

$$\rho(\sigma) \leq \frac{\log(1/p)}{\log(1/p) - 2 \log(1 - \epsilon)} \leq \frac{\log(1/p)}{(1 + \epsilon) \log(1/p)} = \frac{1}{1 + \epsilon} = \frac{1}{\sigma^2}.$$

□

9.4 Exercises

Exercise 9.1. In this exercise, we will develop a 2-approximation LSH scheme for bit strings $s \in \{0, 1\}^d$ of a fixed length d w/r/t *Hamming distance*. The Hamming distance between two strings $s, t \in \{0, 1\}^d$ is this fraction of coordinates in which they differ:

$$(\text{Hamming})(s, t) = \frac{|\{i \in [d] : s_i \neq t_i\}|}{d}.$$

Of course one can treat bit strings as vectors in $\mathbb{R}^d$ where the Hamming distance coincides with the Euclidean distance squared. Here we explore an alternative approach.

1. Consider the randomly constructed hash function $h : \{0, 1\}^d \rightarrow \{0, 1\}$ defined by

$$h(x) = x_i,$$

where $i \in [d]$ is sampled uniformly at random. For two points $s, t \in \{0, 1\}^d$, what is $\mathbf{P}[h(s) = h(t)]$, as a function of the Hamming distance between s and t ?

2. Fix a target distance $r \in [0, 1]$. Construct a data structure that over a set P of n strings $\{0, 1\}^d$ to answer the following query with high probability.

Given a query point $s \in \{0, 1\}^d$, either return a point $x \in P$ with Hamming distance $\leq 2r$ from s , or declare that there are no points within Hamming distance r from s .

In addition to describing the algorithm, one should analyze the preprocessing time and space, the query time, and the probability of correctness.

3. Briefly describe how to use the above data structure to efficiently find 2-approximate nearest neighbors w/r/t Hamming distance.