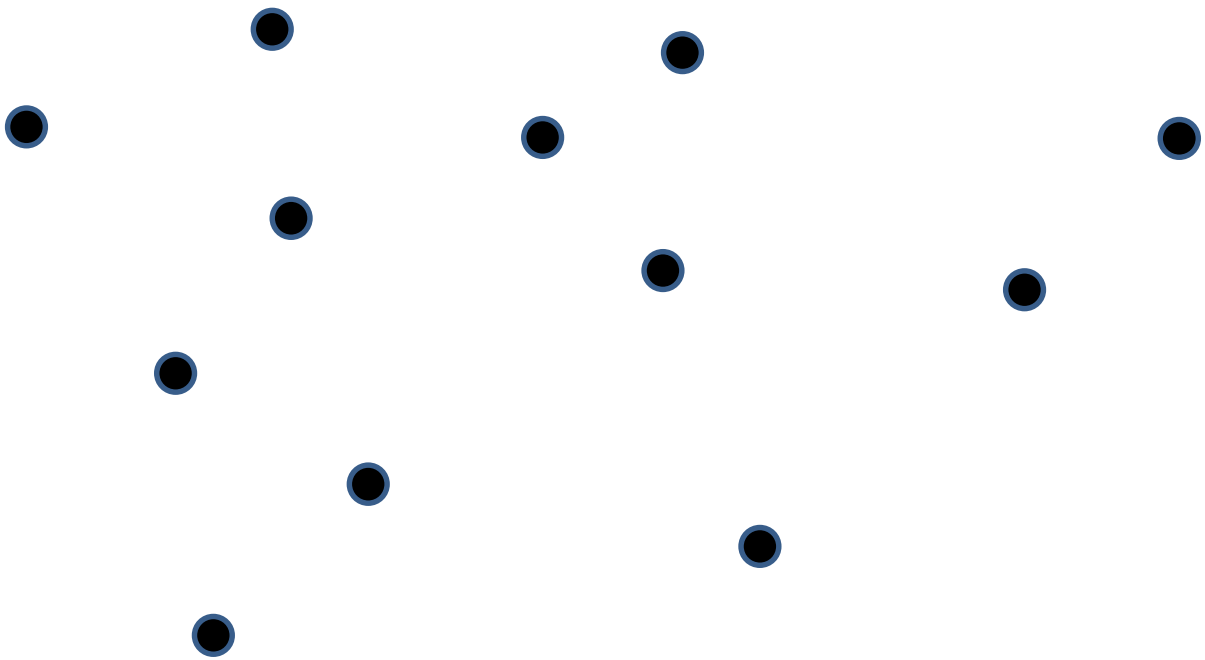
Locality sensitive hashing (LSH)

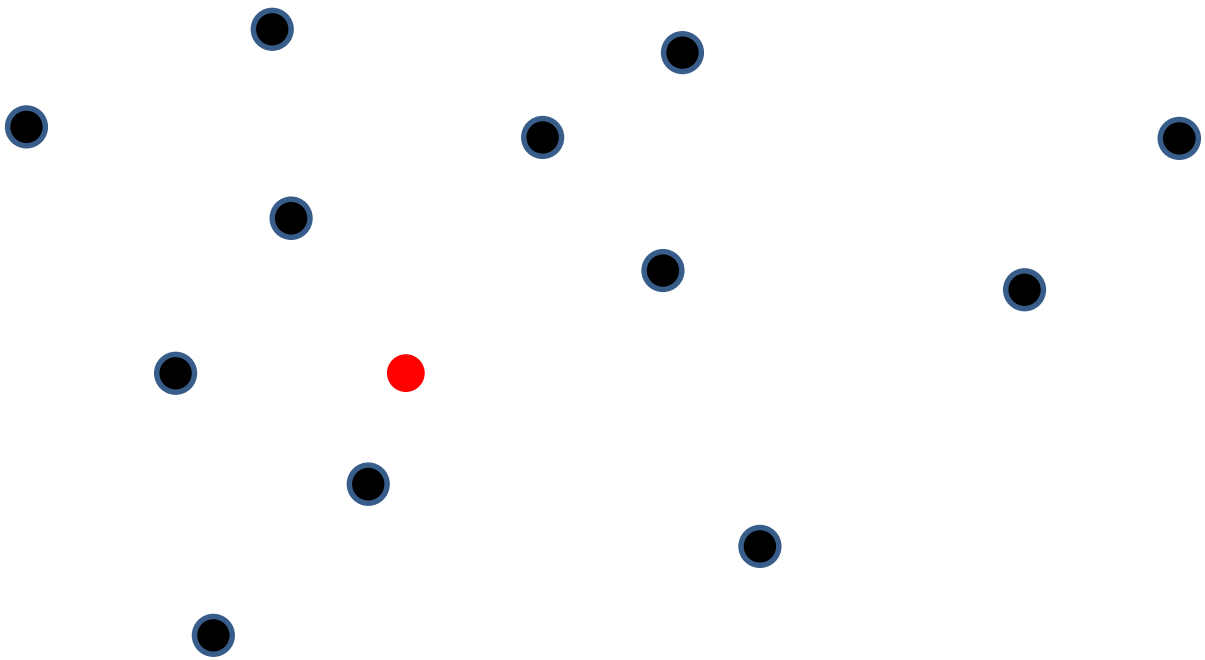
Nearest Neighbor

Given a set P of n points in R^d



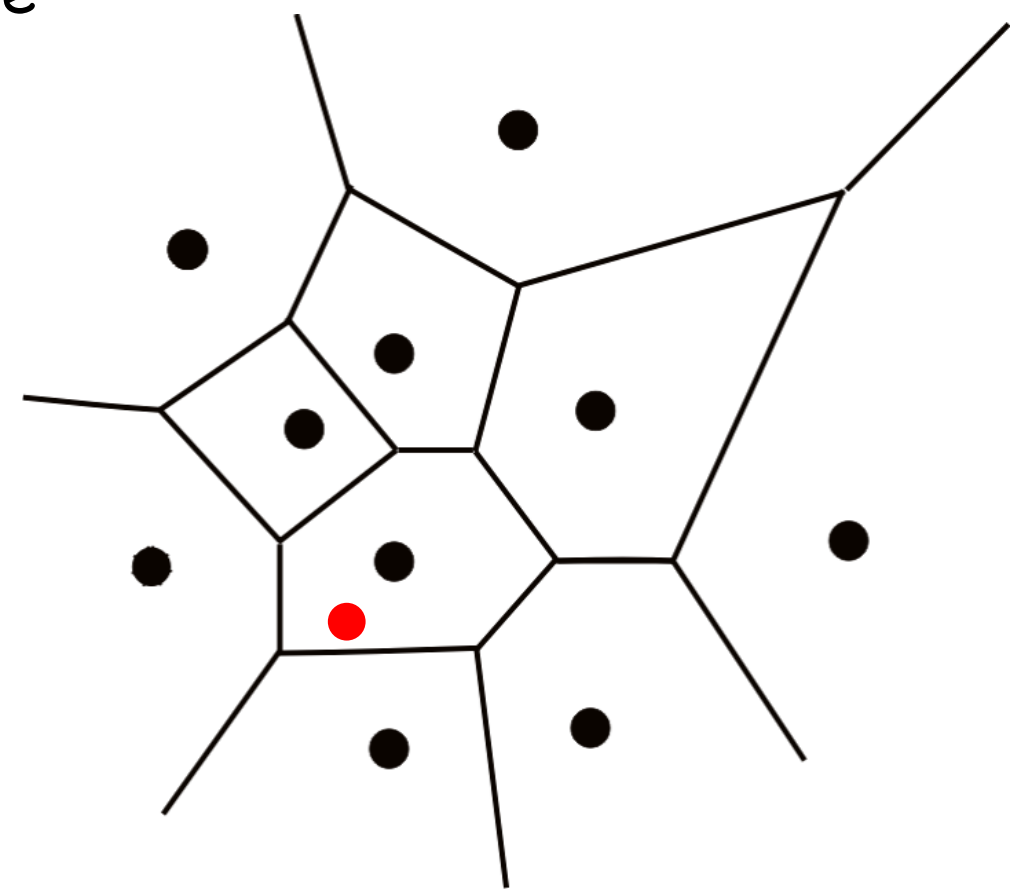
Nearest Neighbor

Want to build a data structure to answer nearest neighbor queries



Voronoi Diagram

Build a Voronoi diagram & a point location data structure



Curse of dimensionality

- In R^2 the Voronoi diagram is of size $O(n)$
- Query takes $O(\log n)$ time
- In R^d the complexity is $O(n^{\lceil d/2 \rceil})$
- Other techniques also scale bad with the dimension

Problem has many variants

- Approximate (will talk about this soon)
- k nearest neighbors
- All neighbors at distance $\leq r$

All nearest neighbors

- Finding the nearest neighbors of every point in the data (near duplicate web pages)

Locality Sensitive Hashing

- We will use a family of hash functions such that close points tend to hash to the same bucket.

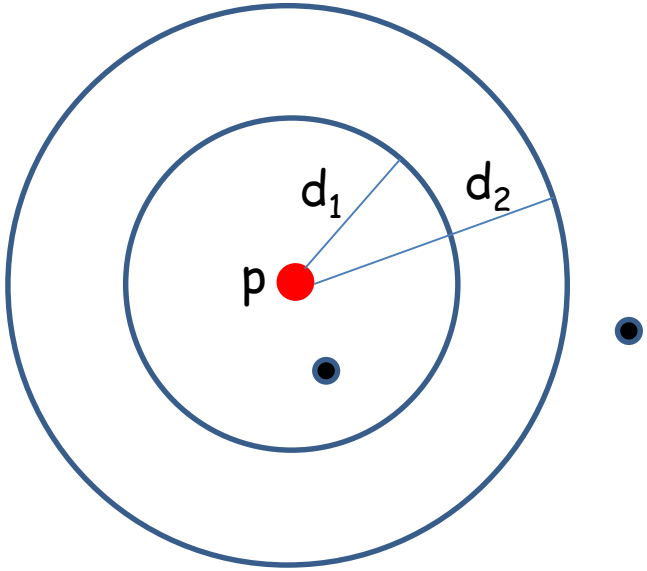
- Put all points of P in their buckets, ideally we want the query q to find its nearest neighbor in its bucket

Locality Sensitive Hashing

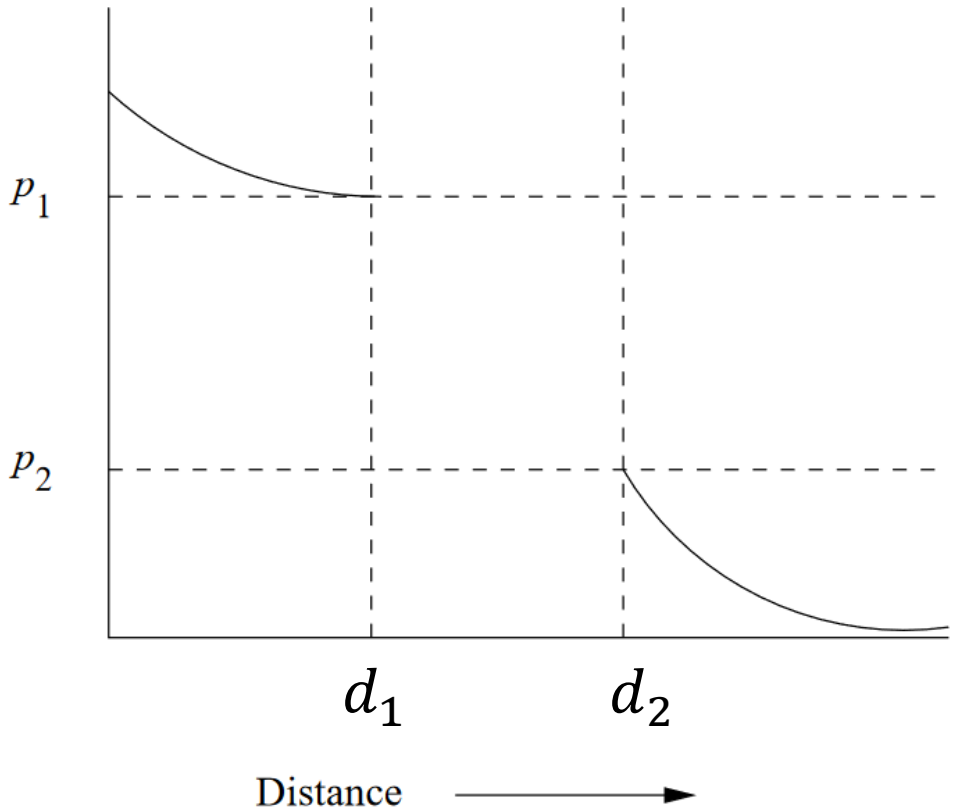
A family H of functions is $(d_1 < d_2, p_1 > p_2)$-sensitive if

$d(p,q) \leq d_1 \rightarrow Pr[h(p) = h(q)] \geq p_1$

$d(p,q) \geq d_2 \rightarrow Pr[h(p) = h(q)] \leq p_2$



Locality Sensitive Family for a distance function $d(x, y)$



LSF for hamming distance

Think of the points as strings of m bits and consider the hamming distance $h(p,q)$

H={h_i(p) = the i-th bit of p} is locality sensitive wrt $h(p,q)$

Pr[h(p) = h(q)] = 1 – ham(p,q)/m

So this family is $\left(d_1, d_2, 1-\frac{d_1}{m}, 1-\frac{d_2}{m}\right)$-sensitive

Jaacard distance and random permutations

Think of p and q as sets

jaccard(p,q) = $|p \cap q| / |p \cup q|$
Jd(p,q) = 1-jaccard(p,q) = 1 - $|p \cap q| / |p \cup q|$

H={h_π(p) = min in π of the items in p}

Pr[h_π(p) = h_π(q)] = jaccard(p,q)

Efficiency: Need to pick π from a min-wise ind. family of permutations

This is $(d_1 = 1 - j_1, d_2 = 1 - j_2, j_1, j_2)$-sensitive family

Jaacard distance and minhash

Think of p and q as sets

Jd(p,q) = 1-jaccard(p,q) = 1 - $|p \cap q| / |p \cup q|$

$H = \{h_r(p) = \min r(x),\ x \in p,\ r \sim U[0,1]\}$

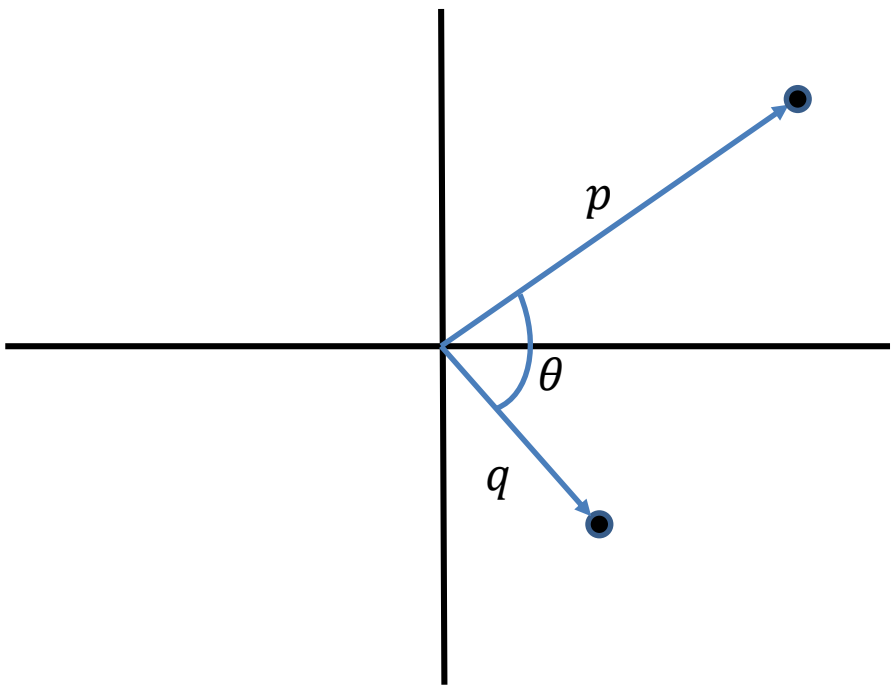
$\Pr[h_r(p) = h_r(q)] = \text{jaccard}(p,q)$

Precision for r should avoid ties

This is $(d_1 = 1 - j_1, d_2 = 1 - j_2, j_1, j_2)$-sensitive family

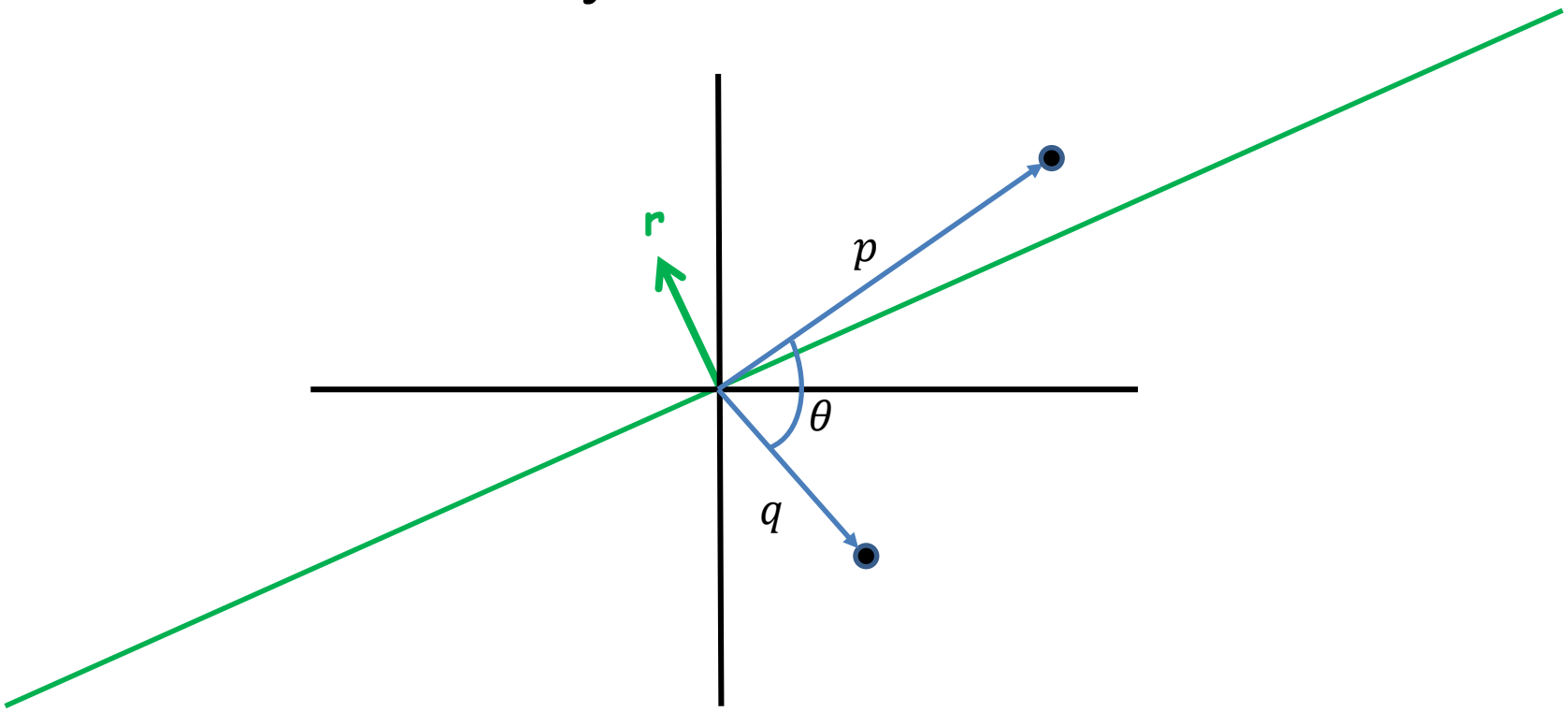
Cosine distance

p and q are vectors and $d(p, q) = \theta$



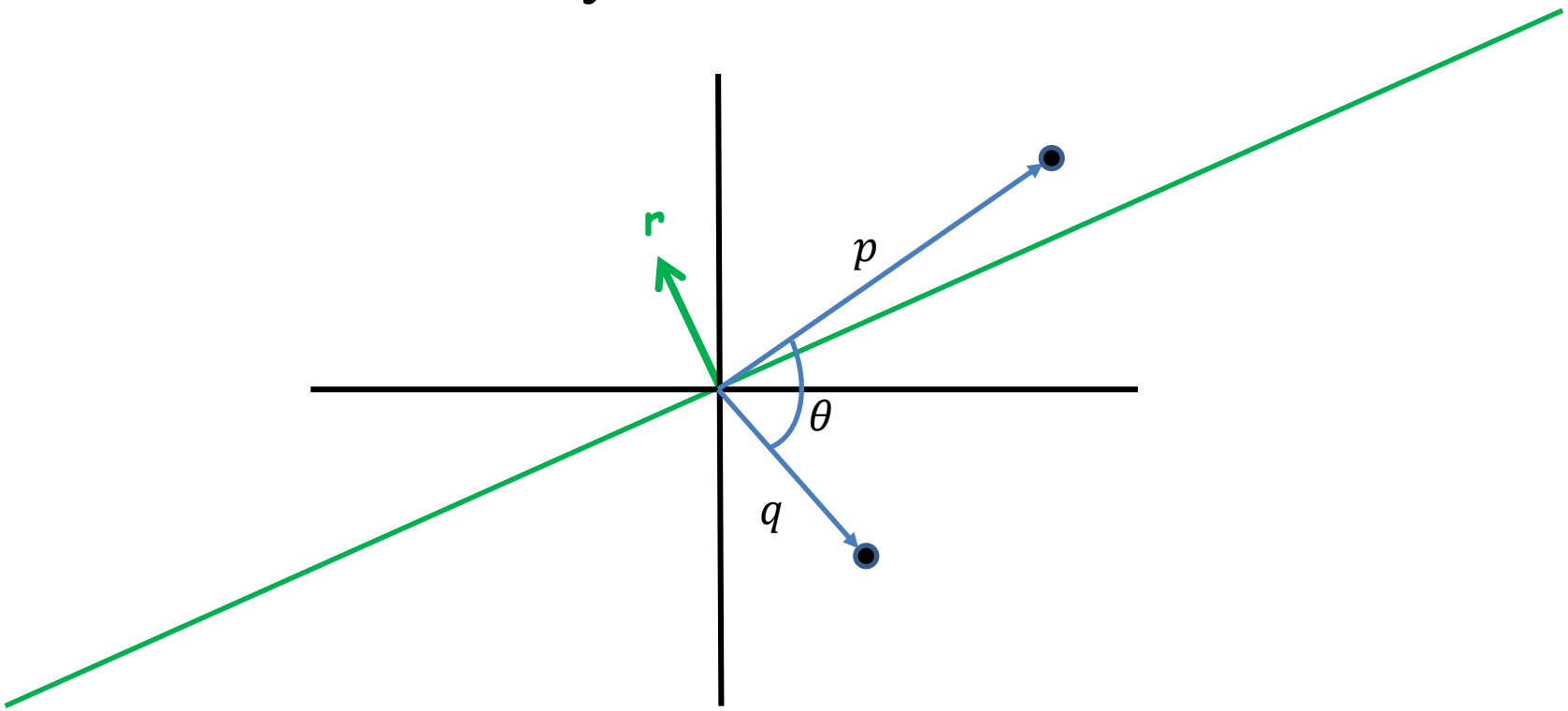
Cosine distance

H = {$h_r(p)$ = 1 if $r \cdot p > 0$, 0 otherwise | r is a random unit vector}



Cosine distance

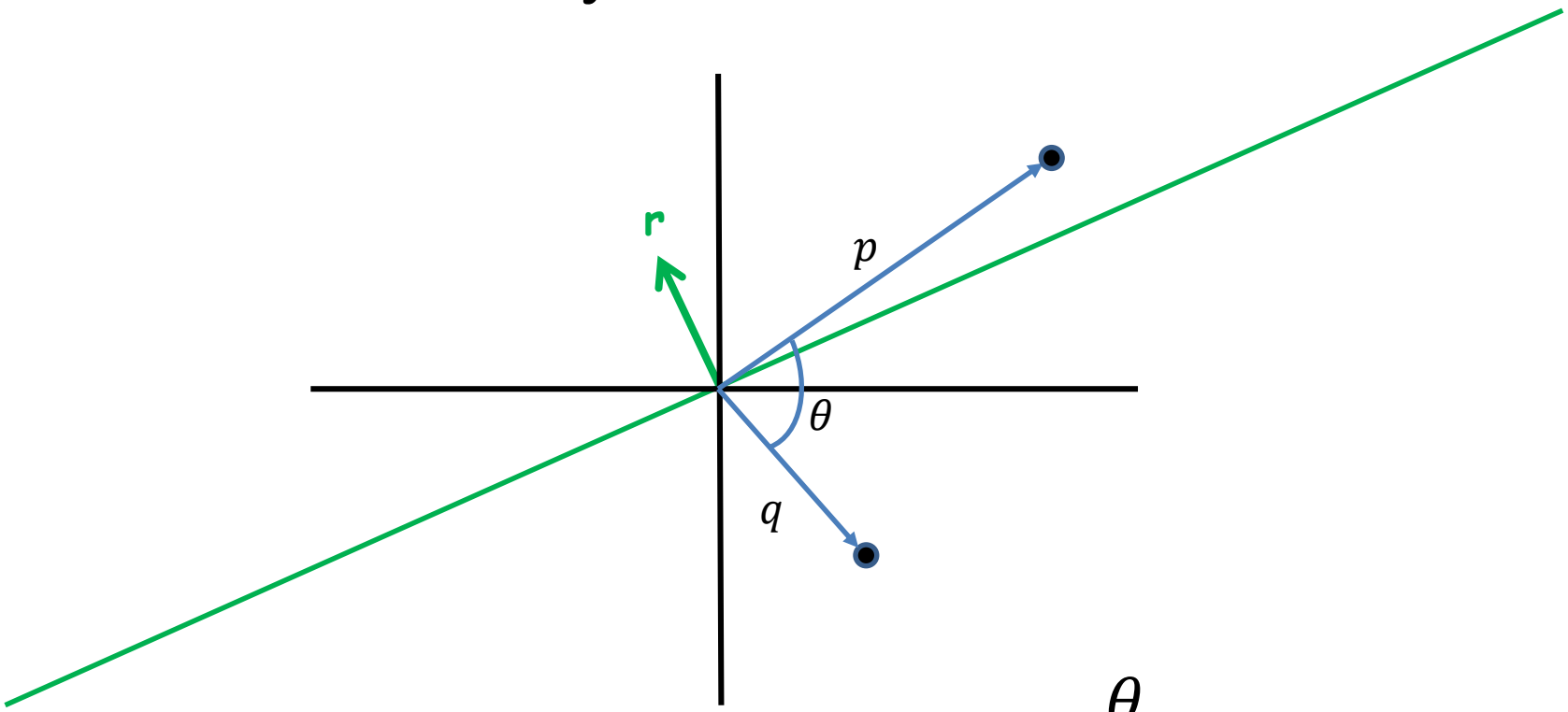
H = {$h_r(p)$ = 1 if $r \cdot p > 0$, 0 otherwise | r is a random unit vector}



$$\Pr[h_r(p) = h_r(q)] = ?$$

Cosine distance

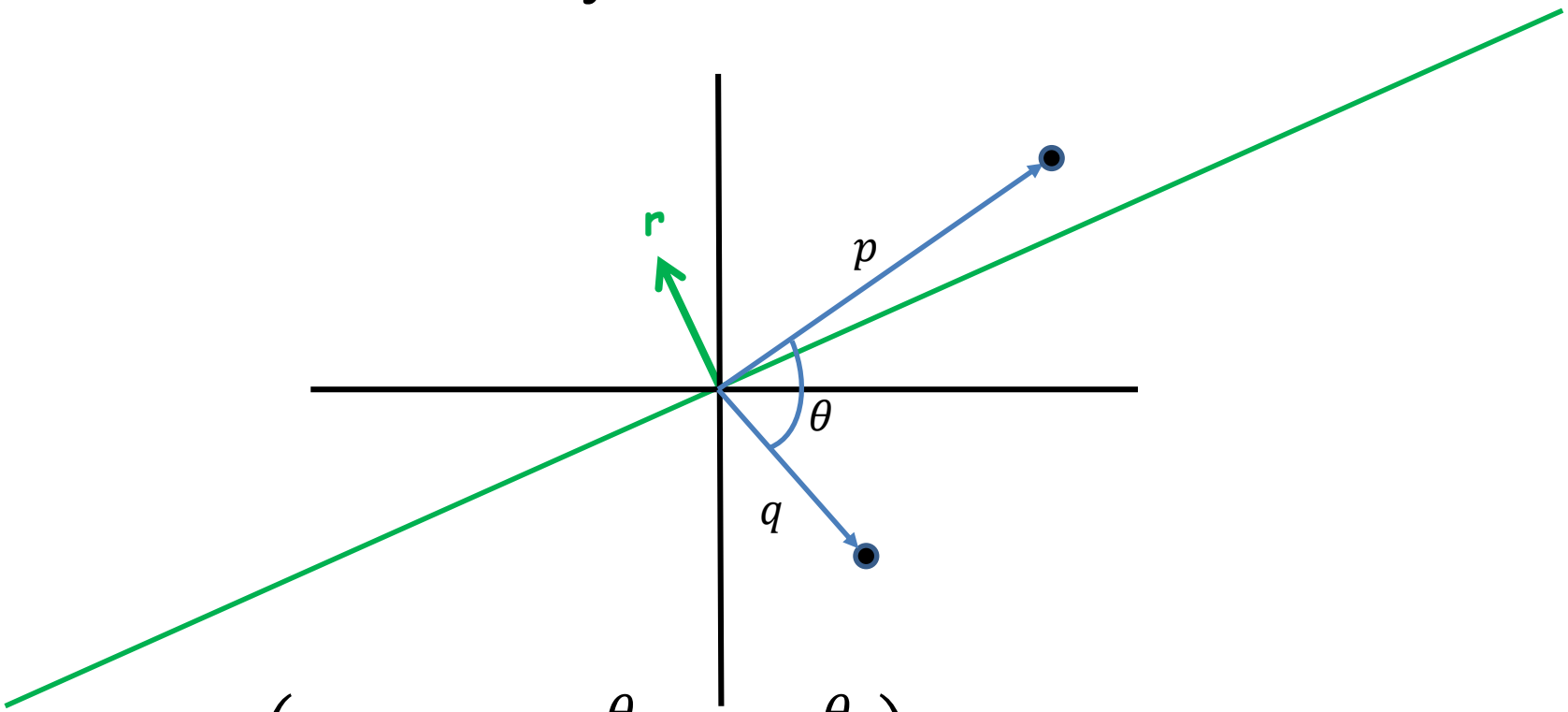
H = {h_r(p) = 1 if r·p > 0, 0 otherwise | r is a random unit vector}



$$\Pr[h_r(p) = h_r(q)] = 1 - \frac{\theta}{\pi}$$

Cosine distance

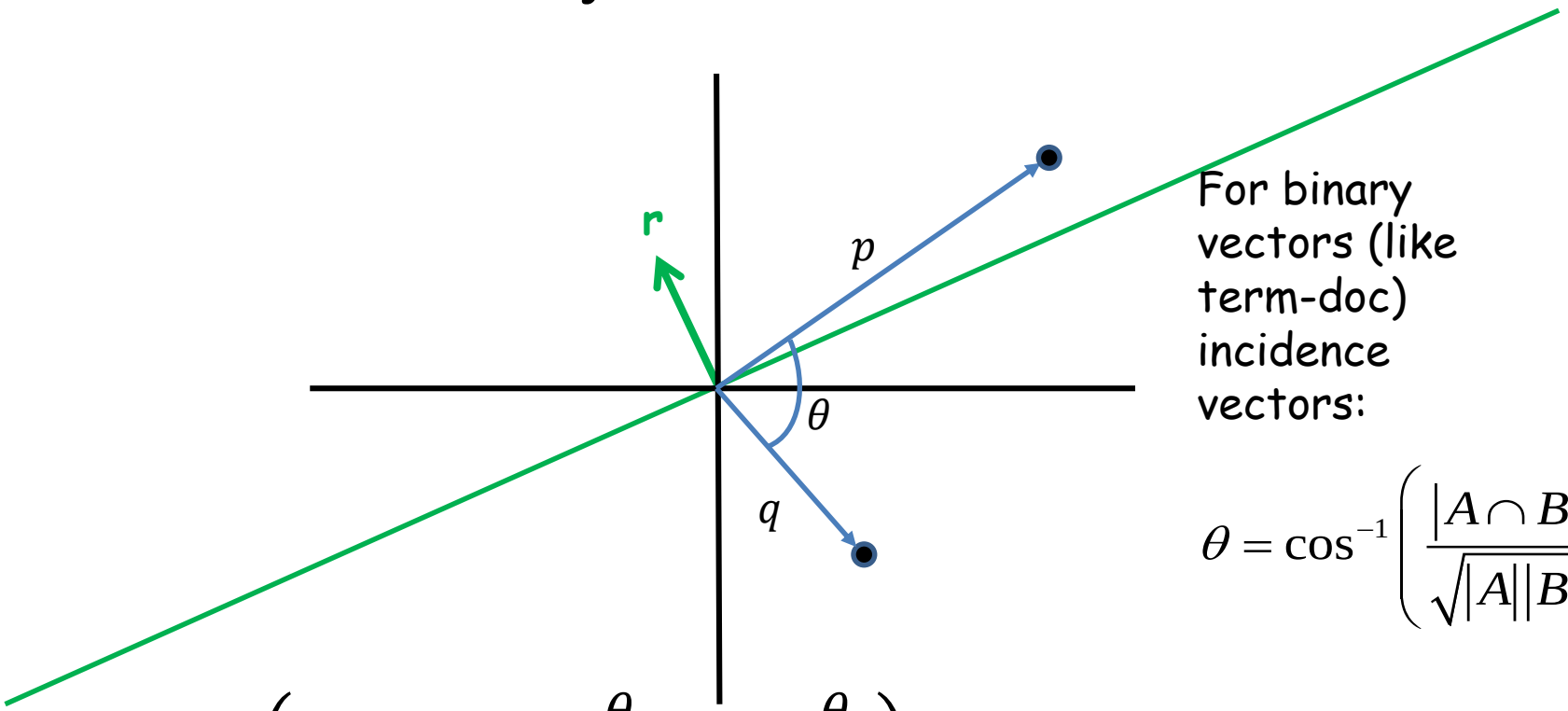
H = {$h_r(p)$ = 1 if $r \cdot p > 0$, 0 otherwise | r is a random unit vector}



This is $\left(\theta_1, \theta_2, 1 - \frac{\theta_1}{\pi}, 1 - \frac{\theta_2}{\pi}\right)$-sensitive family

Cosine distance

H = {$h_r(p)$ = 1 if $r \cdot p > 0$, 0 otherwise | r is a random unit vector}



For binary vectors (like term-doc) incidence vectors:

$$\theta = \cos^{-1}\left(\frac{|A \cap B|}{\sqrt{|A||B|}}\right)$$

This is $\left(\theta_1, \theta_2, 1 - \frac{\theta_1}{\pi}, 1 - \frac{\theta_2}{\pi}\right)$-sensitive family

Combining by "AND"

Reduce the number of false positives by concatenating hash function to get a new family of hash functions

$h(p) = h_1(p)h_2(p)h_3(p)h_4(p)\ldots\ldots h_k(p) = 00101010$

We get a new family of hash functions

$h(p) = h(q)$ iff $h_i(p) = h_i(q)$ $\forall \boldsymbol{i}$

If the original family is (d_1, d_2, p_1, p_2)-sensitive then the new family is $\left(d_1, d_2, (p_1)^k, (p_2)^k\right)$-sensitive

Combining by "OR"

Reduce the number of false negatives

$h(p) = h_1(p), h_2(p), h_3(p), h_4(p), \ldots\ldots, h_L(p) =$
0,0,1,0,1,0,1,0

We get a new family of hash functions

$h(p) = h(q)$ iff $\exists \boldsymbol{i}\ s.t.\ h_i(p) = h_i(q)$

If the original family is (d_1, d_2, p_1, p_2)-sensitive then the new family is
$\left(d_1, d_2, 1-(1-p_1)^L, 1-(1-p_2)^L\right)$-sensitive

"And k" followed by "Or L"

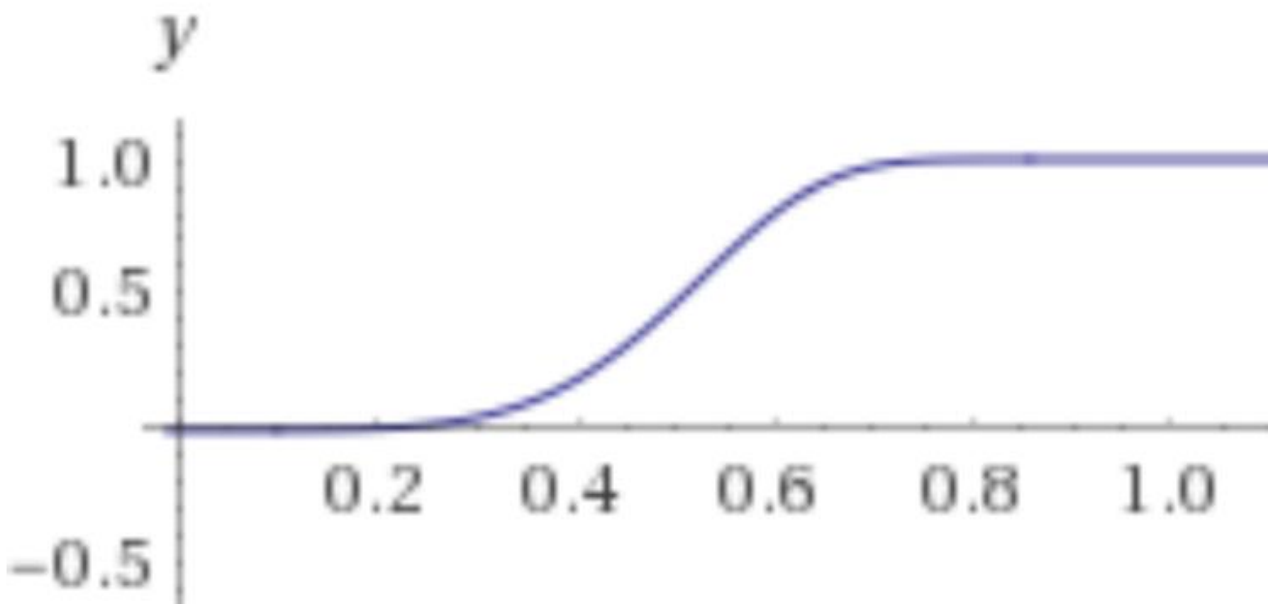
(d_1, d_2, p_1, p_2)-sensitive ➔

$\left(d_1, d_2, 1 - \left(1 - (p_1)^k\right)^L, 1 - \left(1 - (p_2)^k\right)^L\right)$-sensitive

What does this do ?

The function $1-\left(1-p^k\right)^L$

- k=5, L=20



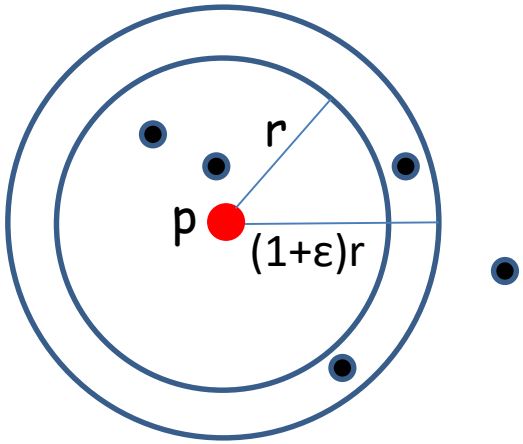
p	$1-\left(1-p^5\right)^{20}$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

For example if (p_1, p_2) were $(0.6,0.4)$ then now they are $(0.802,0.186)$

A theoretical result on NN

(r,ε)-neighbor problem

If there is a neighbor p, such that $d(p,q) \leq r$, return p′, s.t. $d(p',q) \leq (1+\varepsilon)r$.



(r,ε)-neighbor problem

- Lets construct a data structure that succeeds with constant probability

- Focus on the hamming distance first

NN using locality sensitive hashing

- Take a $(r_1, r_2, p_1, p_2) =$
  $(r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)$ - sensitive family

- If there is a neighbor at distance r we catch it with probability p_1

NN using locality sensitive hashing

- Take a $(r_1, r_2, p_1, p_2) =$
  $(r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)$ - sensitive family

- If there is a neighbor at distance r we catch it with probability p_1 so to guarantee catching it we need to "or" $1/p_1$ functions..

NN using locality sensitive hashing

- Take a $(r_1, r_2, p_1, p_2) =$
  $(r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)$ - sensitive family

- If there is a neighbor at distance r we catch it with probability p_1 so to guarantee catching it we need to "or" $1/p_1$ functions..

- But we also get false positives, how many ?

NN using locality sensitive hashing

- Take a $(r_1, r_2, p_1, p_2) =$
  $(r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)$ - sensitive family

- If there is a neighbor at distance r we catch it with probability p_1 so to guarantee catching it we need to "or" $1/p_1$ functions..

- But we also get false positives, how many ?
  $n\frac{p_2}{p_1} \approx n(1-(1-p_2)^{\frac{1}{p_1}})$

NN using locality sensitive hashing

- Take a $(r_1, r_2, p_1, p_2) = (r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)$ - sensitive family
- Make a new function by concatenating ("and") k of these basic functions
- We get a $(r_1, r_2, (p_1)^k, (p_2)^k)$ sensitive family
- If there is a neighbor at distance r we catch it with probability $(p_1)^k$ so to guarantee catching it we need $L=1/(p_1)^k$ functions. We get a $(r_1, r_2, 1-(1-(p_1)^k)^L, 1-(1-(p_2)^k)^L)$ sensitive family
- But we also get false positives in our $1/(p_1)^k$ buckets, how many ? $n(p_2)^k/(p_1)^k$

(r,ε)-Neighbor with constant prob

Scan the first $4n(p_2)^k/(p_1)^k$ points in the buckets and return the closest

A close neighbor ($\leq r_1$) is in one of the buckets with probability $\geq 1-(1/e)$

There are $\leq 4n(p_2)^k/(p_1)^k$ false positives with probability $\geq 3/4$

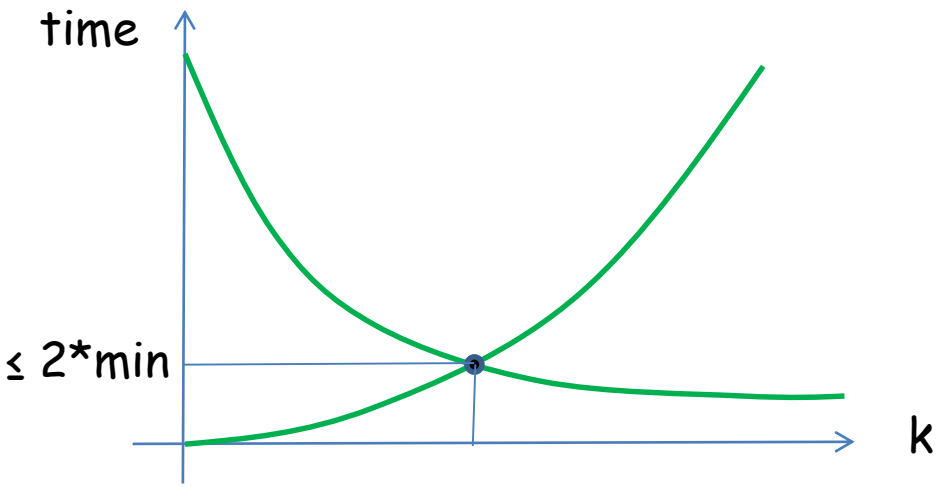
➔ Both events happen with constant prob.

Analysis

Total query time:
(each op takes time prop. to the dim.)

$$\frac{k}{(p_1)^k} + n\left(\frac{p_2}{p_1}\right)^k$$

We want to choose k to minimize this.



Analysis

Total query time:
(each op takes time prop. to the dim.)

$$\frac{k}{(p_1)^k} + n\left(\frac{p_2}{p_1}\right)^k$$

We want to choose k to minimize this:

$$k = n(p_2)^k \quad \leftrightarrow \quad \frac{n}{k} = \left(\frac{1}{p_2}\right)^k$$

$$k = \log_{1/p_2}(n) - \Theta(\log\log n)$$

Summary

Total query time: $$\frac{k}{(p_1)^k} + n\left(\frac{p_2}{p_1}\right)^k$$

Put: $$k = \lceil \log_{1/p_2}(n) \rceil$$

$$t \le \left(\frac{1}{p_1}\right)^{\lceil \log_{1/p_2}(n) \rceil} \le \frac{1}{p_1}\left(\frac{1}{p_1}\right)^{\log_{1/p_2}(n)} = \frac{1}{p_1} n^{\frac{\log(1/p_1)}{\log(1/p_2)}} = \frac{1}{p_1} n^{\rho}$$

space: $$n\frac{1}{p_1} n^{\rho} = \frac{1}{p_1} n^{1+\rho}$$

What is ρ ?

$$\rho = \frac{\log\left(\frac{1}{p_1}\right)}{\log\left(\frac{1}{p_2}\right)} = \frac{\log(p_1)}{\log(p_2)} = \frac{\log\left(1-\frac{r}{m}\right)}{\log\left(1-\frac{(1+\varepsilon)r}{m}\right)} \approx \frac{1}{1+\varepsilon}$$

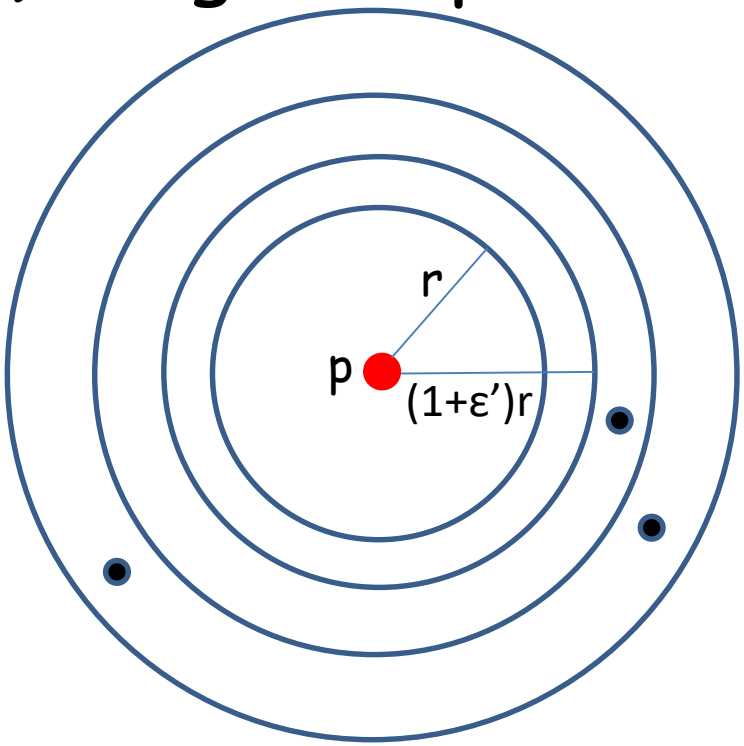
$(1+\varepsilon)$-approximate NN

- Given q find p such that $\forall p' \neq p$
  $d(q,p) \leq (1+\varepsilon)d(q,p')$

- We can use our solution to the (r,ε)-neighbor problem

(1+ε)-approximate NN vs (r,ε)-neighbor problem

- If we know r_{min} and r_{max} we can find (1+ε)-approximate NN using $\log(r_{max}/r_{min})$ $(r,\varepsilon' \approx \varepsilon/2)$-neighbor problems



LSH using p-stable distributions

Definition: A distribution D is 2-stable if when $X_1,\ldots\ldots,X_d$ are drawn from D,
$\sum v_i X_i = \|v\| X$ where X is drawn from D.

So what do we do with this ?

$h(p) = \sum p_i X_i$

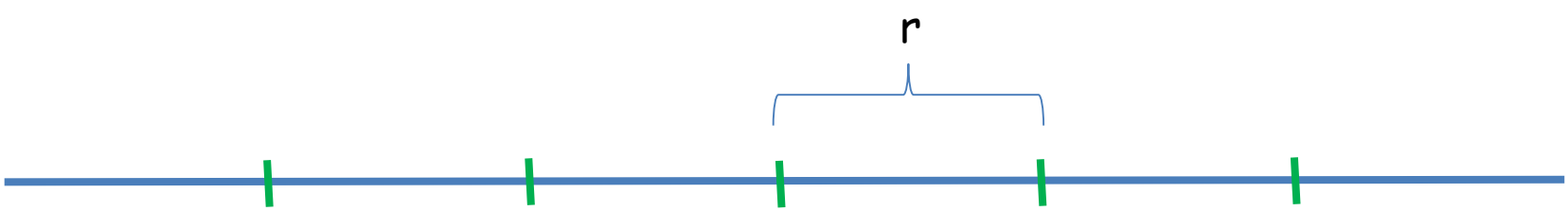
$h(p)-h(q) = \sum p_i X_i - \sum q_i X_i = \sum (p_i - q_i) X_i = ||p-q||X$

LSH using p-stable distributions

So what do we do with this ?

$h(p) = \lfloor (pX)/r \rfloor$

Pick r to maximize ρ…



Bibliography

- M. Charikar: Similarity estimation techniques from rounding algorithms. STOC 2002: 380-388
- P. Indyk, R. Motwani: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. STOC 1998: 604-613.
- A. Gionis, P. Indyk, R. Motwani: Similarity Search in High Dimensions via Hashing. VLDB 1999: 518-529
- M. R. Henzinger: Finding near-duplicate web pages: a large-scale evaluation of algorithms. SIGIR 2006: 284-291
- G. S. Manku, A. Jain , A. Das Sarma: Detecting near-duplicates for web crawling. WWW 2007: 141-150