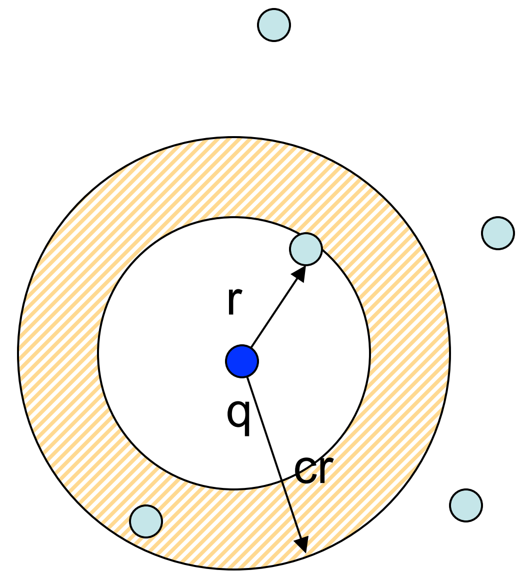
Similarity Search in High Dimensions III

Piotr Indyk

MIT

Approximate Near Neighbor

- c-Approximate r-Near Neighbor: build data structure which, for any query q:
  - If there is a point $p \in P$, $||p-q|| \leq r$
  - it returns $p' \in P$, $||p-q|| \leq cr$



LSH

- A family H of functions $h: R^d \rightarrow U$ is called (P_1,P_2,r,cr)-sensitive, if for any p,q:
  - if $||p-q|| < r$ then $Pr[\ h(p)=h(q)\ ] > P_1$
  - if $||p-q|| > cr$ then $Pr[\ h(p)=h(q)\ ] < P_2$

- Example: Hamming distance
  - $h(p)=p_i$, i.e., the i-th bit of p
    - Probabilities: $Pr[\ h(p)=h(q)\ ] = 1-H(p,q)/d$

p=10010010
q=11010110

Algorithm

- We use functions of the form

  $$g(p)=\langle h_1(p),h_2(p),\ldots,h_k(p)\rangle$$

- Preprocessing:
  - Select $g_1\ldots g_L$
  - For all $p\in P$, hash p to buckets $g_1(p)\ldots g_L(p)$

- Query:
  - Retrieve the points from buckets $g_1(q)$, $g_2(q)$, … , until
    - Either the points from all L buckets have been retrieved, or
    - Total number of points retrieved exceeds $3L$
  - Answer the query based on the retrieved points
  - Total time: $O(dL)$

Analysis [IM'98, Gionis-Indyk-Motwani'99]

- Lemma1: the algorithm solves c-approximate NN with:
  - Number of hash functions:

    $$L=C\, n^{\rho},\ \rho=\log(1/P1)/\log(1/P2)$$

    ($C=C(P1,P2)$ is a constant for P1 bounded away from 0) [O'Donnell-Wu-Zhou'09]
  - Constant success probability per query q
- Lemma 2: for Hamming LSH functions, we have $\rho=1/c$

Proof

- Define:
  - p: a point such that $||p-q|| \leq r$
  - $FAR(q)=\{ p' \in P: ||p'-q|| > c\, r \}$
  - $B_i(q)=\{ p' \in P: g_i(p')=g_i(q) \}$
- Will show that both events occur with >0 probability:
  - E_1: $g_i(p)=g_i(q)$ for some $i=1\ldots L$
  - E_2: $\Sigma_i |B_i(q) \cap FAR(q)| < 3L$

Proof ctd.

- Set $k = \text{ceil}(\log_{1/P2} n)$
- For $p' \in FAR(q)$,

$$\Pr[g_i(p')=g_i(q)] \le P_2^k \le 1/n$$

- $E[\ |B_i(q) \cap FAR(q)|\ ] \le 1$
- $E[\Sigma_i\ |B_i(q) \cap FAR(q)|\ ] \le L$
- $\Pr[\Sigma_i\ |B_i(q) \cap FAR(q)| \ge 3L\ ] \le 1/3$

Proof, ctd.

- $\Pr[\, g_i(p)=g_i(q) \,] \geq 1/P_1{}^k \geq P_1{}^{\log_{1/P2}(n)+1}$
  $\geq 1/(P1\, n^{\rho})=1/L$
- $\Pr[\, g_i(p) \neq g_i(q),\ i=1..L] \leq (1-1/L)^L \leq 1/e$

Proof, end

- $\Pr[E_1 \text{ not true}]+\Pr[E_2 \text{ not true}]$
  $\leq 1/3+1/e = 0.7012.$
- $\Pr[E_1 \cap E_2] \geq 1-(1/3+1/e) \approx 0.3$

Proof of Lemma 2

- Statement: for
  - $P1=1-r/d$
  - $P2=1-cr/d$

  we have $\rho=\log(P1)/\log(P2) \leq 1/c$
- Proof:
  - Need $P1^c \geq P2$
  - But $(1-x)^c \geq (1-cx)$ for any $1>x>0$, $c>1$

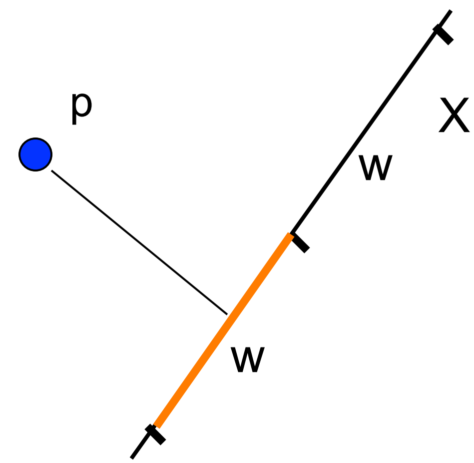
Recap

- LSH solves c-approximate NN with:
  - Number of hash fun: $L=O(n^{\rho})$, $\rho=\log(1/P1)/\log(1/P2)$
  - For Hamming distance we have $\rho=1/c$
- Questions:
  - Beyond Hamming distance ?
    - Embed l_2 into l_1 (random projections)
    - l_1 into Hamming (discretization)
  - Reduce the exponent ρ ?

Projection-based LSH for L2

[Datar-Immorlica-Indyk-Mirrokni'04]

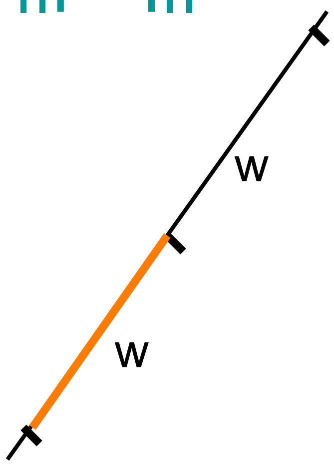
- Define $h_{X,b}(p)=\lfloor(p*X+b)/w\rfloor$:
  - $w \approx r$
  - $X=(X_1 \ldots X_d)$, where X_i is chosen from:
    - Gaussian distribution (for l_2 norm)*
  - b is a scalar



Analysis

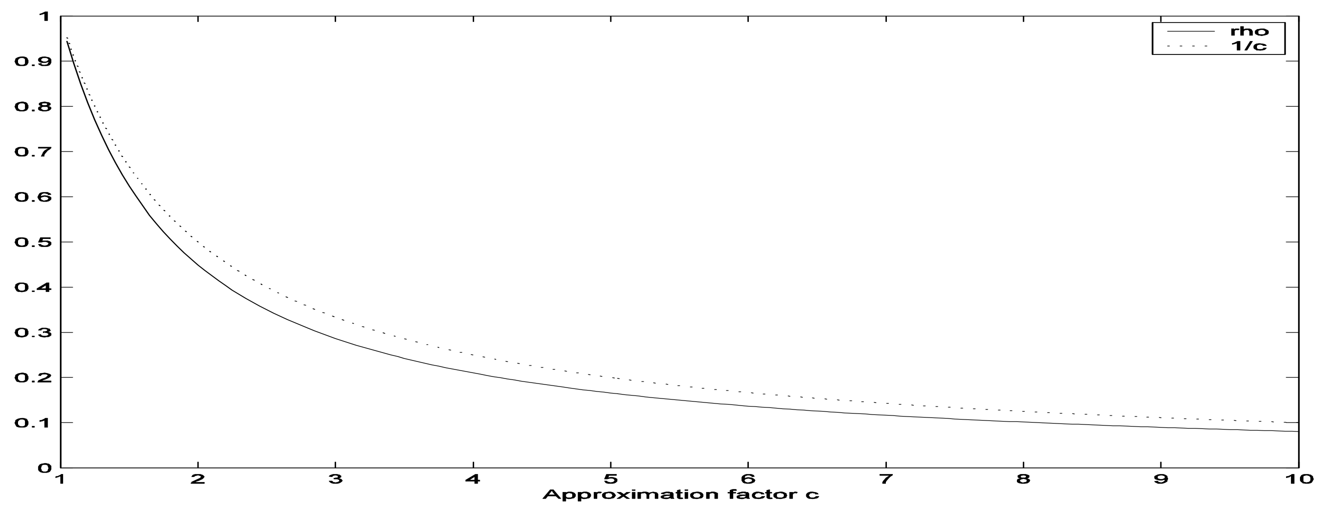
- Need to:
  - Compute $Pr[h(p)=h(q)]$ as a function of $||p-q||$ and w; this defines P_1 and P_2
  - For each c choose w that minimizes

$$\rho=\log_{1/P2}(1/P_1)$$



- Method:
  - For l_2: computational
  - For general l_s: analytic

$\rho(c)$ for l_2

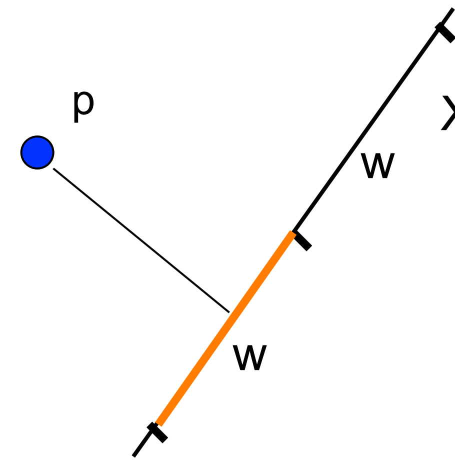


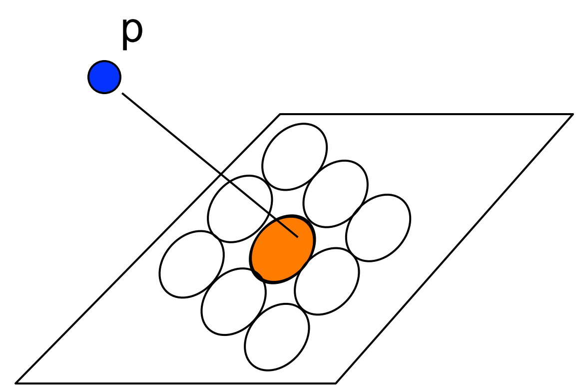
- Improvement not dramatic
- But the hash function very simple and works directly in l_2
  - Basis for the Exact Euclidean LSH package (E2LSH)

New LSH scheme

[Andoni-Indyk'06]

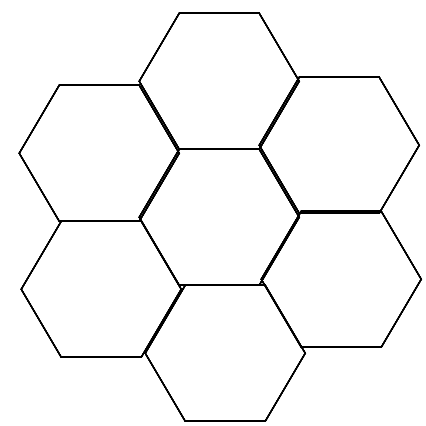
- Instead of projecting onto R^1, project onto R^t, for constant t
- Intervals → lattice of balls
  - Can hit empty space, so hash until a ball is hit
- Analysis:
  - $\rho=1/c^2 + O(\log t / t^{1/2})$
  - Time to hash is $t^{O(t)}$
  - Total query time: $dn^{1/c^2+o(1)}$
- [Motwani-Naor-Panigrahy'06]: LSH in l_2 must have $\rho \geq 0.45/c^2$
- [O'Donnell-Wu-Zhou'09]: $\rho \geq 1/c^2 - o(1)$





New LSH scheme, ctd.

- How does it work in practice ?
- The time $t^{O(t)} dn^{1/c^2+f(t)}$ is not very practical
  - Need $t \approx 30$ to see some improvement
- Idea: a different decomposition of R^t
  - Replace random balls by Voronoi diagram of a lattice
  - For specific lattices, finding a cell containing a point can be very fast →fast hashing



Leech Lattice LSH

- Use Leech lattice in R^{24} , t=24
  - Largest kissing number in 24D: 196560
  - Conjectured largest packing density in 24D
  - 24 is 42 in reverse…
- Very fast (bounded) decoder: about 519 operations [Amrani-Beery'94]
- Performance of that decoder for c=2:
  - $1/c^2$ 0.25
  - $1/c$ 0.50
  - Leech LSH, any dimension: $\rho \approx 0.36$
  - Leech LSH, 24D (no projection): $\rho \approx 0.26$

LSH Zoo

- Have seen:
  - Hamming metric: projecting on coordinates
  - L_2 :random projection+quantization
- Other (provable):
  - L_1 norm: random shifted grid [Andoni-Indyk'05] (cf. [Bern'93])
  - Vector angle [Charikar'02] based on [Goemans-Williamson'94]
  - Jaccard coefficient [Broder'97]

$$J(A,B) = |A \cap B| / |A \cup B|$$

- Other (empirical): inscribed polytopes [Terasawa-Tanaka'07], orthogonal partition [Neylon'10]
- Other (applied): semantic hashing, spectral hashing, kernelized LSH, Laplacian co-hashing, , BoostSSC, WTA hashing,…

Open questions

- Practically efficient LSH scheme for L_2 with $\rho = 1/c^2$
- Theoretically more efficient, e.g., decoder with $t^{O(1)}$ time
- Understand data adaptation (a.k.a. semantic hashing, spectral hashing, kernelized LSH, Laplacian co-hashing, , BoostSSC, WTA hashing,…)
  - Would like an algorithm that is
    - correct (with desired probability) for any query
    - "efficient" on "good" data

Min-wise hashing

- In many applications, the vectors tend to be quite sparse (high dimension, very few 1's)
- Easier to think about them as sets
- For two sets A,B, define the Jaccard coefficient:

$$J(A,B)=|A \cap B|/|A \cup B|$$

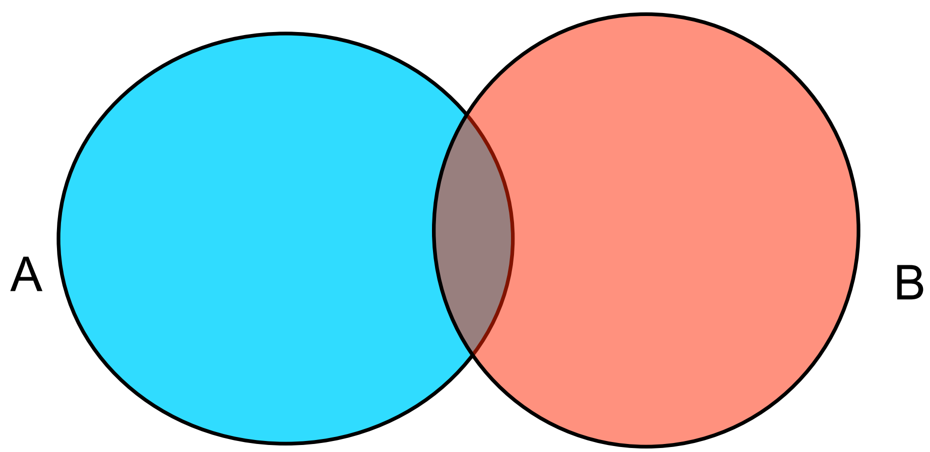
  - If $A=B$ then $J(A,B)=1$
  - If A,B disjoint then $J(A,B)=0$
- How to compute short sketches of sets that preserve $J(.)$?

Hashing

- Mapping:

$$g(A)=\min_{a \in A} h(a)$$

  where h is a random permutation of the elements in the universe
- Fact: $\Pr[g(A)=g(B)]=J(A,B)$
- Proof: Where is $\min(h(A) \cup h(B))$?



Random hyperplane

- Let u,v be unit vectors in R^m
- Angular distance:

  $A(u,v)=$angle between u and v
- Sketching:
  - Choose a random unit vector r
  - Define $s(u)=\mathrm{sign}(u*r)$

Probabilities

- What is the probability of

  sign(u*r)≠sign(v*r) ?

- It is A(u,v)/π