Geometric Data analysis

Curse of dimensionality and LSH

Ioannis Emiris

National & Kapodistrian U. Athens
ATHENA Research Center, Greece

Fall 2020

Nearest Neighbor

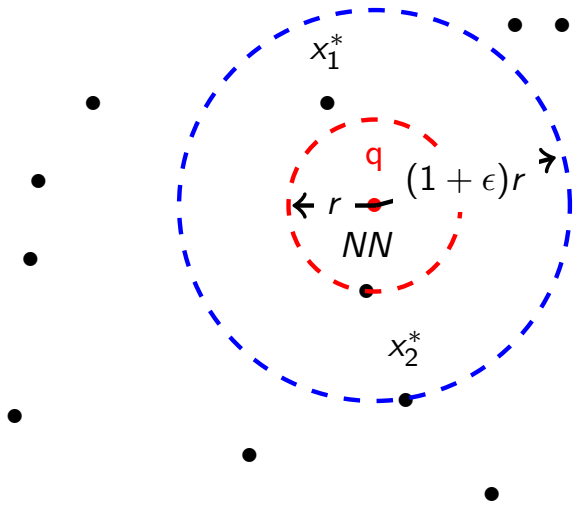
Exact NN

In d-dimensional space D, given set $P \subset D$, and query point $q \in D$, its NN is point $p_0 \in P$: $\text{dist}(p_0, q) \leq \text{dist}(p, q)$, $\forall p \in P$.

Approximate NN

Given set $P \subset D$, approximation factor $1 > \epsilon > 0$, and query point q, an ϵ-NN, or ANN, is any point $p_0 \in P$:

$$\text{dist}(p_0, q) \leq (1 + \epsilon)\text{dist}(p, q), \quad \forall p \in P.$$



Approximate Near-neighbor problem

Definition ((r, c)-Near neighbor)

Preprocess: finite set of points P.

Query: point q, radius r, approximation factor $c > 1$.

- *Range search: Report all $p \in P$ s.t. $dist(q, p) \leq c \cdot r$.*
- *Augmented decision problem (with witness):*
  - *If $\exists\, p_0$ within radius r, output YES and any $p : dist(q, p) \leq c \cdot r$.*
  - *If $\nexists\, p$ within radius $c \cdot r$, then report NO.*
  - *If none of above, report either NO, or YES and some p_0 in cr-ball*

ANN to Near-Neighbor (bounding radius)

Lemma

$c(1+\epsilon)$-ANN reduces to $\log_{1+\epsilon}\Delta$ instances of $((1+\epsilon)^i, c)$-Near-Neighbor decision problems, for $i = \log_{1+\epsilon}\Delta, \ldots, 2, 1$, where $\Delta =$ bounding radius.

Proof

For any query, run ith and $(i+1)$st augmented decision problems:
– Balls cannot be both empty.
– While both answers positive, continue with new radius $(1+\epsilon)^{i-1}$.
– When answers differ, we obtain p_0 within radius $c(1+\epsilon)^{i+1}$, whereas none exists within radius $c(1+\epsilon)^i$.

ANN to Near-Neighbor ($\log n$)

Theorem (Har-Peled,Indyk,Motwani'12)

For set P in a metric space, and $c > 1$, $\delta \ll 1$, $\gamma \in (1/n, 1)$, given a data structure solving the decision (r, c)-Near Neighbor problem with failure probability δ, using space S, and query time Q, there exists a data structure using

$$O(S \log^2 n/\gamma) \text{ space,}$$

answering ϵ-ANN, $1 + \epsilon = \Theta(c)(1 + O(\gamma))$, with query time

$$O(Q \log n),$$

and failure probability $O(\delta \log n)$.

Replaces the dependence on $\log_{1+\epsilon} \Delta$ (bounding radius) by $\log n$.

NN in $\mathbb{R}$

Sort/store n points in balanced binary search tree (red-black, AVL), use binary search for queries:

- Prepreprocessing in $O(n \log n)$ time
- Data structure requiring $O(n)$ space
- Answer the query in $O(\log n)$ time

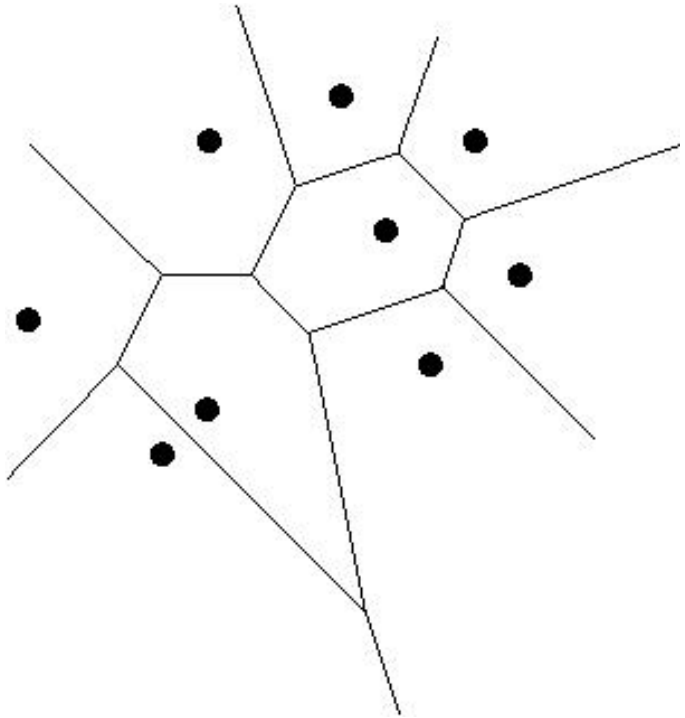
A hash-table with M buckets offers a solution with

- preprocessing in $O(M + n) = O(n)$ time
- space $O(M + n) = O(n)$
- query time $O(1)$

assuming constant time for hashing and constant number of items per bucket.

NN in $\mathbb{R}^2$

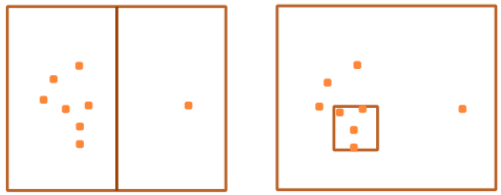
- Preprocessing: Voronoi Diagram in $O(n \log n)$.
- Storage $= O(n)$.
- Given query q, find the cell it belongs (point location) in $O(\log n)$. NN $=$ site of cell containing q.



NN in $\mathbb{R}^d$

Curse of Dimensionality: Voronoi diagram $= O(n^{\lceil d/2 \rceil})$.
Can query be polynomial in d and sublinear in n?

Approximate ϵ-NN:



- BBD-trees: Sp $= O(dn)$, Q $= O((d/\epsilon)^d \log n)$.
- Locality sensitive hashing (LSH): Sp $\simeq dn^{1+\rho}$, Q $\simeq dn^{\rho}$, $\rho = 1/(1+\epsilon)^2$ [Indyk,Motwani'98] [Andoni,Indyk'08]; various metrics. Data-dependent: $\rho = \frac{1}{2(1+\epsilon)^2-1} + o(1)$ [Andoni,Razenshteyn'14].
- Projection-based methods: Sp $= O^*(dn)$, Q $\simeq dn^{1-\Theta(\epsilon^2)}$ [E,Psarros,et al.15-18].

Complexity and extension

ANN in $\mathbb{R}^d$ [Arya, Mount et al.]

Let $S(n), Q(n)$ denote space and ANN query time. Ignoring log factors,

$$S(n)Q^2(n) = \Omega^*\left(\frac{n}{\epsilon^{d-1}}\right).$$

Definition (k-ANNs)

For pointset P and $0 < \epsilon < 1$, given query point q and $k \in \mathbb{N}^*$, find a sequence $S = [p_1, \cdots, p_k] \subset P$ of distinct points s.t. p_i is an ϵ-ANN of the i-th exact NN of q.

BBD-trees return k-ANN in $O((k + (d/\epsilon)^d) \log n)$. Moreover, if $S' \subseteq P$ are the points visited by the search and $S \subseteq S'$ the k points nearest to q among S', then $\forall x \in P \setminus S'$, $(1+\epsilon)dist(x, q) > dist(p_k, q)$.

Outline

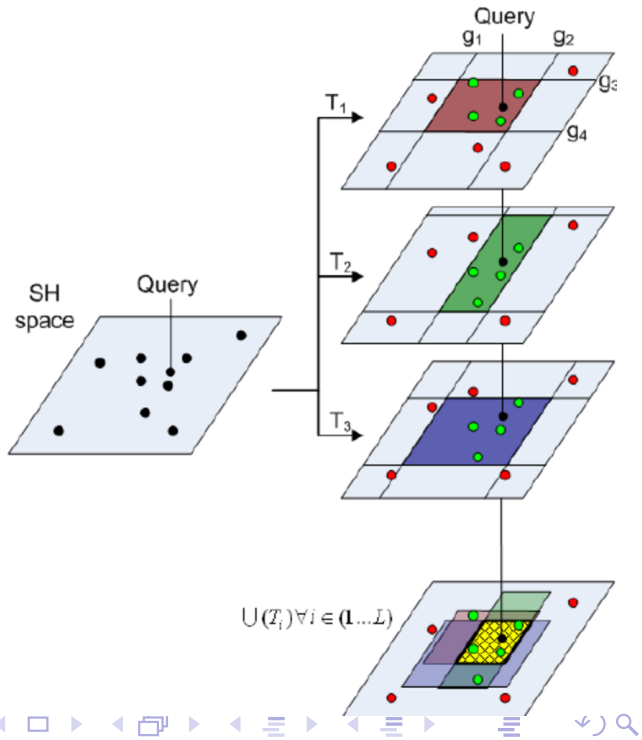
LSH idea

LSH Family

Let $r_1 < r_2$, probabilities $p_1 > p_2$. Function family H is (r_1, r_2, p_1, p_2)-sensitive if, for any points $p \neq q$ and any randomly selected $h \in_R H$:

- if $\text{dist}(p, q) \leq r_1$, then $\text{prob}[h(q) = h(p)] \geq p_1$,
- if $\text{dist}(p, q) \geq r_2$, then $\text{prob}[h(q) = h(p)] \leq p_2$.

$h \in_R H$: h randomly (uniformly) chosen.
Idea: increase collisions of similar strings.
Typically $r_2 = c \cdot r_1$, $c > 1$.



Amplification

Hash-table

LSH creates hash-table using amplified hash functions by concatenation:

$$g(p) = [h_1(p) \,|\, h_2(p) \,|\, \cdots \,|\, h_k(p)],$$

where every $h_i \in_R H$ is distributed uniformly (with repetition) in H.

Some h_i may be chosen more than once for a given g or for different g's. Also called AND-amplification.

Lemma

g is (r_1, r_2, p_1^k, p_2^k)-sensitive.

Large $k \Rightarrow$ larger gap between p_1, p_2. Practical choices are $k = 4$ to 6.

Construction

Preprocess

- Having defined H and amplified hash-function g:
- Select $L\,(= n^{\rho})$ hash-functions $g_1, \ldots, g_L$.
- Initialize L hashtables, hash all points to all tables using g (or ϕ).

Goal: L so that it has $\Theta(1)$ points per bucket.
L is 5 up to function of n, and HashTable size $= \Theta(n)$.

Overall construction is OR-amplification of g: points are "neighbors" if $\exists i$ for which they lie in same bucket.

Lemma

If g is $(r_1, r_2, \delta_1, \delta_2)$-sensitive, then the overall construction represents a $(r_1, r_2, 1-(1-\delta_1)^L, 1-(1-\delta)^L)$-sensitive function.

Range Search

Range (r, c)-Near Neighbor search

Input: r, c, query q
for i from 1 to L **do**
  for each item p in bucket $g_i(q)$ **do**
    if dist$(q, p) < cr$ **then output** p
    end if
  end for
end for

Decision problem: "**return** p" instead of "**output** p".
At end "**return** FAIL"; also FAIL after threshold on #examined points reached.

NN search

Approximate NN

Input: query q
Let $b \leftarrow$Null; $d_b \leftarrow \infty$
for i from 1 to L **do**
  for each item p in bucket $g_i(q)$ **do**
    if large #checked items (e.g. $> 3L$) **then return** b // threshold
    end if
    if $\text{dist}(q,p) < d_b$ **then** $b \leftarrow p$; $d_b \leftarrow \text{dist}(q,p)$
    end if
  end for
  return b
end for

Theoretical bounds for $c(1+\epsilon)$-NN by reduction to $((1+\epsilon)^i, c)$-Neighbor decision problems, $i = 1, 2, \ldots, \lg_{1+\epsilon} \Delta$.

Analysis of bad events

From the definition, with $p_1 > p_2$: ▸ LSH-Defn

$$\|p - q\| \geq cr \implies \mathrm{P}_g\left[g(p) = g(q)\right] \leq p_2^k.$$

Set $k = \frac{\log n}{\log(1/p_2)} = \log_{p_2}(1/n)$, then bound exp'd #falsePositives:

$$\mathbb{E}_g\left[\#x:\ g(x) = g(q), \|x - q\| \geq cr\right] \leq n \cdot p_2^k = 1.$$

For L hashtables, the expected number of false positives is $\leq L$.
Markov's inequality: $\mathrm{P}[X \geq a] \leq \mathbb{E}[X]/a,\ X \geq 0$.
Hence, $\mathrm{P}[\#\text{falsePositives} \geq 3L] \leq 1/3$.

Analysis (cont'd)

True positive:

$$\|p-q\| \leq r \implies \mathrm{P}_g\left[g(p)=g(q)\right] \geq n^{-\frac{\log(1/p_1)}{\log(1/p_2)}}.$$

Set $\lambda = \frac{\log(1/p_1)}{\log(1/p_2)} < 1$, thus missing probability $\leq 1 - n^{-\lambda}$.

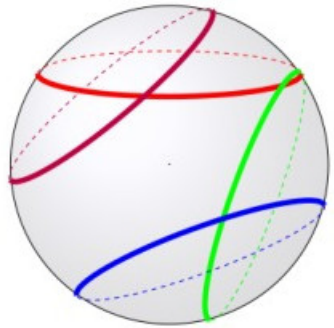
Probability missing one true positive (false negative) in L tables $=$

$$\mathrm{P}_{g_1,\ldots,g_L}\left[\forall i \in [L] : g_i(p) \neq g_i(q)\right] \leq \left(1-\frac{1}{n^\lambda}\right)^L \leq \mathrm{e}^{-\frac{L}{n^\lambda}},$$

using $1+x \leq \mathrm{e}^x$. Set $L = n^\lambda$, and by union bound over two bad events, the total failure probability $\leq \mathrm{P}_1 + \mathrm{P}_2 = 1/3 + 1/\mathrm{e}$.

The constant probability of success can be amplified to $1 - o(1)$ by building logarithmically many independent data structures.

Known LSH-able metrics



- Hamming distance,
- ℓ_2 (Euclidean) distance,
- ℓ_1 (Manhattan) distance,
- ℓ_k distance for any $k \in [1, 2)$,
- ℓ_2 distance on a sphere,
- Cosine similarity,
- Jaccard coefficient.

Recall ℓ_k norm: $\quad \mathrm{dist}_{\ell_k}(x, y) = \sqrt[k]{\sum_{i=1}^{d} |x_i - y_i|^k}.$

Outline

Hamming distance

Definition

Given strings x, y of length d, their Hamming distance $\mathrm{d}_H(x, y)$ is the number of positions at which x and y differ.

Example

Let $x = 10010$ and $y = 10100$. Then, $\mathrm{d}_H(x, y) = 2$.

Definition of hash functions

Given $x = (x_1, \ldots, x_d) \in \{0,1\}^d$:

$$H = \{h_i(x) = x_i \ : \ i = 1, \ldots, d\}.$$

Obviously, $|H| = d$.
Pick uniformly at random $h \in_R H$: Then $\text{prob}[h(x) \neq h(y)] = \mathrm{d}_H(x,y)/d$,

$$\text{prob}[h(x) = h(y)] = 1 - \mathrm{d}_H(x,y)/d.$$

Corollary

The family H is $(r_1, r_2, 1 - r_1/d, 1 - r_2/d)$-sensitive, for $r_1 < r_2$.

LSH in Hamming Space

However probabilities $1 - r_1/d$, $1 - r_2/d$ can be close to each other.

Amplification

Given parameter k, define new family G by concatenation. G is the set of all functions

$$g : \{0,1\}^d \to \{0,1\}^k \ : \ g(x) = [h_{i_1}(x) \,|\, \cdots \,|\, h_{i_k}(x)],$$

where $h_{i_j} \in_R H$ is uniformly chosen for $j = 1, \ldots, k$.

– We must have $L < |G| = d^k$, so as to pick L different g's.
– The range of each g is $[0, 2^k)$, so $k < \lg n$.
– So k may be close to $\lg n - 1$ (unlike later cases where $k = 4, 5$)

Build Hash-tables

Build

Pick uniformly at random L functions $g_1, \ldots, g_L \in_R G$, using $h_i \in_R H$ (chosen uniformly with repetition).

for i from 1 to L **do**

Initialize (one-dim) hash-table T_i of size 2^k:

for each $p \in P$, store p in bucket $g_i(p)$.

end for

Complexity

Build $= O(Lnk)$ H-function calls, where $L \simeq n^{\rho}$.

Store n strings $= O(dn)$ bits,

L hashtables and n pointers to strings per table $= O(Ln)$ pointers.

(r, c)-Neighbors: Query $= O(L(k + d))$, assuming $O(1)$ strings per bucket.

Outline

Euclidean Space

Recall: $\text{dist}_{\ell_2}(x,y)^2 = \sum_{i=1}^{d}(x_i - y_i)^2$.

Definition

Let d-vector $v \sim \mathcal{N}(0,1)^d$ *have coordinates identically independently distributed (i.i.d.) by the standard normal (next slide).*
Set "window" $w \in \mathbb{N}^*$ *for the entire algorithm, pick single-precision* `real` t *uniformly* $\in_R [0, w)$. *For point* $p \in \mathbb{R}^d$, *define:*

$$h(p) = \lfloor \frac{p \cdot v + t}{w} \rfloor \in \mathbb{Z}.$$

– Essentially project p on line of v, shift by t, partition in cells of length w
– Generally $w = 4$ is OK but should increase for range queries of large r
– Also $k = 4$ (but can go up to 10), and L may be 5 (up to 30).

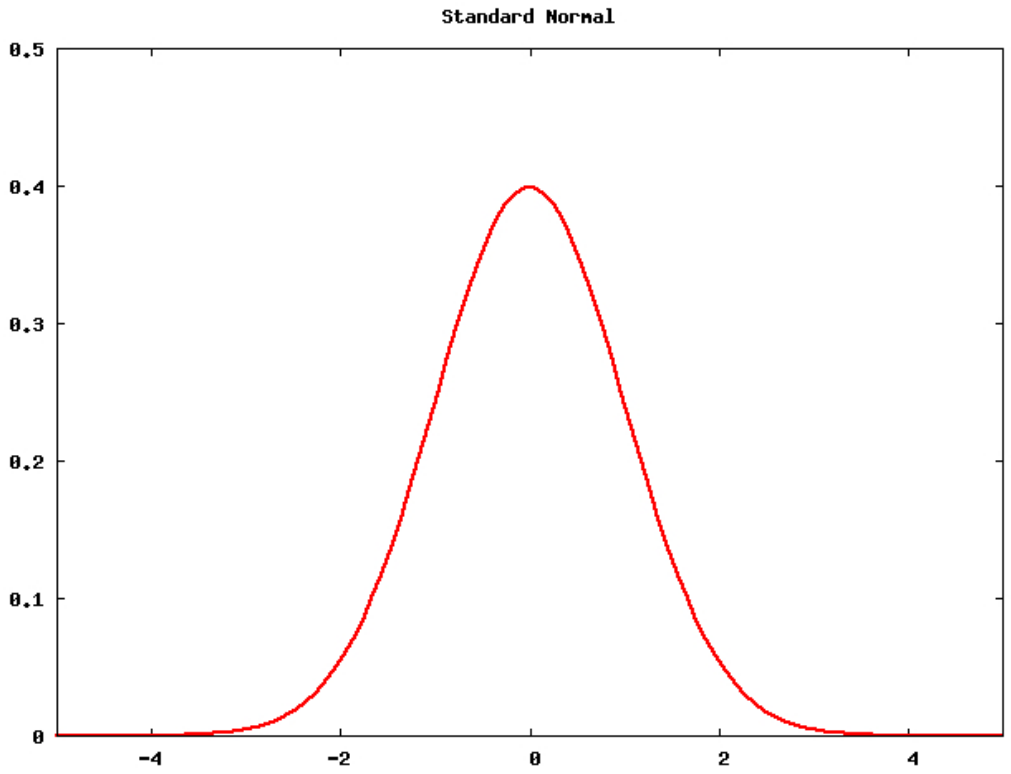
Normal distribution

Vector $v \sim \mathcal{N}(0,1)^d$ has single-precision `real` coordinates distributed according to the standard normal (Gaussian) distribution:

$$v_i \sim \mathcal{N}(0,1), \quad i = 1, 2, \ldots, d,$$

with mean $\mu = 0$, variance $\sigma^2 = 1$ (σ is the standard deviation).

The bell curve:



Normal from Uniform

Given uniform generator [Wikipedia]:

- Marsaglia: Use independent uniform $U, V \in_R (-1, 1)$, $S = U^2 + V^2$. If $S \geq 1$ then start over, otherwise:

$$X = U\sqrt{\frac{-2\ln S}{S}}, \qquad Y = V\sqrt{\frac{-2\ln S}{S}}$$

  are independent and standard normally distributed.

The U, V, X, Y are single-precision `reals`.

Implementation

Given (elementary) hash h_i, set amplified hash $g = [h_1(p)|\cdots|h_k(p)]$.
Yields huge table, many empty buckets. Use random linear combination:

Implement a 1-dim hash-table with indexing function:

$$\phi(p) = (r_1 h_1(p) + r_2 h_2(p) + \cdots + r_k h_k(p) \bmod M) \bmod \text{TableSize},$$

`int` $r_i \in_R \mathbb{Z}$, prime $M = 2^{32-2}$, TableSize $= n/8$ (e.g.).

Note ϕ computed in `int` arithmetic, if all $h_i(p), r_i$ are `int` (≤ 32 bits).
Recall $(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$.

Outline

LSH for Cosine similarity

Consider $\mathbb{R}^d$, equipped with cosine similarity of two vectors:

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|},$$

which expresses the angle between vectors x, y.

Similarity is inversely proportional of distance: For unit x, y, $\text{dist}^2(x, y) = 2 - 2\cos(x, y)$. (not a metric: no triangular ineq.)

For comparing documents or, generally, very long vectors (typically sparse), based on direction only, not length.

Hyperplane LSH

Definition

Let $r_i \sim \mathcal{N}(0,1)^d$, *with each* `real` *coordinate iid* $\mathcal{N}(0,1)^d$. *Define*

$$h_i(x) = \begin{cases} 1, & \text{if } r_i \cdot x \geq 0 \\ 0, & \text{if } r_i \cdot x < 0 \end{cases}.$$

Then $H = \{h_i(x) \mid \text{for every } r_i\}$ *is a locality sensitive family.*

Hyperplane LSH (cont'd)

Intuition

Each r_i is normal to a hyperplane. Two vectors lying on same side of many hyperplanes are very likely similar.

Lemma

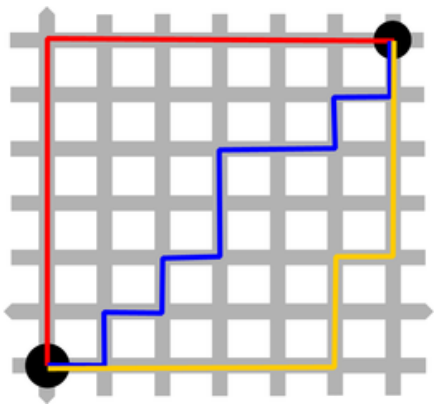
Two vectors match with probability proportional to their cosine.

Amplification: Given parameter k, define new family G by concatenation:

$$G = \{g : \mathbb{R}^d \to \{0,1\}^k \mid g(x) = [h_1(x) \mid h_2(x) \mid \cdots \mid h_k(x)]\}.$$

Outline

Hash-function



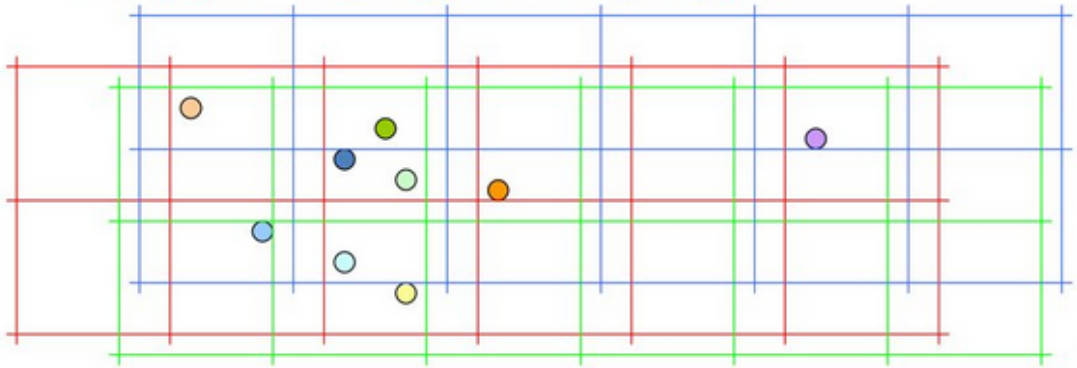
$$\text{dist}_{\ell_1}(x, y) = \sum_{i=1}^{d} |x_i - y_i|.$$

Consider $\mathbb{R}^d$, r is the radius of range search. Pick real $w \gg r$ once, then h specified by independent, real, uniformly distributed $s_0, \ldots, s_{d-1} \in_R [0, w)$.

$$\text{Let } x \mapsto a \in \mathbb{Z}^d : \; a_i = \left\lfloor \frac{x_i - s_i}{w} \right\rfloor \in \mathbb{Z}, \quad i = 0, 1, \ldots, d-1,$$

shifts x by s, then projects to bottom-left corner of its cell in d-dimensional grid of size w.

LSH function for 1-dim hashtable



$$h(x) = a_{d-1} + m \cdot a_{d-2} + \cdots + m^{d-1} \cdot a_0 \bmod M \ \in \mathbb{N},$$

for $\max_i a_i < m < M/2$ for entire algorithm, e.g. $M = 2^{\lfloor 32/k \rfloor}$.

By concatenation, amplified function $g(x) = [h_1(x)|h_2(x)|\cdots|h_k(x)]$.

Lemma. (c, r)-Near-Neighbor decided whp: check $3L$ candidate points, $L = n^\rho$, $\rho = \ln p_1 / \ln p_2 = 1/c + O(r/w)$ [`TarsosLSH` software]

Outline

Distance Metric

Definition (Distance Metric)

A distance metric $d : D^2 \to \mathbb{R}$ *is a function that satisfies:*

- *Non-negativity:* $d(x, y) \geq 0$
- *Isolation:* $x \neq y \Leftrightarrow d(x, y) > 0$
- *Symmetry:* $d(x, y) = d(y, x)$
- *Triangle inequality:* $d(x, y) \leq d(x, z) + d(z, y)$

It follows that $d(x, x) = 0$, and $|d(x, z) - d(z, y)| \leq d(x, y)$.

Example

Distances in vector spaces (e.g. Hamming, Euclidean, Manhattan, any ℓ_k *metric) are all distance metrics. Compact (vector) representation allow to compute mean, total order. . .*

Distance Based Hashing (DBH)

- LSH needs specific families of LSH functions, so it is not applicable to novel, or not studied, metrics.
- DBH produces hash functions tailored to the space by considering only calls to the distance measure and by making no assumptions about the domain.
- Due to the generality of the method there are no theoretical guarantees

[Athitsos,et al.08]

DBH family of functions

Consider metric space (D, d) and data $P \subset D$. Construct family of functions H that behaves like LSH.

Definition (Line projection)

Given $x_1, x_2 \in P \subset D$ define the line projection function

$$h^{x_1,x_2} : D \to \mathbb{R} : x \mapsto \frac{\mathrm{d}(x, x_1)^2 + \mathrm{d}(x_1, x_2)^2 - \mathrm{d}(x, x_2)^2}{2\mathrm{d}(x_1, x_2)}.$$

If D is Euclidean, this is the signed length of projecting vector (x_1, x) on line (x_1, x_2), x_2 lying on the positive axis.

Discretization and balancing

Definition (Discretization)

For hashing, discretize h^{x_1,x_2} *by using thresholds* $t_1, t_2 \in \mathbb{R} \cup \{\pm\infty\}$*:*

$$h^{x_1,x_2}_{t_1,t_2} : D \to \{0,1\} : x \mapsto \begin{cases} 1, & \text{if } h^{x_1,x_2}(x) \in [t_1, t_2] \\ 0, & \text{otherwise} \end{cases}$$

The t_1, t_2 should map half the objects of P to 0 and the other half to 1:

Definition (Set of valid thresholds V)

For $x_1, x_2 \in P$*, the set of thresholds yielding "balanced" h is*

$$V(x_1, x_2) = \{[t_1, t_2] : prob_{x \in P}[h^{x_1,x_2}_{t_1,t_2}(x) = 0] = 1/2\}.$$

Hash functions

Definition

Consider the "balanced" functions

$$H = \{h^{x_1,x_2}_{t_1,t_2} : x_1, x_2 \in P \text{ and } [t_1, t_2] \in V(x_1, x_2)\}.$$

Using random $h_i \in_R H$ we define L hash functions by concatenation

$$g_i(x) = [h_{i1}(x) \,|\, h_{i2}(x) \,|\, \cdots \,|\, h_{ik}(x)],\ i = 1, \ldots, L.$$

Implement:

– Pick $x_1, x_2 \in_R P$ uniformly among points for which oracle/distance matrix defined; this defines $h^{x_1,x_2}(\cdot)$.

– Evaluate $h^{x_1,x_2}(x) \in \mathbb{R}$ for all $x \in P$ (or a large sample).

– Set $t_1 =$ median of $\{h^{x_1,x_2}(x) : x \in P\}$, $t_2 = \infty$; or at the $1/4, 3/4$ mark.