Ανάπτυξη Λογισμικού για Δυσεπίλυτα Αλγοριθμικά Προβλήματα

Ενότητα 1: Nearest neighbors

Γιάννης Εμίρης

Τμήμα Πληροφορικής & Τηλεπικοινωνιών
Πανεπιστήμιο Αθηνών

Χειμερινό 2017

Outline

1. Problem statement

2. Locality sensitive hashing
   - Euclidean space
   - Cosine similarity

3. Random storage in the binary cube

Nearest Neighbor

Exact NN

Let us consider ambient (d-dimensional) space D. Given set $P \subset D$, and query point $q \in D$, its NN is point $p_0 \in P$:

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

Approximate NN

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

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

Two (r, c)-neighbor problems

Approximation factor $c = 1 + \epsilon > 1$.

Definition

Input: finite set of strings/points P, approximation factor $c > 1$, radius r, query q.

Output of (r, c)-Neighbor problem:

-- Range search: Report all $p \in P$ s.t. $dist(q, p) \leq c \cdot r$.
In practice, if c is not given, take $c = 1$.

-- Decision problem. If $\exists\, p_0$ within radius r, output any $p : dist(q, p) \leq c \cdot r$;
otherwise: output a point p within cr or NULL.

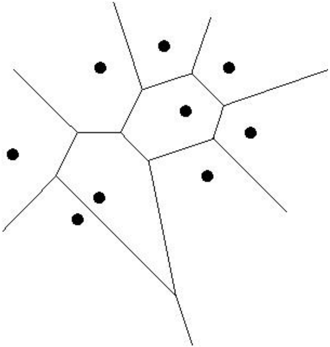
NN in $\mathbb{R}$

Sort/store the points, use binary search for queries, then:

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

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.



Exact NN:

- Voronoi diagram $= O(n^{\lceil d/2 \rceil})$;
- State of the art: kd-trees: Sp $= O(dn)$, Query $\simeq O(d \cdot n^{1-1/d})$.

Hence the Curse of Dimensionality: Can we solve NN in poly-time in d and faster than linear-time in n?

Approximate ϵ-NN:

- BBD-tree (Arya,Mount et al.'94'98) and kd-tree: optimal Sp $= O(dn)$, $Q \simeq O(\log^d n)$.
- Locality sensitive hashing (LSH): Sp $\simeq dn^{1+1/(1+\epsilon)}$, Q $\simeq dn^{1/(1+\epsilon)}$.
- New projection-based methods: Optimal Sp $= O(dn)$, Q $\simeq dn^{\rho}$, ρ somewhat larger than in LSH (Emiris,Psarros et al.)

Outline

1. Problem statement
2. Locality sensitive hashing
   - Euclidean space
   - Cosine similarity
3. Random storage in the binary cube

LSH definition

Classic hash-tables answer membership. Idea: use hash-table for proximity query by mapping similar strings to same bucket ie. increase collisions of similar strings.

LSH Family

Let $r_1 < r_2$, probabilities $P_1 > P_2$. A family H of functions is (r_1, r_2, P_1, P_2)-sensitive if, for any points $p \neq q$ and any randomly selected function $h \in_R H$,

- 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 $\text{prob}[h(q) = h(p)] \leq P_2$.

Notation: $h \in_R H$ means h is randomly chosen (following the uniform distribution) from H.

(symmetric) LSH (Wikipedia)

Amplification

Hash-table

LSH creates every hash-table using an amplified hash function g by combining k functions $h_i \in_R H$, chosen uniformly at random (with repetition) from H.

This implies some h_i may be chosen more than once for a given g, or for different g's (LSH builds several hash-tables).

Typically g is defined by concatenation:

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

Construction

Preprocess

- Having defined H and hash-function g:
- Randomly select L amplified hash functions $g_1, \ldots, g_L$.
- Initialize L hash-tables, hash all points to all tables using g.

Large $k \Rightarrow$ larger gap between P_1, P_2. Practical choices are $k = 4$ to 6, $L = 5$ (or 6), and HashTable size $n/4$ (alternatively $n/2$ or $n/8$).

Range (r, c)-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 $\text{dist}(q, p) < cr$ **then output** p
    end if
  end for
end for

In practice, if c not given assume $c = 1$.

Decision problem: "**return** p" instead of "**output** p".
At end "**return** FAIL"; may also FAIL if many examined points.

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 number of retrieved items (e.g. $> 3L$) **then return** b // trick
    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, $i = 1, 2, \ldots, \log_{1+\epsilon} d$.

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

(Andoni-Indyk:J.ACM'08)

Outline

1. Problem statement
2. Locality sensitive hashing
   - Euclidean space
   - Cosine similarity
3. Random storage in the binary cube

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:*

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

-- Essentially project p on the line of v, shift by t, partition into cells of length w.
-- In general, $w = 4$ is good but should increase for range queries with large r.
-- Also $k = 4$ (but can go up to 10), and L may be 5 (up to 30).

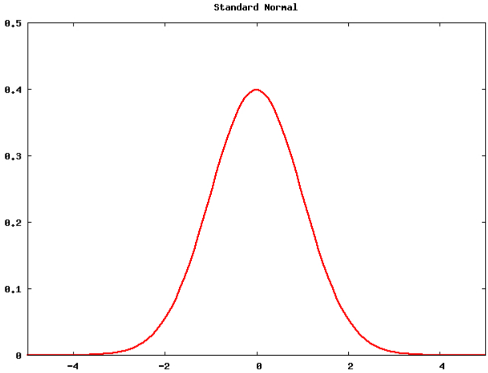
Normal distribution

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 `real`s.

Hash-table

Amplified function $g(p) = [h_1(p)|h_2(p)| \cdots |h_k(p)]$ would yield a k-dimensional hashtable with many empty buckets.

So we define ϕ as random combination of the h_i's as follows:

Classic hash-function

We build a 1-dim hash-table with classic index:

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

s.t. `int` $r_i \in_R \mathbb{Z}$, prime $M = 2^{32} - 5$ if $h_i(p)$ are `int`, TableSize$= n/2$ (or $n/4$)

Notice ϕ computed in `int` arithmetic, if all $h_i(p)$, r_i are `int` ($\leq$ 32 bits).

Can have smaller TableSize$= n/8$ or $n/16$ (heuristic choice).

Recall $(a \square b) \bmod m = ((a \bmod m) \square (b \bmod m)) \bmod m$.

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. Notice similarity is opposite of distance: $\text{dist}(x, y) = 1 - \cos(x, y)$.

For comparing documents or, generally, long vectors based on direction only, not length.

Shall be approximated by random projections (next slide).

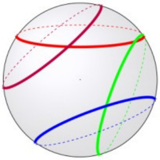
Random projection

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.*



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:

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

Hyperplane LSH

Observation

Cosine dissimilarity $\approx$ Euclidean distance on the sphere

Heuristic for the Euclidean distance

Bring the pointset to an *isotropic* position: find a point T s.t. all points are at a similar distance from T; make T the new origin. Use Hyperplane LSH to solve ANN for the Euclidean distance.

Computationally cheap: compute the mean T of a randomly sampled subset; make T the new origin.

Outline

Binary (0/1) hypercube

Projection

- Input: Metric space admitting family of LSH functions h_i.
- For each h_i, $f_i(h_i)$ maps buckets to $\{0, 1\}$ uniformly.
  If h_i maps points to $\{0, 1\}$, f_i is the identity.
- Define $f : x \mapsto [f_1(h_1(x)), \ldots, f_{d'}(h_{d'}(x))] \in \{0, 1\}^{d'}$.
- Preprocess: Store data in vertices of cube, dimension $d' = \lg n$.

Search

1. Project query
2. Check points in same vertex
3. Explore nearby vertices (Hamming distance) until x found: $\text{dist}(x, q) \leq (1 + \epsilon)r$, threshold#points checked, or threshold#vertices probed.

Important topologies

Theorem

For ℓ_1 and ℓ_2 metrics, this solves the ANN decision problem efficiently, thus yielding a solution for the ϵ-ANN optimization problem with space and preprocessing in $O^*(dn)$, and query time in $O^*(dn^\rho)$, $\rho = 1 - \Theta(\epsilon^2)$.

Project

- Main parameters:
  d': new (projection) dimension,
  $probes = \max \#$buckets searched,
  M: max #candidate points examined.

- Compare to exhaustive search.
- Print query time, storage, multiplicative error, #exact-answers.