Nearest Neighbor via Locality Sensitive Hashing

Piotr Indyk

Set Similarity Business

Set similarity: $D(A,B) = \frac{|A \cap B|}{|A \cup B|}$

- $\mathcal{H} = \{h_\pi : h_\pi(A) = \max_{a \in A} \pi(a)\}$

- $\Pr_{h \in \mathcal{H}}[h(A) = h(B)] = D(A,B)$

Questions:

- How to deal with π ?

- Can we extend $D(\cdot)$ to multisets ?

Permuting The Universe

- Hash all words to $U = \{0 \ldots u\}$
  (u large enough to make collisions unlikely)
- To permute U we can apply:
  - Linear permutation: $\pi(x) = ax + b \bmod u$,
    a and b random.
    * Easy to implement
    * Not random enough! E.g.,

$$Pr[h(\{0\}) = h(\{0 \ldots k\})] \approx \frac{\log k}{k}$$

  - Polynomials: $\pi(x) = a_0 + a_1 x_1 + \ldots a_k x^k$ mod u
    * Not permutations
      (but can bound the probability of collision)
    * For any $\epsilon > 0$, setting $k = O(\log 1/\epsilon)$ gives

$$Pr[h(A) = h(B)] = D(A, B) \pm \epsilon |A \cup B|$$

Extension to Multisets

Fuzzy logic:

- An occurence of x in A has a multiplicity. I.e., the characteristic function $\mu_A(x)$ is a non-negative integer.
- $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$
- $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$

Can extend similarity measure, and the min hashing to multisets.

Near Neighbor

(Dynamic) Approximate Near Neighbor:

- insertions/deletions
- if there is a point within distance r from q, return some point within distance $(1+\epsilon)r$ from q
  (r fixed)

Locality-Sensitive Hashing

A family $\mathcal{H} = \{h : U \to S\}$ is called *(r_1, r_2, P_1, P_2)-sensitive* for D if for any $q, p \in U$

- if $D(p, q) \leq r_1$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \geq P_1$,
- if $D(p, q) > r_2$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \leq P_2$.

We assume $P_1 > P_2$ and $r_1 < r_2$.

Examples

- Hamming metric $\{0,1\}^d$:
  - $\mathcal{H} = \{h(b_1 \dots b_d) = b_i, i = 1 \dots d\}$
    (i.e., sample one bit at random)
  - $\Pr_{\mathcal{H}}[h(q) = h(p)] = 1 - D(p,q)/d$
- Set similarity: $D(A,B) = \frac{|A \cap B|}{|A \cup B|}$
  - $\mathcal{H} = \{h_\pi : h_\pi(A) = \max_{a \in A} \pi(a)\}$
  - $\Pr_{h \in \mathcal{H}}[h(A) = h(B)] = D(A,B)$

Multi-index Hashing

To solve NN with parameters ϵ, r: set $r_1 = r$, $r_2 = (1+\epsilon)r$

Define $\mathcal{G} = \{g | g(p) = h_1(p).h_2(p) \ldots h_k(p)\}$

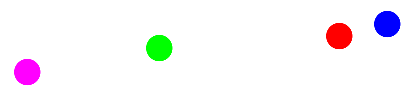
(for Hamming metric - sample k random bits)

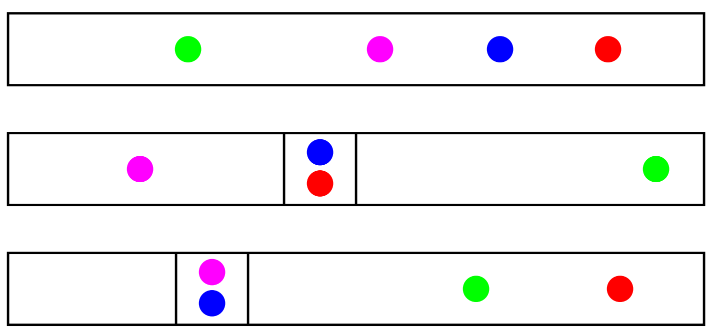
Preprocessing: prepare indices for $g_1, \ldots, g_l$

Add p: store p in buckets $g_1(p), \ldots, g_l(p)$
Delete p: remove p from buckets $g_1(p), \ldots, g_l(p)$

Query: check $g_1(q) \ldots g_l(q)$ and report the closest among first (say) $3l$ points

Time: $O(dl)$
Storage: $O(dn + nl)$





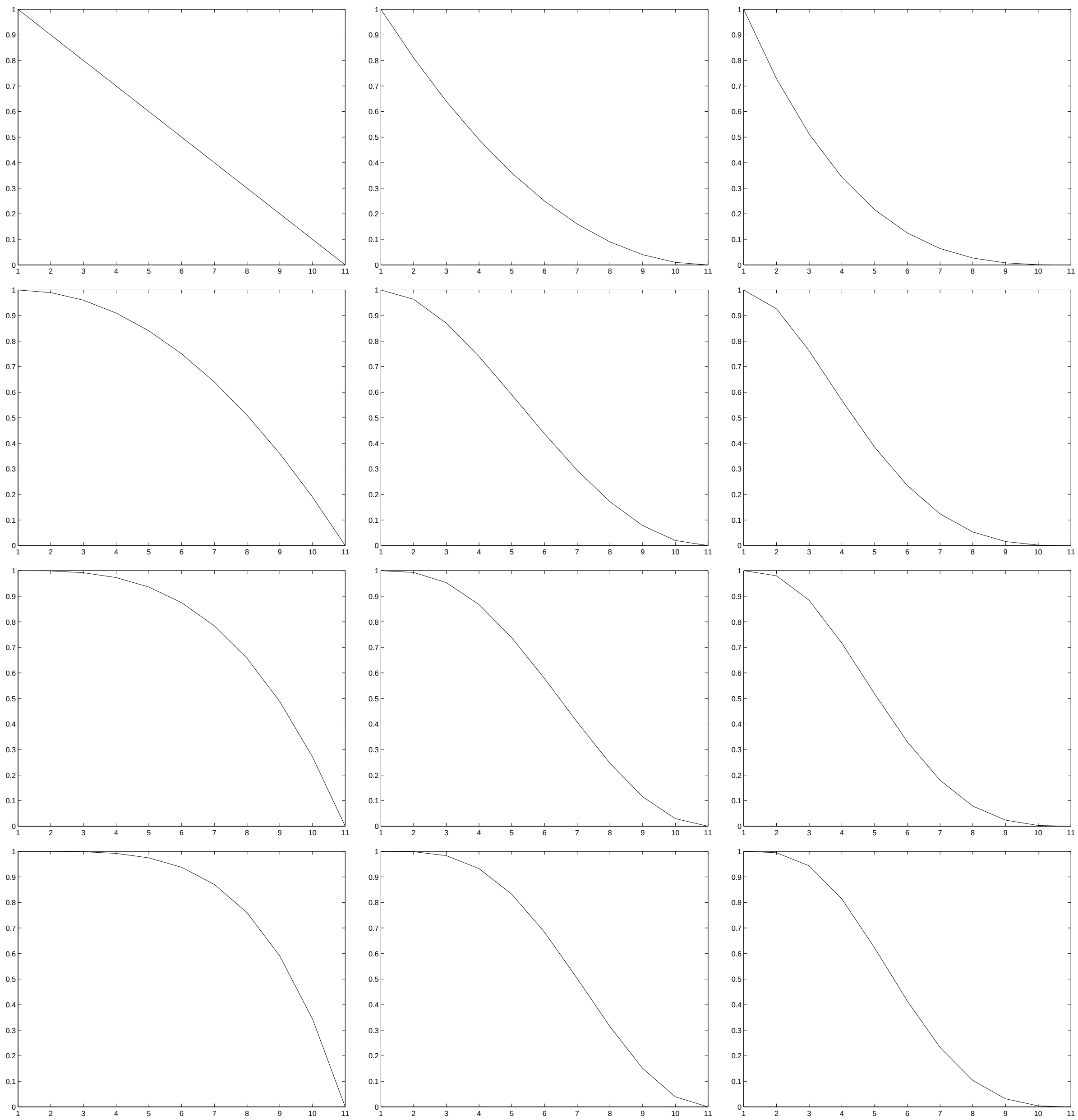
LSH: analysis

Question: How many indices l do we need ?

Theorem: Setting $l = n^\rho$ for $\rho = \frac{\log 1/P_1}{\log 1/P_2}$ is sufficient with constant probability.

(Hamming metric $\Rightarrow \rho = 1/(1+\epsilon)$)

“Proof”



LSH: Proof

Define:

- p^* - a point s.t. $D(q, p^*) \leq r$
- FAR(q) - all p s.t. $D(q, p) > (1 + \epsilon)r$
- BUCKET$_j(q)$ - all p s.t. $g_j(p) = g_j(q)$

Events:

- E_1: $\sum_{j=1}^{l} |\text{FAR}(q) \cap \text{BUCKET}_j(q)| \leq 3l$
- E_2: $g_j(p^*) = g_j(q)$ for some g_j, $1 \leq j \leq l$

Will show: $\Pr[\overline{E_1}] < 1/3$ and $\Pr[\overline{E_2}] < 1/e < 1/2$

Proof: Bad collisions

Let $p \in \mathsf{FAR}(q)$. Then

$$\Pr[p \in \mathsf{BUCKET}_j(q)] \leq P_2^k$$

For $k = \log_{1/P_2} n$

$$\Pr[p \in \mathsf{BUCKET}_j(q)] \leq P_2^{\log_{1/P_2} n} = 1/n$$

Thus

$$E[|\mathsf{FAR}(q) \cap \mathsf{BUCKET}_j(q)|] \leq n \cdot 1/n = 1$$

$$E[\sum_{j=1}^{l} |\mathsf{FAR}(q) \cap \mathsf{BUCKET}_j(q)|] \leq l$$

By Markov inequality

$$\Pr[\sum_{j=1}^{l} |\mathsf{FAR}(q) \cap \mathsf{BUCKET}_j(q)| > 3l] = \Pr[\overline{E_1}] \leq 1/3$$

Proof: Good collisions

For any g_j:

$$\Pr[g_j(p^*) = g_j(q)] \geq P_1^k = P_1^{\log_{1/P_2} n} = n^{-\frac{\log_{1/P_1} n}{\log_{1/P_2} n}} = n^{-\rho}$$

For $l = n^\rho$ we have

$$\begin{aligned} \Pr[\overline{E_2}] &\leq (1 - \Pr[g_j(p^*) = g_j(q)])^l \\ &\leq (1 - n^{-\rho})^{n^\rho} \\ &\leq 1/e \end{aligned}$$

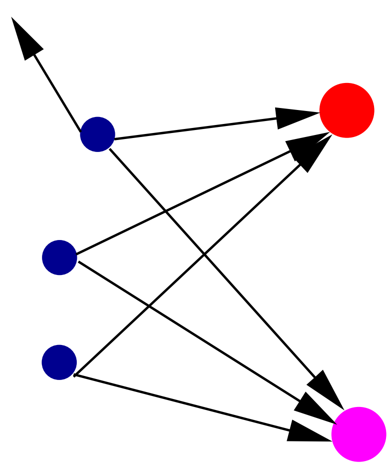
Web clustering

Goal: similarity search/clustering of the Web.

Problem: Huge data set !

Known approaches:

- detecting near-replicas [Broder-Glassman-Manasse-Zweig'97]
- link-based methods [Dean-Henzinger'99, Clever]



Would like to find pages with similar content based on text information (e.g., containing similar words).

Approach

- web page $P \to$ a set A of tuples of words:

$$P = \text{“This is an example web page”}$$

$$A = \{\text{“this is an”}, \text{“is an example”}, \ldots\}$$

- compare A and B by using

$$D(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- clustering ($\approx$ S-LINK):
  - take all pairs of similar documents
  - compute connected components

Algorithms

- [BGMZ'97]:
  - consistent sampling of tuples
  - finding all intersecting pairs $< A, B >$
  - filtering
  - performance (for 30 M pages) :
    * 10-tuples: $\approx 2 \cdot 10^{10} B$
    * 1-tuples: $\approx 10^{15} B^*$

- LSH (for 25 M pages):
  - 67 indices, 300 MB per index
  - essentially same time for 10-tuples
  - most important: same for 1-tuples

Syntactic Approach: Algorithm

- tuple size $= 10$
- 10-tuples of words
- algorithm:
  - sample (consistently) 1:25 tuples
  - list all

$$< DOC_1, DOC_2, TUPLE >$$

    s.t. $TUPLE$ appears in both DOC_1 and DOC_2
  - group $<, , >$ according to $< DOC_1, DOC_2, \cdot >$
  - compute the intersections

A bonus "war story"

The aforementioned project did not proceed without problems.

Problem: the home page of colleague's advisor got clustered with:

- assorted pornography
- web pages on alcohol abuse

Problem II: our algorithm was provably correct, i.e., probability of failure was one in a million (we calculated it exactly).

What happened ?

- x a word (really, word's "signature", but ignore that)

- We used hash function $h(x) = (ax \bmod P) \bmod 2^8$
  - $P = 2^{64} - 57$ (more or less)
  - a randomly chosen

- For various reasons, x divisible by 8 always (we were sampling 1 in 8 words)

- Implementation bug: forgot to use long long int $\Rightarrow$ ax was computed modulo 2^{64} (rounding)

- mod P had almost always no effect, since $P \approx 2^{64}$

- x divisible by $8 \Rightarrow (ax)$ divisible by $8 \Rightarrow (ax) \bmod 2^8$ divisible by 8

- 3 lowest bits of $h(x)$ were almost always 0, so the *actual* range size was 2^5, not 2^8

- Enough for unexpected word collisions to occur...

Moral: do your hashing well, or you might never graduate.

References

- Linear permuations: Broder, Charikar, Frieze, Mitzenmacher, "Min-wise independent permutations", STOC'98. Available at
`http://www.cs.princeton.edu/~moses/papers/`

- Polynomial functions: Indyk, "A small approximately min-wise independent family of hash functions", SODA'99. Available at my web page.

- Locality Sensitive Hashing: Indyk,Motwani, "Approximate nearest neighbor: towards removing the curse of dimensionality", STOC'98, section 4.2. Note: "Near Neighbor" is called "PLEB" in that paper.

- Web clustering I: Broder et al, WWW6, 1997.

- Web clustering II: Haveliwala, Gionis, Indyk, "Scalable Techniques for Clustering the Web", WebDB 2000. Available at my web page.