Lecture 5 – Similarity Hashing

Hash function

$h : \mathcal{X} \to [1..N]$ (typically, sometimes $\mathbb{R}$)

→ useful as random number generator replacement
(instead of rand, rand, rand use hash(1), hash(2)...)

→ alternatively also Seed(x), rand()
(but this is very expensive)

Useful things to do with a hash fn

1) random number generator $\frac{h(x)}{N} \in [0,1]$
2) Rademacher hash Parity(h(x))
is $\{0,1\}$ or $\{\pm 1\}$ with prob. $1/2$
3) Basic requirement $\Pr_h\{h(x) = c\} = \frac{1}{N}$
k-wise ~~pairwise~~ indep $\Pr_h\{h(a_1) = c_1, \dots, h(a_k) = c_k\} = N^k$
for $\{a_i\}$ distinct
4) for simplicity (unrealistic) use ideal hash fn.

Permutations from hashes (aka Feistel hash)

$(x, y) \to (y, x \text{ XOR } h(y))$

repeat 3x to get crypt. strong hash

Luby - Rackoff Pseudorandom Permutation Generator

$\Rightarrow$ $x \to \text{Crypt}(x)$ is 1:1

Locality Sensitive Hash fnctn

map $x \to h(x)$ such that similarity between (x, x') is preserved.

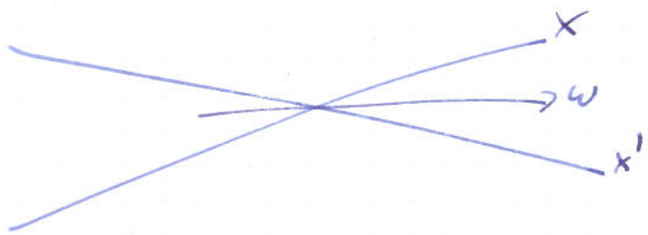
e.g. whp.

$$\left| \frac{d(h(x), h(x'))}{D(x, x')} - 1 \right| \le \varepsilon$$

not possible for all $D(x, x')$ but available e.g. for Euclidean metrics

example for later: Mx for $M_{ij} \sim N(0,1)$
(this is the Indyk & Motwani Example)

Similarity Estimation Techniques from Rounding Algorithms (M. Charikar '02)



1) draw $w_i \sim N(0, \sigma^2 \cdot \mathbb{1}_d)$
2) map $x \to \underbrace{\text{sgn}(\langle w_1, x\rangle \dots \langle w_n, x\rangle)}_{=: h(x)}$

Theorem: $\mathop{\mathbb{E}}_h \frac{\pi}{2n} \|h(x) - h(x')\|_1$

$= \arccos \langle x, x' \rangle$

for $\|x\| = \|x'\| = 1$

Lecture 5 - Similarity Hashing

Estimator for inner product:

$$\langle x, x' \rangle \approx \|x\| \cdot \|x'\| \cdot \frac{1}{2m} \|h(x) - h(x')\|$$

this is <u>much</u> cheaper:

- → modern CPUs can do 64 - 256 bit Hamming distance comp. in 1 clock cycle ($\frac{1}{2}$ ns)
- → only need to store 1 float ($\|x\|$), 64 bit hash regardless of the dimensionality of the data

<u>Theorem</u>:

$$\Pr\left\{ \left| \langle x, x' \rangle - \cos \frac{\pi}{2m} \|h(x) - h(x')\| \right| > \varepsilon \right\} \leq 2 \exp\left(- m \varepsilon^2 / \pi^2\right)$$

<u>Proof</u>: need McDiarmid inequality (like Hoeffding)

- → replace 1 hash function
- → argument doesn't change by more than $\frac{\pi}{m}$
- → cos doesn't change more either
- ↯ but $\mathbb{E} \cos \frac{\pi}{2m} \|h(x) - h(x')\| \neq \langle x, x' \rangle$
- → but by inverting the condition we can get it

Definition: LSH scheme with distr.:

$$\mathrm{Sim}(x, x') := \Pr_{h \in \mathcal{H}} \left\{ h(x) = h(x') \right\}$$

Lemma: $\mathrm{Sim}(x, x')$ is a kernel

Proof: $\delta(h(x), h(x'))$ is a kernel

Lemma: $d(x, x') := 1 - \mathrm{Sim}(x, x')$ satisfies the triangle inequality

Proof: For any given h we have

$$\left[1 - \delta(h(x), h(x'))\right] + \left[1 - \delta(h(x'), h(x''))\right] - \left[1 - \delta(h(x), h(x''))\right]$$

$$= 1 + \delta(h(x), h(x'')) - \delta(h(x), h(x')) - \delta(h(x'), h(x''))$$

≥ 0 Taking expectation over h

Example: Sim Hash (a la Charikar)

$$x \to \mathrm{sgn}(\langle \omega, x \rangle)$$

Example: Minwise Hash (Broder)

map set $\pi^{-1} \min(\pi(S)) =: h_\pi(S)$

⇒ selects a random element from S

$$\Pr_\pi \left\{ h_\pi(S) = h_\pi(S') \right\} = \frac{|S \cap S'|}{|S \cup S'|}$$

Lecture 5 – Similarity Hashing

Proof:

→ smallest element in $S \cup S'$ is with prob $\frac{|S \cap S'|}{|S \cup S'|}$ in $|S \cap S'|$

→ Prob that $h(S) = h(S')$ only if the element is in both sets ∎

Detour: Shingles (Broder et al.)

→ instead of picking single element pick k smallest entries

→ not quite correct but good enough

Better: Conditional Random Sampling (Li, Pi, Church et al.)
(use the hash ID, too)

Application: Duplicate detection on webpages

1) $(h_i(x), \text{docid}) \to$ sort by $h_i(x)$
2) count (docid, docid) pairs to find collision candidate
3) count (docid, docid) to confirm links

$\Rightarrow O(\text{duplicates}^2)$ for large clusters

Back to hash functions

→ we have general $\text{sim}(x, x') = \Pr_h\{h(x) = h(x')\}$

→ can we get a binary $h(x)$?

Define $\text{sim}'(x, x') := \Pr_{h,\delta}\{\delta(h(x)) = \delta(h(x'))\}$

$$= \underbrace{\frac{1}{2}}_{\text{random match for different hashes}} + \frac{1}{2} \underbrace{\delta(h(x), h(x'))}_{\text{sim}(x, x')}$$

Corollary: we can embed $\text{sim}(x, x')$ on cube

Application: Fast EM sampling (Ahmed, Pavlov, Smola)

The problem: in clustering we have to evaluate

$$p(y|x) \propto p(x|\theta_y) \cdot p(y|\theta)$$
$$= \exp(\langle \ell(x), \theta_y \rangle - g(\theta_y)) \cdot p(y|\theta)$$

Computing $\langle \ell(x), \theta_y \rangle$ is fatal:

$O(d \cdot k)$

d: 10^4 dims for bag of words

k: $10^4 - 10^5$ clusters

CPU cost is $\sim 10^9$ FLOPS for 1 count per sampling round. This is fatal ↯

Variant A) approximate by Sim hash and use Chernoff bound for rejection sampler

$\Rightarrow$ This is terrible since $\exp(\varepsilon)$ where $\varepsilon = O\left(\frac{1}{|d|}\right)$

Variant B) Metropolis-Hastings scheme

have $p(x)$ to sample

but can only afford $q(x'|x)$

$\Rightarrow$ accept with prob $\min\left(1, \frac{p(x')\, q(x|x')}{p(x)\, q(x'|x)}\right)$

$\Rightarrow$ Special case $q(x'|x) = q(x')$ to obtain

$$\min\left(1, \frac{p(x')\, q(x)}{p(x)\, q(x')}\right)$$

$\rightarrow$ So use approx $\underbrace{\|x\|\,\|\theta\|\,\|h(x) - h(\theta)\|}_{3 \text{ Flops}}$ as proposal $\Rightarrow$ $3 \cdot 10^5$ Flops per proposal

$\rightarrow$ good acceptance ratio $\Rightarrow$ fast mixing

Application: Similarity search

$\rightarrow$ hashing
$\rightarrow$ permute
$\rightarrow$ sort
$\rightarrow$ find neighbors

} repeat and find best match