Locality-sensitive hashing (LSH)

Hashing as "sketching"

Hashing as checksumming

- *Idea of checksum*: compute a (small) *fingerprint* $h(x)$ of a (large) object x such that small modifications of x alter $h(x)$

- *Goal*: verify the *integrity* of x

- *Examples of checksum algorithms*:
  - parity check, modsum
  - Luhn algorithm
  - cryptographic hash functions
  - …

Hashing as sketching

- *Idea (opposite to checksum)*: compute a (small) *sketch* (*signature*) $h(x)$ of a (large) object x such that small modifications of x (are likely to) preserve $h(x)$

- *Goal*: instead of comparing (large) objects, compare (small) sketches

Avoiding pairwise comparisons

- *Examples*:
  - Finding near-duplicates in a large set of documents
  - Clustering
  - Near-neighbor search (NNS)

- *General approach for NNS*
  - hash objects using LSH
  - hash the query, look at corresponding bucket
  - check objects of the bucket
  - ⇒ false negatives and false positives

Locality-sensitive hashing (LSH)

- given objects $\{x_1, \ldots, x_n\}$ (points in space of high dimension d)
- assume a distance *dist* and associated similarity relation $sim(x, y) \in [0,1]$ ($dist(x, y) = 0$ iff $sim(x, y) = 1$)
- define a family H of hash functions such that

$$\mathrm{P}_h[h(x) = h(y)] = sim(x, y)$$

- that is: hash collision captures object similarity

Locality-sensitive hashing (LSH)

- given objects $\{x_1, \ldots, x_n\}$ (points in space of high dimension d)
- assume a distance *dist* and associated similarity relation $sim(x, y) \in [0,1]$ ($dist(x, y) = 0$ iff $sim(x, y) = 1$)
- define a family H of hash functions such that

$$\mathrm{P}_h[h(x) = h(y)] = sim(x, y)$$

- that is: hash collision captures object similarity

- estimating similarity: sample hash functions $h_1, \ldots, h_k \in H$

$$\mathrm{E}[\#\{i \mid h_i(x) = h_i(y)\}]/k \;=\; sim(x, y)$$

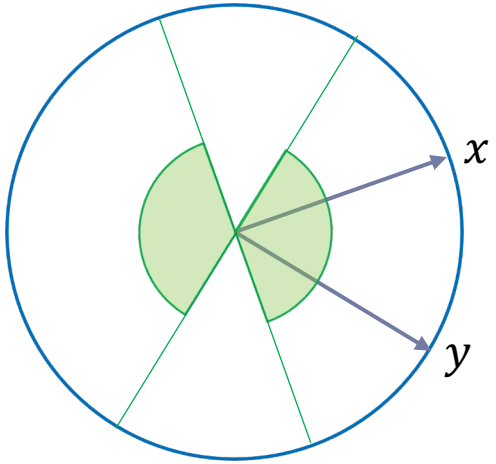
Example 1: Hamming distance

- objects: bitvectors $x \in \{0,1\}^d$
- Hamming distance: $H(x, y)$ is the nb of unequal corresponding bits, e.g. $H(00011,10001) = 2$
- Hamming similarity $sim_H(x, y) = 1 - H(x, y)/d$
- define $h_i(x) = x_i$ for random bit $i \in [1, d]$
- *Claim*: $\mathrm{P}[h_i(x) = h_i(y)] = sim_H(x, y)$ (prove)

- $\frac{1}{k} \cdot \mathrm{E}\left[H\left(\left(h_1(x), \ldots, h_k(x)\right), \left(h_1(y), \ldots, h_k(y)\right)\right)\right] = H(x, y)$

Example 2: Angular distance

- $x, y \in \mathbb{R}^n, \theta(x, y) = (\text{angle between } x \text{ and } y)/\pi$
- *Ex*: $\theta\left(\left(\frac{\sqrt{3}}{2}, \frac{1}{2}\right), \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)\right) = \frac{1}{12}$
- angular similarity: $1 - \theta(x, y)$
- for a random unit vector r, $h_r(x) = sign < x, r >$
- *Claim*: $\mathrm{P}[h_r(x) = h_r(y)] = 1 - \theta(x, y)$



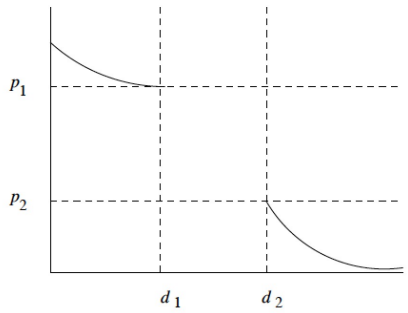
- $\frac{1}{k} \cdot \mathrm{E}\left[H\left(\left(h_1(x), \ldots, h_k(x)\right), \left(h_1(y), \ldots, h_k(y)\right)\right)\right] = \theta(x, y)$

Approximate Near-Neighbor (ANN)

- Hash objects of the dataset using chaining
- Given a query x, look through all objects y in bucket $h(x)$, compute the true distance $dist(x, y)$ and report those with $dist(x, y) < d$ (filtering)

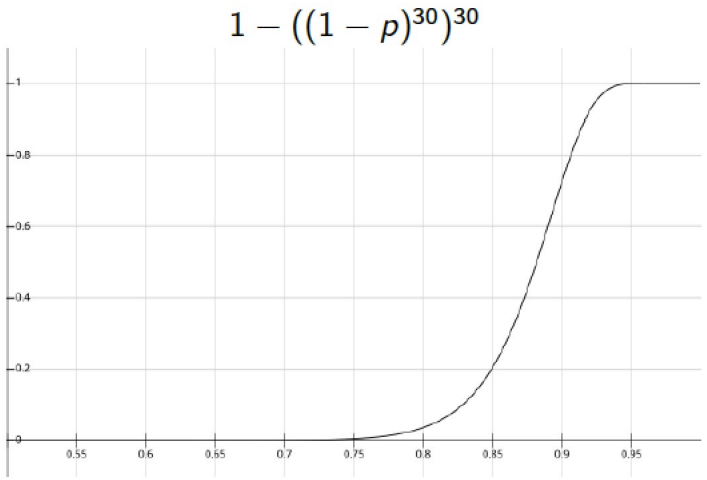
- MUCH faster than pairwise comparison
- false negatives

- (r, cr, P, p)-sensitive family of hash functions:
  - $dist(x, y) < r \Rightarrow \mathrm{P}[h(x) = h(y)] > P$
  - $dist(x, y) > cr \Rightarrow \mathrm{P}[h(x) = h(y)] < p$



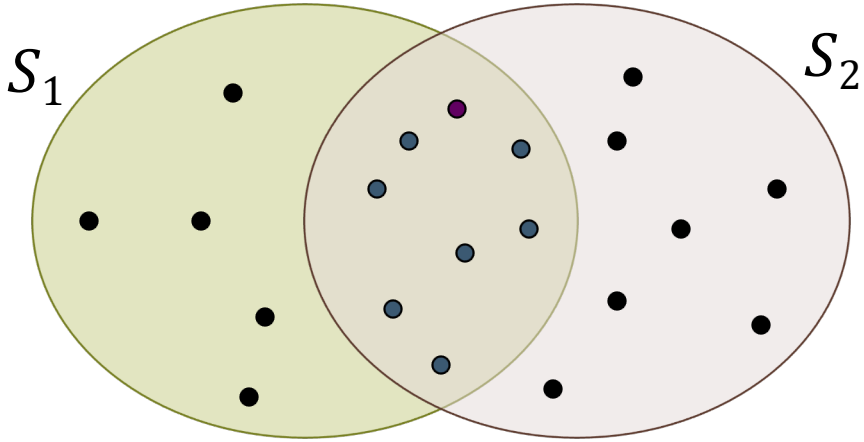
Gap amplification

- let $h(x) =< h_1(x), \ldots, h_k(x) >$ and h_i's belong to a (r, cr, P, p)-sensitive family
- by using L distinct hash tables (in OR fashion), we can construct a $\left(r, cr, 1-\left(1-P^k\right)^L, 1-(1-p^k)^L\right)$-sensitive family
- *Example:* using 4-tuple hash functions and 4 hash tables, a $(0.2,0.6,0.8,0.4)$-sensitive family turns to $(0.2,0.6,0.8785,0.0985)$-sensitive
- the construction can be cascaded to achieve arbitrary large gap



LSH on sets: Jaccard distance

- consider sets over a universe U
- Jaccard similarity $JS(S_1, S_2) = |S_1 \cap S_2| / |S_1 \cup S_2|$
- Jaccard distance $JD(S_1, S_2) = 1 - JS(S_1, S_2)$



- *Examples*:
  - similarity of customers w.r.t. purchased items
  - similarity of products w.r.t. customers who ordered them (Amazon, Netflix, …)

MinHash for Jaccard distance

- consider a random ordering $\pi: \mathcal{U} \rightarrow \{1, \ldots, |\mathcal{U}|\}$
- for a set $S \subseteq \mathcal{U}$, define $h(S) = \min_{x \in S}\{\pi(x)\}$
- then for two sets S_1, S_2, we have

$$P[h(S_1) = h(S_2)] = JS(S_1, S_2)$$

- Proof: …

MinHash signatures

- consider random orderings $\pi_1, \ldots, \pi_k : \mathcal{U} \to \{1, \ldots, |\mathcal{U}|\}$
- orderings are difficult to handle ⇒ replace them by (random) hash functions $\pi_1, \ldots, \pi_k : \mathcal{U} \to \{1, \ldots, N\}$ for a sufficiently large N
- for a set S, define its *signature* to be

$$sig(S) = < \min_{x \in S}\{\pi_1(x)\}, \ldots, \min_{x \in S}\{\pi_k(x)\} >$$

- then we have

$$\mathrm{E}[\#\{i | sig(S_1)_i = sig(S_2)_i\}]/k = JS(S_1, S_2)$$

- *variant*: $sig(S)$ is the set of k minimum values of $\{\pi(x) | x \in S\}$ for a single π
- then $\mathrm{E}[|sig(S_1) \cap sig(S_2)|]/k = JS(S_1, S_2)$

MinHash: toy example

$S_1 = \{0,3\}, S_2 = \{2\}, S_3 = \{1,3,4\}, S_4 = \{0,2,3\}$

$\pi_1(x) = (x + 1) \bmod 5$

$\pi_2(x) = (3x + 1) \bmod 5$

$sig(S_1) =< 1,0 >, sig(S_2) =< 3,2 >$

$sig(S_3) =< 0,0 >, sig(S_4) =< 1,0 >$

Then

$JS(S_1, S_4)$ estimated to 1 (true answer $2/3$)

$JS(S_1, S_3)$ estimated to $1/2$ (true answer $1/4$)

$JS(S_3, S_4)$ estimated to $1/2$ (true answer $1/5$)

$JS(S_1, S_2)$ estimated to 0 (true answer 0)

Locality-Sensitive Hashing for Earthquake Detection: A Case Study of Scaling Data-Driven Science

Kexin Rong*, Clara E. Yoon†, Karianne J. Bergen‡, Hashem Elezabi*,
Peter Bailis*, Philip Levis*, Gregory C. Beroza†

Stanford University

ABSTRACT

In this work, we report on a novel application of Locality Sensitive Hashing (LSH) to seismic data at scale. Based on the high waveform similarity between reoccurring earthquakes, our application identifies potential earthquakes by searching for similar time series segments via LSH. However, a straightforward implementation of this LSH-enabled application has difficulty scaling beyond 3 months of continuous time series data measured at a single seismic station. As a case study of a data-driven science workflow, we illustrate how domain knowledge can be incorporated into the workload to improve both the efficiency and result quality. We describe several end-to-end optimizations of the analysis pipeline from pre-processing to post-processing, which allow the application to scale to time series data measured at multiple seismic stations. Our optimizations enable an over 100× speedup in the end-to-end analysis pipeline. This improved scalability enabled seismologists to perform seismic analysis on more than ten years of continuous time series data from over ten seismic stations, and has directly enabled the discovery of 597 new earthquakes near the Diablo Canyon nuclear power plant in California and 6123 new earthquakes in New Zealand.

1 INTRODUCTION

Locality Sensitive Hashing (LSH) [29] is a well studied computational primitive for efficient nearest neighbor search in high-dimensional spaces. LSH hashes items into low-dimensional spaces such that similar items have a higher collision probability in the hash table. Successful LSH applications include entity resolution [64], genome sequence comparison [18], text and image search [41, 52], near duplicate detection [20, 46], and video identification [37].

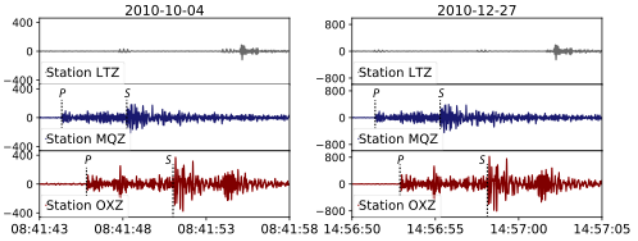


Figure 1: Example of near identical waveforms between occurrences of the same earthquake two months apart, observed at three seismic stations in New Zealand. The stations experience increased ground motions upon the arrivals of seismic waves (e.g., P and S waves). This paper scales LSH to over 30 billion data points and discovers 597 and 6123 new earthquakes near the Diablo Canyon nuclear power plant in California and in New Zealand, respectively.

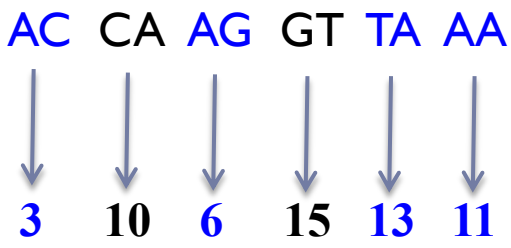
Second, despite large measurement volumes, only a small fraction of earthquake events are cataloged, or confirmed and hand-labeled by domain scientists. As earthquake magnitude (i.e., size) decreases, the frequency of earthquake events increases exponentially. Worldwide, major earthquakes (magnitude 7+) occur approximately once a month, while magnitude 2.0 and smaller earthquakes can occur several thousand times a day. At low magnitudes, it is increasingly difficult to detect earthquake signals because earthquake energy approaches the noise floor, and conventional seismological analyses can fail to disambiguate between signal and noise. Nevertheless, detecting these small earthquakes is important in uncovering unknown seismic sources [24, 32], improving the un-

MinHash for sequences (documents)

- General scenario:
  - represent a sequence (text) as a set of *k-mers* (*Q-grams*, *k-shingles*), i.e. tuples of consecutive letters (words) of fixed size
  - measure document similarity by Jaccard similarity and apply the MinHash framework
  - possibly do gap amplification
- first proposed by Broder (1997) with application to webpage similarity search

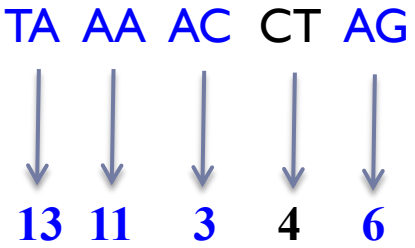
MinHash with m min values: example

ACAGTAAC



π

TAAACTAAG



MinHash={3,6,10}

$m = 3$

MinHash={3,4,6}

True Jaccard index: $4/(2+4+1)=4/7$

MinHash estimate: $2/3$

Ondov et al. Genome Biology (2016) 17:132
DOI 10.1186/s13059-016-0997-x

Genome Biology

SOFTWARE Open Access

Mash: fast genome and metagenome distance estimation using MinHash

Brian D. Ondov[1], Todd J. Treangen[1], Páll Melsted[2], Adam B. Mallonee[1], Nicholas H. Bergman[1], Sergey Koren[3] and Adam M. Phillippy[3*]

Abstract

Mash extends the MinHash dimensionality-reduction technique to include a pairwise mutation distance and *P* value significance test, enabling the efficient clustering and search of massive sequence collections. Mash reduces large sequences and sequence sets to small, representative sketches, from which global mutation distances can be rapidly estimated. We demonstrate several use cases, including the clustering of all 54,118 NCBI RefSeq genomes in 33 CPU h; real-time database search using assembled or unassembled Illumina, Pacific Biosciences, and Oxford Nanopore data; and the scalable clustering of hundreds of metagenomic samples by composition. Mash is freely released under a BSD license (https://github.com/marbl/mash).

Keywords: Comparative genomics, Genomic distance, Alignment, Sequencing, Nanopore, Metagenomics

Background

When BLAST was first published in 1990 [1], there were less than 50 million bases of nucleotide sequence in the public archives [2]; now a single sequencing instrument can produce over 1 trillion bases per run [3]. New methods are needed that can manage and help organize this scale of data. To address this, we consider the general problem of computing an approximate distance between two sequences and describe Mash, a general-purpose toolkit that utilizes the MinHash technique [4] to reduce large sequences (or sequence sets) to compressed sketch representations. Using only the sketches, which can be thousands of times smaller, the similarity of the original sequences can be rapidly estimated with bounded error. Importantly, the error of this computation depends only on the size of the sketch and is independent of the genome size. Thus, sketches comprising just a few hundred values can be used to approximate the similarity of arbitrarily large datasets. This has important applications for large-scale genomic data management and emerging long-read, single-molecule sequencing technologies. Potential applications include any problem where an approximate, global distance is acceptable, e.g. to triage and cluster sequence data, assign species labels, build large guide trees, identify mis-tracked samples, and search genomic databases.

The MinHash technique is a form of locality-sensitive hashing [5] that has been widely used for the detection of near-duplicate Web pages and images [6, 7], but has seen limited use in genomics despite initial applications over ten years ago [8]. More recently, MinHash has been applied to the relevant problems of genome assembly [9], 16S rDNA gene clustering [10, 11], and metagenomic sequence clustering [12]. Because of the extremely low memory and CPU requirements of this probabilistic approach, MinHash is well suited for data-intensive problems in genomics. To facilitate this, we have developed Mash for the flexible construction, manipulation, and comparison of MinHash sketches from genomic data. We build upon past applications of MinHash by deriving a new significance test to differentiate chance matches when searching a database, and derive a new distance metric, the Mash distance, which estimates the mutation rate between two sequences directly from their MinHash sketches. Similar "alignment-free" methods have a long history in bioinformatics [13, 14]. However, prior methods based on word counts have relied on short words of only a few nucleotides, which lack the power to differentiate between closely related sequences and produce distance

* Correspondence: adam.phillippy@nih.gov
[3]Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, MD, USA
Full list of author information is available at the end of the article

Examples

- 50K-byte documents (50K chars)
  - let $k = 4$
  - consider sketches of 1K bytes (250 32-bit values)
  - then the Jaccard distance between two documents will typically be estimated within a few %

- bacterial genomes ~5M chars (1.25 Mb)
  - let $k = 16$, signature size = 400
  - 400 32-bit values (1.6Kb) are sufficient to discriminate microbial genomes

Features of MinHash

- Simple! easy to compute!
- Easy to maintain for growing datasets
- Size (m) does not depend on the size of the dataset
- Suitable for comparing datasets of (roughly) the same size
- Low level of false positives ("accidental similarity")
- Very space-efficient, 4-5 orders of magnitude compression

Recommended book

J.Leskovec, A.Rajaraman, J.Ullman, *Mining Massive Datasets*, Cambridge University Press, 2020 (3rd edition)

http://www.mmds.org/