Freie Universität Berlin

MinHashing and sketching

Nano course probabilistic counting and sketching

Knut Reinert

Summer 2025

Introduction

Today

- We discuss how we can store sets of sets of k-mers in discrete or probabilistic data structures.
- What operations can be supported in what space and time.

Literature

- Nicola Prezza: Skript **CM0622 - Algorithms for Massive Data**, University of Venice (arxiv)
- Rayan Chiki, Jan Holub, and Paul Medvedev: **Data structures to represent sets of k-long DNA sequences**.
- Brian D. Ondov , Todd J. Treangen , Páll Melsted, Adam B. Mallonee , Nicholas H. Bergman, Sergey Koren and Adam M. Phillippy: **Mash: fast genome and metagenome distance estimation using MinHash**
- David Koslicki, Hooman Zabeti: **Improving MinHash via the containment index with applications to metagenomic analysis**

Introduction

Sketching

Let x be some data: a set, a string, an integer, etc. A **data sketch** is the output of a randomized function f (in general, the combination of a certain number of hash functions) mapping x to a sequence of bits $f(x)$ with properties 1-3 below, plus (depending on the application) also property 4:

1. The bit-size of $f(x)$ is much smaller than the bit-size of x (usually, sub-linear or even poly-logarithmic).
2. $f(x)$ can be used to compute (efficiently) some properties of x. For example, if x is a multiset then $f(x)$ could be used to compute an approximation of the number of distinct elements contained in x, or the most frequent element in x.

Introduction

Sketching

3. $f(x)$ can be updated (efficiently) if x gets updated. Importantly, it should be possible to update $f(x)$ without knowing x. For example:
   - if we add an element y to a set x, it should be possible to compute $f(x \cup \{y\})$ knowing just $f(x)$ and y (not x).
   - More general, given two sketches $f(x_1)$ and $f(x_2)$, it should be possible to compute the sketch of the composition of x_1 and x_2 (under some operator). For example, if x_1 and x_2 are sets we could be interested in obtaining the sketch of $f(x_1 \cup x_2)$, without knowing x_1 and x_2.
4. If x and y are similar according to some measure of similarity (e.g. Euclidean distance), then $f(x)$ and $f(y)$ are likely to be similar (according to some measure of similarity, not necessarily the same as before).

MinHashing

MinHash is a sketching algorithm used to estimate the similarity of sets. It was invented by Andrei Broder in 1997 and initially used in the AltaVista search engine to detect duplicate web pages and eliminate them from search results. Here we report just a definition and analysis of MinHash. MinHash is a technique for estimating the Jaccard similarity (or index) $J(A, B)$ of two sets A and B:

Definition (Jaccard similarity)

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

Without loss of generality, we may assume that we work with sets of integers from the universe $[1, n]$.

MinHashing

Definition (Minhash function)

Let h be a hash function. The MinHash hash function of a set A is defined as $h'(A) = \min\{h(x) : x \in A\}$, i.e. it is the minimum of h over all elements of A.

Definition (Minhash estimator)

Let $J'_h(A, B)$ be the indicator RV. defined as follows:
$J'_h(A, B) = 1$, if $h'(A) = h'(B)$, $J'_h(A, B) = 0$ otherwise.
Note that $J'_h(A, B)$ is a Bernoullian RV. We prove the following remarkable property:

Lemma

If $h : [1, n] \to [1, n]$ is a uniform permutation, then $E[J'_h(A, B)] = J(A, B)$

MinHashing

Proof

Let $|A \cup B| = N$. For $i \in A \cup B$, consider the event $smallest(i) = (\forall j \in A \cup B - \{i\})(h(i) < h(j))$, stating that i is the element of $A \cup B$ mapped to the smallest hash $h(i)$ (among all elements of $A \cup B$).

Since h is a permutation, exactly one element from $A \cup B$ will be mapped to the smallest hash (i.e. $smallest(i)$ is true for exactly one $i \in A \cup B$), so $\{smallest(i)\}_{i \in A \cup B}$ is a partition of cardinality $N = |A \cup B|$ of the event space. Moreover, the fact that h is completely uniform implies that $P(smallest(i)) = P(smallest(j)) = 1/N$ for all $i, j \in A \cup B$.

Now, if $smallest(i)$ is true and $i \in A \cap B$, then $J'_h(A, B) = 1$, if i is not in the intersection, then the estimate is 0. It is then easy to write down the proof (exercise).

MinHashing

Reducing the variance

The RV $J'_h(A, B)$ is not a good estimator since it is a Bernoullian RV and thus h has a large variance: in the worst case ($J(A, B) = 0.5$), we have $Var[J'_h(A, B)] = 0.25$ and thus h the expected error (standard deviation) of $J'_h(A, B)$ is $\sqrt{Var[J'_h(A, B)]} = 0.5$.

This means that on expectation we are off by 50% from the true value of $J(A, B)$. We know how to solve this issue: just take the average of k independent such estimators, for sufficiently large k.

MinHashing

Reducing the variance

Let $h_i : [1, n] \to [1, n]$, with $i = 1, \ldots, k$, be k independent uniform permutations. We define the MinHash sketch of a set A to be the k-tuple: $h_{min}(A) = (h'_1(A), h'_2(A), \ldots, h'_k(A))$. Then, we estimate $J(A, B)$ using the following estimator:

Improved MinHash estimator

$$J^+(A, B) = \frac{1}{k} \sum_{i=1}^{k} J'_{h_i}(A, B)$$

Note that the improved MinHash estimator for a union of two sets can be computed in $O(k)$ time given the MinHash sketches of two sets.

Min Hashing

Concentration bound

We can immediately apply the double-sided additive Chernoff-Hoeffding bound and obtain that $P(|J^+(A,B) - J(A,B)| \geq \epsilon) \leq 2e^{-\epsilon^2 k/2}$ for any desired absolute error $0 < \epsilon \leq 1$. Fix now any desired failure probability $0 < \delta \leq 1$. By solving $2e^{-\epsilon^2 k/2} = \delta$ we obtain $k = 2\ln(2/\delta)/\epsilon^2$. We can finally state:

Theorem

Fix any desired absolute error $0 < \epsilon \leq 1$ and failure probability $0 < \delta \leq 1$. By using $k = 2\ln(2/\delta) \in O(\log(1/\delta))$ hash functions, the estimator $J^+(A,B)$ exceeds absolute error ϵ with probability of at most δ

$$P(|J^+(A,B) - J(A,B)| \geq \epsilon) \leq \delta$$

MinHashing

To summarize, we can squeeze down any subset of $[1, n]$ to a MinHash sketch of $O(\frac{\log(1/\delta)}{\epsilon^2})$ bits so that, later, in $O(\frac{\log(1/\delta)}{\epsilon^2})$ time we can estimate the Jaccard similarity between any pair of sets (represented with MinHash sketches) with arbitrarily small absolute error ϵ and arbitrarily small failure probability δ.

Note that it is easy to combine the MinHash sketches of two sets A and B so to obtain the MinHash sketch of $A \cup B$ (similarly, to compute the MinHash sketch of $A \cup \{x\}$ given the MinHash sketch of A):

$$h_{min}(A \cup B) = (\min\{h'_1(A), h'_1(B)\}, \ldots, \min\{h'_k(A), h'_k(B)\})$$

Fractional MinHashing

Irber introduced in 2020 the concept of fractional MinHashing. While simple, it has the advantage that you can derive an unbiased estimate of the containment index and hence the Jaccard index (can be computed from the containment index).

Definition

Given two arbitrary sets A and B, which are subsets of a domain Ω, the containment index $C(A, B)$ is defined as $C(A, B) := \frac{|A \cap B|}{|A|}$. Let h be a perfect hash function $h : \Omega \to [0, H]$ for some $H \in \mathbb{R}$. For a scale factor s, where $0 \leq s \leq 1$, a FracMinHash sketch of a set A is defined as follows:

$$FRAC_s(A) = \{h(a) \mid a \in A \text{ and } h(a) \leq Hs\}.$$

Fractional MinHashing

FracMinHash estimate

The scale factor s is an easily tunable parameter that can modify the size of the sketch. Using this FracMinHash sketch, we define the FracMinHash estimate of the containment index $\hat{C}_{\text{frac}}(A, B)$ as follows:

$$\hat{C}_{\text{frac}}(A, B) := \frac{|FRAC_s(A) \cap FRAC_s(B)|}{|FRAC_s(A)|}$$

Then Irber showed:

Theorem

For $0 < s < 1$, if A and B are two nonempty sets such that $A \backslash B$ and $A \cap B$ are nonempty, the following holds:

$$E\left[\hat{C}_{\text{frac}}(A, B)\mathbb{1}_{|FRAC_s(A)|>0}\right] = \frac{|A \cap B|}{|A|}\left(1 - (1-s)^{|A|}\right).$$

Fractional MinHashing

In light of the Theorem, we note that $\hat{C}_{\text{frac}}(A,B)$ is not an unbiased estimate of $C(A,B)$: The expected value of $\hat{C}_{\text{frac}}(A,B)$ is not equal to $C(A,B)$. However, for sufficiently large $|A|$ and s, the bias factor ($1-(1-s)^{|A|}$) is sufficiently close to one.

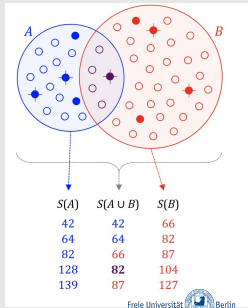
Alternatively, if $|A|$ is known (or estimated, e.g., by using HyperLogLog or the estimate in Sec. A.6) (Flajolet et al. 2007), then

$$C_{\text{frac}}(A,B) := \frac{|FRAC_s(A) \cap FRAC_s(B)|}{|FRAC_s(A)|\,(1-(1-s)^{|A|})} \mathbb{1}_{|FRAC_s(A)|>0}$$

is an unbiased estimate of the containment index $C(A,B)$.

Application MASH

After introducing the general principles the rest of the lecture follows the publications about Mash and the containment index. Mind that this is an active field and while Mash was a seminal paper since then other methods have emerged. Mash provides two basic functions for sequence comparisons: **sketch** and **dist**. The sketch function converts a sequence or collection of sequences into a MinHash sketch (see next Figure). Mind, that one can use a family of hash functions in general, while Mash uses only 1.



Application MASH

Reminder Jaccard

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \approx \frac{|S(A \cup B) \cap S(A) \cap S(B)|}{|S(A \cup B)|}$$

Fig. 1 Overview of the MinHash bottom sketch strategy for estimating the Jaccard index. First, the sequences of two datasets are decomposed into their constituent k-mers (*top*, *blue and red*) and each k-mer is passed through a hash function *h* to obtain a 32- or 64-bit hash, depending on the input k-mer size. The resulting hash sets, *A* and *B*, contain |*A*| and |*B*| distinct hashes each (*small circles*). The Jaccard index is simply the fraction of shared hashes (*purple*) out of all distinct hashes in *A* and *B*. This can be approximated by considering a much smaller random sample from the union of *A* and *B*. MinHash sketches *S*(*A*) and *S*(*B*) of size $s = 5$ are shown for *A* and *B*, comprising the five smallest hash values for each (*filled circles*). Merging *S*(*A*) and *S*(*B*) to recover the five smallest hash values overall for *A*∪*B* (*crossed circles*) yields *S*(*A*∪*B*). Because *S*(*A*∪*B*) is a random sample of *A*∪*B*, the fraction of elements in *S*(*A*∪*B*) that are shared by both *S*(*A*) and *S*(*B*) is an unbiased estimate of *J*(*A*,*B*)

Application MASH

The **dist** function compares two sketches and returns an estimate of the **Jaccard index** (i.e. the fraction of shared k-mers), a P value, and the Mash distance, which estimates the rate of sequence mutation under a simple evolutionary model.

In more detail:

For a sketch size s and genome size n, a sketch can be efficiently computed in $O(n \log s)$ time by maintaining a sorted list of size s and updating the current sketch only when a new hash is smaller than the current sketch maximum.
Further, the probability that the i-th hash of the genome will enter the sketch is s/i, so the expected runtime of the algorithm is $O(n + s \log s \log n)$, which becomes nearly linear when $n >> s$.

Application MASH

Mash follows Broder's original formulation and merge-sorts two sketches $S(A)$ and $S(B)$ to estimate the Jaccard index. The merge is terminated after s unique hashes have been processed (or both sketches exhausted), and the Jaccard estimate is computed as $j = \frac{x}{s'}$ for x shared hashes found after processing S' hashes.

Because the sketches are stored in sorted order, this requires only $O(s)$ time and effectively computes:

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

which is an unbiased estimate of the true Jaccard index.

Application MASH

The Jaccard index J is a useful measure of global sequence similarity because it correlates with ANI (Average Nucleotide Identity), a common measure of global sequence similarity. However J is sensitive to genome size and simultaneously captures both point mutations and gene content differences.

Using Chernoff bounds, one can show that the probability of MinHash for deviating from the true value grows exponentially as the size of the true Jaccard value decreases to zero. An alternative is the so-called **containment index**.

Containment index

Consider the case of estimating the Jaccard index between two sets A and B of very different size. The traditional minhash randomly samples from the union $A \cup B$ and uses the number of sampled points that fall in $A \cap B$ to estimate the Jaccard index. With more sampled elements falling in $A \cap B$, the more accurate the Jaccard estimate will be.

Part A) of the next Figure demonstrates the case of sampling 100 random points from $A \cup B$ leading to 3 points lying in $A \cap B$. In the containment min hash approach, we randomly sample elements only from the smaller set (in this case, A) test if this element is in B (and hence in $A \cap B$). This is used to estimate the containment index, which is then used to estimate the Jaccard index itself.

Containment index

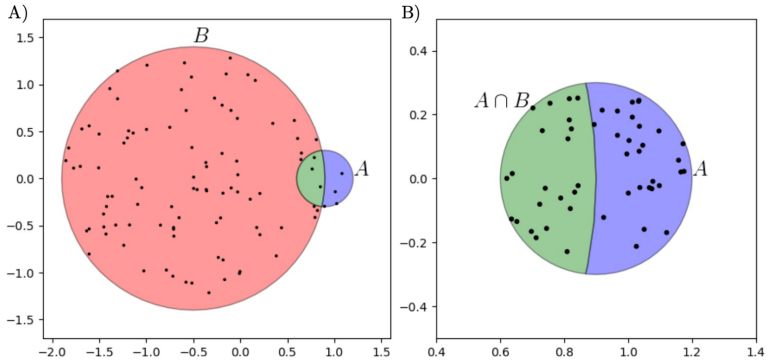


FIGURE 1. Conceptual comparison of classical min hash to the proposed containment approach when estimating the Jaccard index of very different sized sets. A) Sampling 100 points from $A \cup B$ (as is done in the classical min hash approach) leads to finding only 3 elements in $A \cap B$. B) Sampling just 50 points of A and testing if a point $x \in A \cap B$, finds 22 elements in $A \cap B$. This latter approach will be seen to lead to a better estimate of the Jaccard index.

Containment index

Analysis

Let us look at the analysis. For this, we will use the classic Chernoff bounds in its multiplicative form to estimate the probability of relative error in the probabilistic methods we consider:

Theorem

Suppose $X_1, X_2, \ldots, X_n$ are independent, identically distributed Bernoulli random variables and let $X = \sum X_i$ and $\mu = E(X)$. Then it holds:

$$P(X \leq (1-\delta)\mu \leq e^{\frac{-\delta^2\mu}{3}}$$

and

$$P(X \leq (1+\delta)\mu \leq e^{\frac{-\delta^2\mu}{3}}.$$

Hence

$$P(|\frac{X-\mu}{\mu}| \geq \delta) \leq 2 \cdot e^{\frac{-\delta^2\mu}{3}}.$$

Containment index

Analysis

Given two non-empty sets A and B, we wish to estimate $J(A, B)$. Fix $k \in N$ and select a family of (min-wise independent) hash functions $\{h_1, \ldots, h_k\}$ of $k \in N$ each with domain containing $A \cup B$.
For a set S in the domain of the hash functions, define $h_i^{min}(S) = argmin_{s \in S} h_i(s)$ as an element of S that causes h to achieve its minimum value on S. Given appropriate hash functions (or an order on S), this minimum is unique. Define the random variables $X_i = 1$ if $h_i^{min}(A) = h_i^{min}(B)$, and $X_i = 0$ otherwise.

The probability of a collision (that is $h_i^{min}(A) = h_i^{min}(B)$ and hence $X_i = 1$) is equal to the Jaccard index of A and B and hence the expectation is given by

$$E(X_i) = \frac{|A \cap B|}{|A \cup B|} = J(A, B)$$

Containment index

Analysis

Thus for $X^k = \sum X_i$ the expectation is given by $E(X^k) = k \cdot J(A, B)$ and hence $J_{est} = \frac{X^k}{k}$ is used as an estimate for the Jaccard index.
Note that in practice, a single hash function h is commonly used and the elements hashing to the smallest k values are used in place of the h_i.

Applying the two-sided Chernoff bounds from the equation to the classic min hash approach yields:

$$P\left(\left|\frac{\frac{X^k}{k} - J(A,B)}{J(A,B)}\right| \geq \delta\right) \leq 2e^{\delta^2 k J(A,B)/3}$$

Thus, two quantities control the accuracy of this method for δ fixed: k and $J(A, B)$. We can now set a threshold t for the probability of deviation from the Chernoff bounds and derive then the required k for achieving this threshold (exercise).

Containment index

Analysis

For the containment index, a similar and much more favorable estimate than for standard MinHashing can be derived.
The containment min hash approach the authors propose differs from the classic min hash in that the family of k hash functions $\{h_1, \ldots, h_k\}$ (or alternatively k minimal values of the same hash function) have as domain A and the method randomly samples from A instead of $A \cup B$. This results in estimating the containment index $C = C(A, B)$ instead of the Jaccard index $J = J(A, B)$, but this can be done as follows:

$$J_Y^{est} = \frac{|A| C_{est}}{|A| + |B| - |A| C_{est}}$$

Containment index

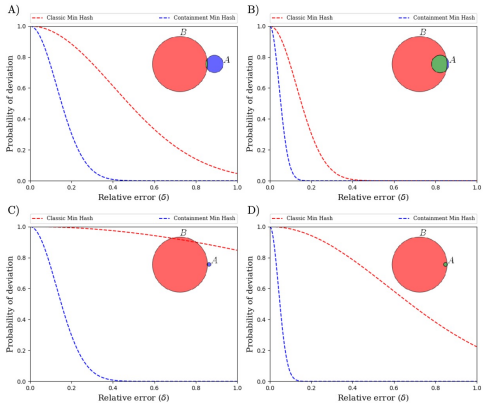


FIGURE 2. Comparison of the probability of deviation (red line: equation (2.5), blue line: Proposition (2.2)) versus the relative error δ with a fixed number of hash functions $(1,000)$. Relative sizes and Jaccard indexes of the sets in A)-D) are overlain as colored discs. Throughout, the containment min hash estimate of the Jaccard index has a much lower probability of error than the classical min hash approach. Note that in part D), there is approximately an 80% chance to have a deviation greater than 0.4 in the classic approach, whereas in the containment approach, the chance of having a deviation of greater than just 0.2 is almost zero.

Containment index

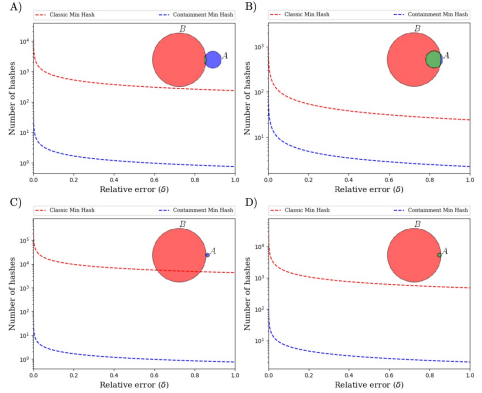


FIGURE 3. Comparison of the number of required hashes and relative error δ with probability of deviation $\leq 1\% := t$. The relative sizes and Jaccard indexes of the sets in A)-D) are overlain pictorially as colored discs. In all cases, the containment min hash estimate of the Jaccard index uses significantly fewer hash functions. For example, in part D), the classic min hash method (red line) needs $\approx 35,339$ hash functions to have less than 1% chance to have greater than 1% relative deviation in its estimate, whereas with the same thresholds, the containment min hash estimate of the Jaccard index (blue line) needs only ≈ 152 hash functions.

Containment index

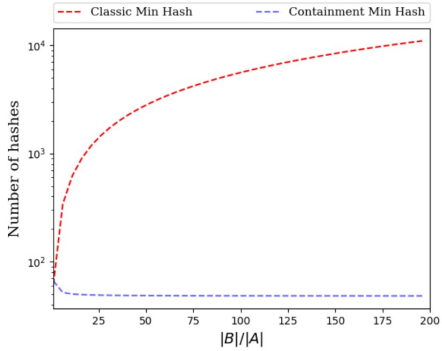


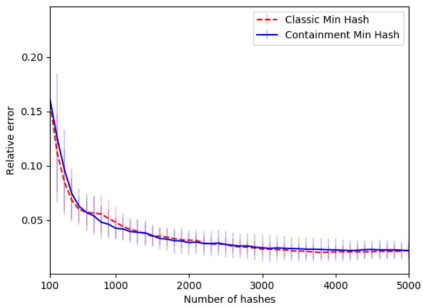
FIGURE 4. Comparison of the number of hash functions required by each method as a function of the relative sizes of the sets under consideration. The relative error $\delta = 0.1$ and the probability of deviation $t = 0.01$ are fixed. Increasing $\frac{|B|}{|A|}$, the number of required hash functions required for the classic min hash approach is increasing (red line). However, for the containment approach, the number of hash functions is nearly constant for $1 \leq \frac{|B|}{|A|} \leq 200$ (blue line) which is in agreement with equation 3.4.

Containment index

For the first set of simulated data, the authors used GemSIM to simulate $10K$ reads (of length 100) from 20 randomly selected bacterial genomes G_i (considered here as the set of all k-mers in the genome). They fixed the k-mer size to $k = 11$.

They then formed a set M_1 of all 11-mers in all reads in the simulated metagenome and repeated this 16 times. The second set of simulated data was produced similarly, except for the fact that they used $1M$ reads of the same length as before and formed a set M_2 of all 11-mers in all reads of the metagenome. Again, they then repeated this 16 times. Then they estimated $J(M_1, G_i)$ (the sets are roughly the same size) and $J(M_2, G_i)$ (the sets are very different size) using the standard MinHash and the Containment approach.

Containment index



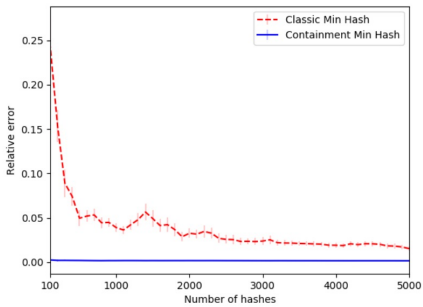


FIGURE 6. Comparison of the relative error of the containment min hash approach to the classical min hash estimate of the Jaccard index on simulated biological data. a) On 16 replicates of samples consisting of 20 genomes G_i with only 10K reads, showing the similarity of the methods in estimating $J(M_1, G_i)$ when the sets to be compared are roughly the same size. b) On 16 replicates of samples consisting of 20 genomes G_i with 1M reads, demonstrating the improvement of the containment approach in estimating $J(M_2, G_i)$ when the sets are of significantly different size.