Algorithms for Massive Data

Project

Davide Vitagliano 904120

Academic Year 2024/2025

1. Project Overview

This project implements an alignment-free method for estimating evolutionary distances between genomic sequences and reconstructing phylogenetic trees. It utilizes **Fractional MinHash (FracMinHash)** for scalable sketching and **Bloom Filters** for compact storage of these sketches. The tool supports computing distance matrices and building trees using **UPGMA** and **Neighbor-Joining (NJ)** algorithms.

The implementation effectively combines **Fractional MinHash** (for consistent, scalable subsetting of genomic data) with **Bloom Filters** (for compressed storage and fast set operations). By estimating set cardinalities directly from the bit vectors, it avoids storing full 64-bit hash values, significantly reducing memory usage while maintaining accuracy for distance estimation.

2. Data Structures

2.1 Bloom Filter (BloomFilter.h/cpp)

A standard Bloom Filter is used as the underlying storage mechanism for the sketches to maintain memory efficiency.

- **Structure**: Uses a bit array (std::vector<uint64_t>) to store set membership information in O(m) bits.
- **Hashing**: Implements a double-hashing simulation. A single 64-bit hash (from the input item) is mixed with an index *i (0 to h-1)* using a MurmurHash3 finalizer to generate h distinct bit positions.
- **Operations**:
  - add(item): Sets h bits corresponding to the item in O(h) time.
  - contains(item): Checks if all h bits are set (not explicitly used in FracMinHash logic but provided) in O(h) time.
  - merge(other): Performs a bitwise OR to merge two filters (useful for union estimation) in O($\frac{m}{64}$) time.

2.2 Fractional MinHash (FracMinHash.h/cpp)

This is the core sketching data structure. Unlike standard MinHash, which selects a fixed number of smallest hashes (bottom-k), FracMinHash selects a fixed fraction of the hash space.

- **K-mer Representation**:
  - **Rolling Hash**: DNA sequences are processed using a rolling window of length k.
  - **Encoding**: Bases (ACGT) are encoded into 2 bits each. The k-mer is represented as a packed 64-bit integer.
  - **Canonicalization**: The integer value of the k-mer and its reverse complement are compared, and the smaller unique value is used (canonical k-mer).
- **Hash Scrambling**:

    - The canonical k-mer integer is “scrambled” using a reversible pseudo-random permutation (SplitMix64 variant) and a seed. This ensures that k-mer values are uniformly distributed over [0, 2^64-1].
- **Storage**:
    - Hybrid approach: Instead of storing the sampled hash values in a std::set or std::vector, the sampled hashes are inserted directly into a **Bloom Filter**.
    - Scale Factor (s): A scale factor determines the sampling rate. A threshold $T = \frac{2^{64}-1}{s}$ is defined. Only hashes $h < T$ are inserted into the Bloom Filter.

3. Algorithms

3.1 Sketching Algorithm

- **Stream Processing**: The input genome sequence is read as a stream.
- **K-mer Extraction**:
    - A rolling window updates the current k-mer hash values (forward and reverse complement) in constant time O(1) per base.
    - Invalid characters (non-ACGT) reset the window.
- **Sampling**:
    - For each valid k-mer x, compute randomized hash $H(x)$.
    - **Fractional Condition**: If $H(x) < T$, the k-mer is selected.
- **Insertion**: Selected k-mers are added to the internal Bloom Filter.
    - **Worst case time:** O($n * h$). This occurs if every k-mer in the genome satisfies $H(x) < T$ (e.g., if s=1), requiring a Bloom Filter insertion (O(h) at every step.
    - **Expected time:** O($n * (1 + s * h)$). Since the hash function is uniform, only a fraction s of k-mers will trigger the insertion operation. With typical values of s (e.g., 0.001), the term $s * h$ is negligible, making the operation effectively O(n).

3.2 Distance Estimation (Jaccard)

The Jaccard Index $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ is estimated using the properties of Bloom Filters and the inclusion-exclusion principle.

- **Cardinality Estimation**: The number of distinct items n^* stored in a Bloom Filter is estimated using the formula:

$$n^* = -\frac{m}{k}\ln\left(1 - \frac{X}{m}\right)$$

  Where:
    - m: Total number of bits in the filter.
    - k: Number of hash functions.
    - X: Number of set bits (population count).
- **Set Operations**:
    - $|A|$: Estimated from Bloom Filter of sketch A.
    - $|B|$: Estimated from Bloom Filter of sketch B.

    - $|A \cup B|$: Estimated from the bitwise OR of Bloom Filters A and B.
- **Intersection Estimation**: derived as $|A \cap B| = |A| + |B| - |A \cup B|$.
- **Jaccard Index**: $J = \frac{|A \cap B|}{|A \cup B|}$.
- **Distance**: $D = 1 - J$.

3.3 Phylogeny Reconstruction (phylogenerator.h)

The tool implements two classic distance-based phylogenetic tree construction algorithms:

- **UPGMA (Unweighted Pair Group Method with Arithmetic Mean)**:
    - Hierarchical clustering method.
    - Iteratively merges the pair of clusters with the smallest distance.
    - Assumes a constant rate of evolution (molecular clock).
    - Outputs a rooted tree.
- **Neighbor-Joining (NJ)**:
    - Bottom-up clustering method.
    - Calculates a Q-matrix to adjust distances based on divergence from the rest of the tree.
    - Does not assume a constant molecular clock.
    - Produces an unrooted tree (though represented as rooted in the output format).

4. Usage Flow

- **--create-sketch**:
    - Reads sequence from Stdin.
    - Computes optimal Bloom Filter parameters (m, k_{hashes}) based on genome size and scale.
    - Generates the FracMinHash sketch stored in a Bloom Filter.
    - Saves the sketch (metadata + bit array) to a binary file.
- **--distance**:
    - Loads multiple sketch files.
    - Computes pairwise distances $(1 - Jaccard)$ between all inputs.
    - Outputs a distance matrix in Phylip format.
    - Constructs and outputs UPGMA and NJ trees in Newick format.

5. Experiments

To test the tool's correctness, experiments were conducted on both synthetic and real-world genomic data.

A unit test validated the core logic by sketching three short, hardcoded DNA sequences with known edit distances, verifying that the estimated Jaccard distances matched expected theoretical values. For real-world validation, the tool processed streaming genomic data for five diverse species (Homo sapiens, Mus musculus, Loxodonta africana, Drosophila melanogaster, and Saccharomyces cerevisiae).

These experiments confirmed the system's ability to handle varying genome sizes using a consistent Bloom Filter size and scaling factor, successfully generating a distance matrix and reconstructing phylogenetic trees (UPGMA and Neighbor-Joining) that reflect expected biological relationships.

After running the tool with the above mentioned species, fixing the parameters:

- k equal to 13
- scale equal to 0.00001
- seed equal to 1469598103934665603

It resulted in bitvectors of length 306732 bits and 7 hash velues for the bits to be flipped. The resulting memory usage on disk is 37 KB per sketch, because the number of retained hash values in the sketch are ~70-300, depending on the length of the genome.

The computation time to create each sketch is ~60 seconds for the longer genomes (e.g. Homo sapiens), and ~2 seconds for the shorter genomes (e.g. Saccharomyces cerevisiae). While the computation time to build the distance matrix is ~0.0015 seconds.

If the scale is increased by a factor of 10 (0.0001) the number of retained hashes increase to ~800-3100, but also the sketch dimension increases to 374 KB. The computation time for each sketch doesn't degrade. Finally, the estimated distances in the matrix differ only by ~0.01, thus it is useless to use much more memory for the sketches without improving the accuracy of the estimates.

The tool works with some optional variables passed as command arguments for the *--create-sketch* command:

- **--k:** the k-mer size, it overrides *--d-max* if both are provided. It defaults to 13.
- **--d_max:** max evolutionary distance. It is used to calculate k if *--k* is not set.
- **--scale:** the sampling rate for the Fractional MinHash. It defaults to 0.0001.
- **--seed:** the seed used in the hash function scrambling. It defaults to (1469598103934665603).
- **--genome-size:** the approximate size in bytes of the largest genome to be computed. It defaults to the human genome of 3200000000 bytes.

6. Bibliograhpy

1. MurmurHash https://en.wikipedia.org/wiki/MurmurHash
2. Lightweight compositional analysis of metagenomes with FracMinHash and minimum metagenome covers https://www.biorxiv.org/content/10.1101/2022.01.11.475838v2
3. Estimating similarity and distance using FracMinHash https://pmc.ncbi.nlm.nih.gov/articles/PMC12082993/