CSCI 1820/2820: An overview
Spring 2022

• Ch. 1 BLAST Algorithm and Karlin-Altschul Statistics
• Ch. 2 Genome Assembly and Haplotype Assembly Algorithms
• Ch. 3 Hidden Markov Models (HMM) Algorithms: The Learning Problem
• Ch. 4 Recombination and Ancestral Recombination Graphs (ARGs)
• Ch. 5 Rigorous Clustering and Spectral Clustering Algorithms
• Ch. 6 Algorithms for Constructing Suffix Trees in Linear Time
• Ch. 7 Protein Folding Algorithms (Introduction)
Ch. 1: BLAST Algorithm

Questions: When a DNA sequence or protein sequence is a biological sequence? How can we computationally identify them?

Examples of problems we need to solve along the way:

Problem 1. General scoring schemes – and the max scoring subsequence

Problem 2. The Gambler’s Ruin/Random Walks
The BLAST Algorithm

Authors
- Stephen Altschul
- Warren Gish
- Webb Miller
- Eugene W. Myers
- David Lipman

“Basic Local Alignment Search Tool”

Journal of Molecular Biology (1990) 215, 403-410
Karlin Altschul Equation

$$E = kmNe^{-\lambda s}$$

- $m$: Number of letters in query
- $N$: Number of letters in db
- $mN$: Size of search space
- $\lambda s$: Normalized score
- $k$: minor constant
Gambler’s Ruin problem
In Sir Ronald Fisher we trust!
Dr. Margaret Oakley Dayhoff
The Mother & Father of Bioinformatics
Smith and Waterman at Los Alamos, New Mexico
Photo by David Lipman, Taken Summer of 1980
Karlin-Astschul Statistics Theory

- Samuel Karlin and Stephen Altschul
Ch. 2: Genome Assembly and Haplotype Assembly Algorithms

Examples of problems we need to solve along the way

Problem 1. Poisson statistics and DNA and Assembly

Problem 2. Ham Smith’s DNA breaking in a Lab with no windows

Questions: What algorithms to use to assemble DNA pieces into a contigs?
How long are the contigs?
How much the DNA target region is covered by the contigs?
Hamiltonian Paths Algorithms for Genome Assembly

Gene Myers

Craig Venter
Eulerian Paths Algorithms for Genome Assembly

Pavel Pevzner

Michael Waterman
Construct the sequence graph on \((k-1)\)-mers

- \(f_1\): TTCAGG
- \(f_2\): TTCATGG
- \(f_3\): ATGGACA
- \(f_4\): TTCAT
- \(f_5\): CATCGAC
- \(f_6\): TCGAC
- \(f_7\): GACATC
- \(f_8\): ACATCGA

(5, 1, 4)
(7, 3, 6)
(8, 2, 5)
Construct the sequence graph on (k-1)-mers

For each k-mer \((a_1...a_k)\), we create an edge between nodes labeled \(a_1...a_{k-1}\) and \(a_2...a_k\).

If those nodes do not exist yet, we add them to the graph.

We label the edge by its k-mer, \(a_1...a_k\).

We also store the set of position values \((f, i, j)\) in each edge, which identify all occurrences of that k-mer by (fragment index, start position, end position)*

<table>
<thead>
<tr>
<th>(f)</th>
<th>Sequence</th>
<th>(a_1...a_{k-1})</th>
<th>(a_2...a_k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TTCAGG</td>
<td>TTCA</td>
<td>TCAG</td>
</tr>
<tr>
<td>2</td>
<td>TTCATGG</td>
<td>TCAG</td>
<td>CAGG</td>
</tr>
<tr>
<td>3</td>
<td>ATGGACA</td>
<td>CAGG</td>
<td>TCAT</td>
</tr>
<tr>
<td>4</td>
<td>TTCAT</td>
<td>TCAT</td>
<td>CATG</td>
</tr>
<tr>
<td>5</td>
<td>CATCGAC</td>
<td>CATG</td>
<td>ATGG</td>
</tr>
<tr>
<td>6</td>
<td>TCGAC</td>
<td>ATGG</td>
<td>TCGA</td>
</tr>
<tr>
<td>7</td>
<td>GACATC</td>
<td>TCGA</td>
<td>CAGAC</td>
</tr>
<tr>
<td>8</td>
<td>ACATCGA</td>
<td>CAGAC</td>
<td>ACAT</td>
</tr>
</tbody>
</table>
Graph reductions: singletons

TTC

TCA

TCAT

CAT

CATC

ACAT

ATG

ATGG

TGG

TCGA

CGA

GAC

CAG

AGG

GGA

GGAC

TCAG

TTCA
Align the reads to the assembled sequence

First, we apply hashing methods to identify where each fragment might align well to the sequence.

This will produce “candidate diagonals.”

We can then perform alignment along those diagonals, which is more efficient than using the entire edit graph.
Statistics of Sequence Graphs: vertices

\[ \mathbb{E}(\text{True}) = L' \sum_{i=1}^{\infty} (1 - R^i) \mathbb{P}(X = i) \]

\[ = L' \sum_{i=1}^{\infty} \left( \frac{e^{-c} c^i}{i!} - \frac{e^{-c} (cR)^i}{i!} \right) \]

\[ = L' (1 - e^{-c(1-R)}) . \]

pmf of Poisson

\[ \Pr(X=k) = \frac{\lambda^k e^{-\lambda}}{k!} \]

using Taylor expansion for \( e^x \):

\[ e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \]

Summing the number of false vertices and true vertices produces:

The expected number of vertices \( \mathbb{E}(|V|) = RT + [1 - e^{-c(1-R)}]L' \).
Assembly Progression (Macro View)
Ch. 3: HMM - the Learning Problem

What does machine learning an HMM model mean?

Maximum Likelihood and the Expectation-Maximization problem
How do we reconstruct genealogies of a sample of individuals incorporating past mutations and recombinations?

Recombination + Phylogenetic Trees = ARG
Ch. 3: Spectral Clustering

GRAPH LAPLACIANS

- Quick example

Sentences in red and graphs are cited from A Tutorial on Spectral Clustering (Ulrike von Luxburg). See reference list at the end for detail.
Sentences in red and graphs are cited from A Tutorial on Spectral Clustering (Ulrike von Luxburg). See reference list at the end for detail.
RANDOM WALK POINT OF VIEW

• What is random walk?
  – A random walk on a graph is a stochastic process which randomly jumps from vertex to vertex.

• How does it walk?
  – Formally, the transition probability of jumping in one step from vertex \( v_i \) to vertex \( v_j \) is proportional to the edge weight \( w_{ij} \) and is given by \( p_{ij} := w_{ij} / d_i \).
  – The transition matrix \( P = (p_{ij})_{i,j=1,...,n} \) of the random walk is thus defined by

\[
P = D^{-1}W.
\]

• Initial condition?
  – a unique stationary distribution \( \pi = (\pi_1, \ldots, \pi_n)' \), where \( \pi_i = d_i / \text{vol}(V) \).

• Clustering in random walk?
  – Finding a partition of the graph, such that the random walk stays along within the same cluster and seldom jumps between clusters.
  – Intuitively, it is the same as the graph cut.

Sentences in red and graphs are cited from A Tutorial on Spectral Clustering (Ulrike von Luxburg). See reference list at the enfor detail.
Ch. 6 Suffix Trees in Linear Time
Ch. 7 Protein Folding Algorithms (Intro)

• Protein Folding on Lattice Models

• AlphaFold and Deep Learning
High-level Overview of Architecture of AlphaFold

Deep learning uses sequential modules (layers) to progressively extract information (learn) from the input data.
The Protein Folding Problem

Mixed character of the problem:

continuous mathematics -- geometry of surfaces &
discrete mathematics -- combinatorics of folds

Statistical Mechanics models