Bioinformatics Past Paper

- Algorithm, Software, Examples
  - [Which] Name
  - [Input] assumptions/conditions/constraints
    - foundation of reasoning / fill incomplete info/gap
    - exclude potential limitations
  - [Output] requirements
  - [How] methods, description, main steps
  - [Why] Motivation / Intention / Purposes
  - [Extension] comparison (pros and cons), speedup, application, complexity.

Lecture 1: Genetics

Reference online resource: Teaching

Youtube @bioinfalgorithms accompanying the textbook "Bioinformatics Algorithms: An Active Learning Approach".

features	DNA	RNA
strand structure	double helix	single-stranded
nitrogenous base	A, T, C, G	A, U, C, G
base pairs	A-T, C-G	A-U, C-G
sugar	Deoxyribose	Ribose
function	long-term storage of genetic info	protein synthesis and gene regulation
stability	more stable (-H)	stable due to hydroxyl group (-OH)

nitrogenous bases: Adenine (A), Thymine (T), Cytosine (C), Guanine (G), Uracil (U).

- The total number of codons is $4^3 = 64$.
- The total number of amino acids is 20.
- Codon degeneracy: multiple codons can are mapped toP the same amino acid, for redundancy if codon are mutated.

concept	gene	genome
definition	specific DNA segment for proteins or RNA encoding	complete set of genetic material
functionality	heredity, gene code for proteins	all genes and non-coding sequences
size	various sizes	entire DNA content of the organism

region and feature	DNA genes	programming functions
start marker	promoter region	function declaration (e.g.,`def func()`)
coding region	exons (encode protein information)	function body
non-coding region	introns (sequences)	comments or placeholder code
end marker	termination signal (of transcription)	closing brace or return statement
purpose	defines the structure and expression of a gene	defines the structure and behavior of funcs
execution	transcription and translation to produce proteins	execution of the function when called

DNA -- *transcription* $\rightarrow$ messenger RNA (mRNA) -- *translation* $\rightarrow$ protein, codon, amino acid.

Firstly, transcription happens, where a specific segment of DNA is copied into messenger RNA (mRNA). Secondly, translation occurs, where the mRNA is used to synthesize a protein, by attaching to a ribosome.

- The ribosome (composed of ribosomal RNA (rRNA) and proteins) reads the **codons** (three nucleotides of mRNA specifying a particular amino acid) until the ending signal appears.
- *Transfer RNA (tRNA)* molecules bring amino acids to the ribosome. Each tRNA has an anticodon that is complementary to the mRNA codon.
- The ribosome facilitates the formation of peptide bonds between amino acids, linking them together to form a *growing polypeptide chain*.

Lecture 2: Sequence alignment

Hamming distance vs edit distance

Dynamic programming (dp)

- Banded dp

Longest Common Subsequence (LCS)

- vs. global alignment
- edit distance $= m + n - 2 \times \mathrm{LCS(m, n)}$.

Scoring matrices

- Point Accepted Mutation (PAM) matrix

Gaps

- mismatch and gap extension penalty

    - If decrease gap penalty, more gaps (fewer sequence homologous regions), vice versa.
    - If decrease mismatch penalty / gap extension penalty, less gaps (more regions of similarity).
- non-linear, affine

Global alignment (Needleman-Wunsch)

- Gap penalty

Local alignment (Smith-Waterman)

Identify internal sequence duplications

- via self-alignment.

extension: *speedup* (Four Russians)

extension: *linear space* (Hirshberg, divide and conquer)

RNA secondary structure prediction (Nussinov-Jacobson folding)

- limitations: identify pseudo-knot and branched loops is difficult because the same interact with different segments.

Lecture 3: Phylogeny

Phylogenetic trees infer evolutionary relationships among biological species/entities based on their physical or genetic (DNA or amino acid sequences) characteristics similarities and differences.

Reference online resource: Hierarchical/Agglomerative clustering

	distance-based	**parsimony-based**
input	pairwise additive distance matrix	character tables
assumption	distances are additive	principle of minimal changes
good for	additive changes, faster	detailed and character-based
weakness	no information at internal node	homoplasy vs. shared traits
	over-simplification	slower for large scale
examples	UPGMA, neighbor-joining	small (or large) parsimony

Distance-based methods

Distance matrix

The *additive* tree condition meant that for any two leaves, the distance between them is the sum of edge weights of the path between them.

The ultra-metric tree condition: distance from root to any leaf is the same (i.e., age of root).

- Branch lengths represent evolutionary change, allowing for direct comparison of divergence among taxa.
- Molecular Clock Hypothesis: genetic change accumulates at a constant rate across lineages over time.
    - Thus, the rate of mutation or evolutionary change is uniform, allowing for the use of branch lengths as measures of time.
- Relationship: ultra-metric $\subseteq$ additive.

Four-point condition between four taxa: for any four elements, define $d_{ij} + d_{kl} = T$,

$$d_{ij} + d_{kl} < d_{il} + d_{jk} = d_{ik} + d_{jl} = T + 2a.$$

```
i \       / k
    --a--
j /       \ l
```

	UPGMA	**Neighbor-Joining (NJ)**
input	additive distance matrix	additive distance matrix
output	rooted ultra-metric tree	un-rooted additive tree
evolution rate	constant	varying
complexity	$O(n^2)$	$O(n^3)$

Parsimony-based methods

vs. distance methods

Sankoff parsimony

- mutation effect on phylogenetic analysis
- complexity
- extension: add/remove a node

Bootstrap validation

Multiple alignments

For a k-way sequence alignment of length n,

- there are n^k nodes in the alignment graph,
- each node has $2^k - 1$ incoming edges,
- the Hirshberg algorithm can get rid of one $O(n)$ in space complexity.

iterative refinement `CLUSTAL` algorithm

- given N sequences, align each sequence against each other.
- use the score of the pairwise alignments to compute a distance matrix.

- build a guide tree (tree shows the best order of progressive alignment).
- Progressive Alignment guided by the tree, by merging sub-alignments.

	dp	greedy	progressive (CLUSTAL)
time	$O(n^k \cdot 2^k)$	$O(k \cdot n^2)$	$O(k \cdot n^2)$
space	$O(n^k)$	$O(k) - O(n)$	$O(n) - O(k \cdot n)$
pros	global optimal	faster and less memory	balances both
		scales better with large k	good for moderate k
cons	high costs	suboptimal solution	guide tree accuracy
	for larger k	greedy alignment order	suboptimal solution

Evaluation

- entropy of a multi-alignment is calculated as a column score as the sum of the negative logarithm of this probability of each symbol.
- a completely conserved column would score 0, since $-[\log(1) + 3\log(0)] = 0$.

Approximate search

Lecture 4: Clustering

Clustering in gene expression microarray data

- compare expression levels in different conditions
- explore temporal expression levels evolution

K-center, K-means, Hierarchical, Markov clustering

Evaluation

Markov clustering (MCL) algorithm

- complexity

Soft K-means vs Hard

Louvain: modularity, Leiden algorithm

Lecture 5: Genome sequencing

Reconstruct the original genome, given a set of overlapping short reads from machines.

Hamiltonian graph -- Hamiltonian path (every node, NP-complete)

- Bellman–Held–Karp algorithm
  - boolean `dp[j][S_i]`, denoting a valid node subset S_i ending at node j.

    - for all neighbors k of j, extend `dp[j][S_i]` by `dp[k][S_i \ {j}]` and $k - j$.
    - $O(n^2 2^n)$, NP-complete.

De Bruijn graph -- Eulerian path (every edge, easier)

- Balanced node: in-degree = out-degree
- Semi-balanced node: in-degree = out-degree ± 1 (differs at most 1)
- Connected Graph: each node is reachable from some other node
- Strongly connected Graph: each node is reachable from every other node
- Eulerian Graph (a Eulerian cycle)
    - algorithm: Hierholzer's algorithm (ant) with $O(|E|)$.
- Euler's theorem
    - a connected graph is Eulerian if and only if every vertex has even degree.
    - a graph is Eulerian if and only if it is a balanced connected graph. (semi-)

E

Lecture 6: Genome assembly

Use an additional *reference* genome to augment sequencing or match (read) patterns.

Suffix trees

Compression

Burrows-Wheeler Transform (BWT)

Read / Exact **pattern matching**

- Sequencing De Bruijn graph construction takes a lot of memory and time.
- Fitting via alignment: $O(|\mathrm{Patterns}| \times |\mathrm{Genome}|)$.
- Joint traversal (match or backtrack) via two trie pointers in parallel.
    - patterns prefix trie: $O(|\mathrm{LongestPattern}| \times |\mathrm{Genome}|)$.
    - genome suffix trie: $O(|\mathrm{LongestPattern}| + |\mathrm{Genome}|)$.
        - construction: char nodes $T(|G|^2)$, $S(|G|^2)$; substr nodes $T(|G|)$, $S(\sim 20 \times |G|)$.
        - invert BWT + suffix array: $S(\sim 4 \times |G|)$.

Inexact pattern matching, with at most d mismatches.

- potential candidates: at least one of the $d + 1$ **seeds** is error-free. Check the entire pattern against the Genome.
- invert BWT with extended mismatch + suffix array.

Lecture 7: Hidden Markov models

Application: identify parts; Exons, Introns prediction; Protein secondary structure prediction; CG islands.

Evaluation: TP, FP, TN, FN, sensitivity, specificity, precision, F1 score

Denote HMM: transition matrix P, emission matrix Q, k states and M training sequences, with total number N each.

State sequence $\pi = \{q_1, ..., q_k\}$, observation sequence $X = \{x_1, x_2, \ldots, x_N\}$,

Algorithm	Inputs	Outputs	Time	Space
Viterbi / decode	HMM, X	π	$O(N \cdot k^2)$	$O(N \cdot k)$
Forward / eval	HMM	$Pr(x_1, ..., x_i \vert \pi_i)$	$O(N \cdot k^2)$	$O(N \cdot k)$
Backward / eval	HMM	$Pr(x_{i+1}, ...x_n \vert \pi_i)$	$O(N \cdot k^2)$	$O(N \cdot k)$
Viterbi train / learning	$X_1, ..., X_M$	P, Q	$O(M \cdot N \cdot k^2)$	$O(M \cdot N \cdot k)$
Baum-Welch / learning	$X_1, ..., X_M$	P, Q	$O(M \cdot N \cdot k^2)$	$O(M \cdot N \cdot k)$

Lecture 8: Computing and storage

DNA for Computing

Hamiltonian path problem (also knowns as the Traveling Salesman Problem) is NP-complete.

- to find a path in a graph $G = (V, E)$ that visits each vertex exactly once.

Leonard Adleman's DNA computing algorithm (1994) via generate and test.

- **Generate** all possible Hamiltonian paths in a graph G.
  - Step 1: encode the city names and routes as DNA sequences.
- **Test** each path to check if it is Hamiltonian,
  - total length of the path,
    - Step 2: sort by length in an electronic gel (field).
    - Step 3: filter by length via cutting out the band of interest.
  - start vertex, end vertex,
    - Step 4: amplify via PCR (Polymerase Chain Reaction) test.
  - each vertex once,
    - Step 5: affinity purification (hybridization) test of the complimentary strand.
- Output the Hamiltonian path.

vs. computational methods

- Advantages: synthesizing short single stranded DNA is now a routine process,
  - so the initial step is straightforward and cheap.
  - In a test tube the “algorithm” runs in parallel.
- However, the complexity still increases exponentially.
  - For Adleman’s method, what scales exponentially is not the computing time, but rather the

amount of DNA.
- Another limitation is the error rate for each operation.

Random access in DNA storage

Organick et al. (2018) stored and retrieved more than 200 megabytes of data.

- encoding: ID | Addr | Payload | Error correction code, append distinct primers, synthesis.
  - attach distinct primers to each DNA molecules set, to carry the file information.
  - redundant information for increased robustness.
- decoding: sequencing, cluster reads and consensus algorithm, error correction.
  - retrieve the file by selectively amplifying and sequencing the molecules with the primer marking the desired file.
- test their scheme via a primer library that allowed them to uniquely tag data stored in DNA.
  - encoded 35 digital files into 13M DNA sequences, each 150-nucleotides long.

Extension:

- opportunities (or advantages): longevity (durable), power usage and information density.
- challenges (or disadvantages): cost and read/write speed (DNA synthesis and sequencing).

Lecture 9: Stochastic Simulation Algorithm (SSA)

Dobb-Gillespie algorithm (1976)

- to simulate coupled biochemical reactions in a *well stirred* container, where the mean and variance from multiple runs are reported for statistical stability.
- assumption: the time steps τ so small that only one reaction has occurred.
- algorithm: given a set of M reactions, and N species in the system, with X_i molecules of species i.
  - $t = 0$, $\mathbf{X} = [X_1, \dots, X_N]$.
  - while $t < T$ do:
    - $\alpha_0 = \sum_{i=1}^{M} \alpha_i$, complexity $O(M)$.
    - $\tau = \mathrm{Exp}(\alpha_0) = -\frac{1}{\alpha_0}\ln(r_1)$, where $r_1 \sim \mathrm{Uniform}(0,1)$.
    - $t' = t + \tau$
    - $P(j\text{-th reaction}) = \frac{\alpha_j}{\alpha_0}$, where $j = 1, \dots, M$ and $r_2 \sim \mathrm{Uniform}(0,1)$.
    - $\mathbf{X'} = \mathbf{X} + \nu_j$, complexity $O(N)$.
  - end while
  - output $\mathbf{X'}$ and t'.
- utility: a better representation of cell metabolism and genetic networks.