1 General Formula

2 Biology Basics

3 Hierarchy of Biological Systems

Atom	Oragnism
Molecule	Population
Organelle	Community
Cell	Ecosystem
Tissue	Biome
Organ	Biosphere
Organ System	

3.1 High (macro) Levels – Ecology

Biosphere All life on Earth
Biome Continental scale grouping of ecosystems.
Ecosystem Groups of organisms from all biological domains within a physical environment.
Community Interspecific groups of interacting populations.
Population Groups of organisms of the same species.
Organism The basic living system, a functional grouping of lower-level components.

3.2 Middle Levels – Organization of Your Body

Organ system Functional groups of organs.
Organ Functional groups of tissues.
Tissue Functional groups of cells.
Cell Basic unit of all life and the grouping of organelles.
Organelle Functional groups of biomolecules, biochemical reactions, and interactions (e.g., nucleus, mitochondria).

3.3 Micro Level (Biological Molecules)

- Lipids
- Nucleic acids
- Carbohydrates
- Proteins

3.4 Nucleobases and the Genetic Code

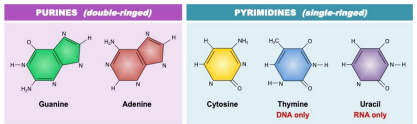
- Nucleobases (abbr.: bases) are the minimal units of genetic code.
- DNA/RNA comprises four types of bases.
- Matching bases form base pairs: A - T/U; G - C.

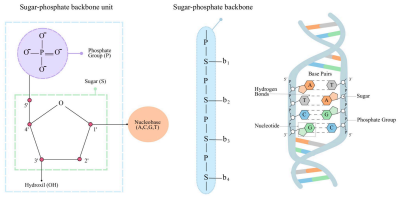
3.5 Central Dogma of Molecular Biology

DNA (Genome) Great book with instructions.
RNA Bona-fide transcript of relevant parts.
Protein Functional products.

3.6 DNA: Structure and Genome

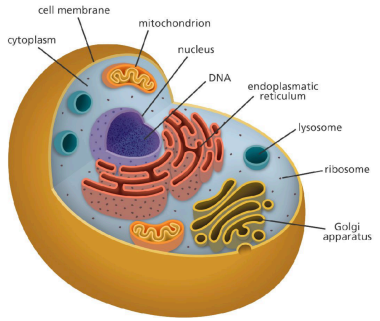
- DNA (deoxyribonucleic acid) carries genetic information.
- The entire set of DNA of a cell is a genome.
- The human genome consists of 46 chromosomes (23 pairs).
- DNA resides in the nucleus in the form of chromosomes.

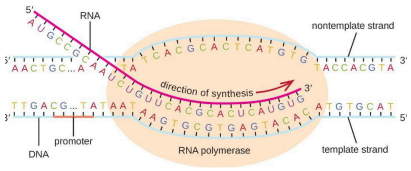


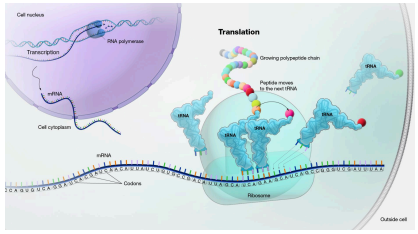


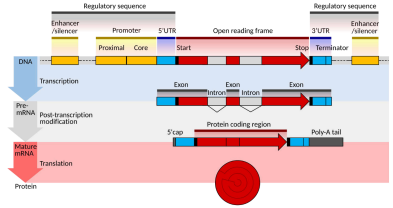
3.7 DNA Processes

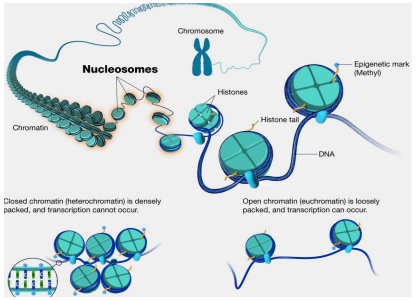
- DNA is replicated during cell division by DNA polymerase (an enzyme that catalyzes the synthesis of DNA molecules).
- DNA is transcribed into messenger RNA (mRNA) by RNA polymerase.
- mRNAs are translated into proteins in ribosomes (mini machines floating in the cytoplasm).
- The translation regulation machinery goes beyond the primary structure.
- Three nucleobases (called a codon) code one amino acid.











Second letter

	U	C	A	G	
U	UUU Phe UUC Phe UUA Leu UUG Leu	UCU Ser UCC Ser UCA Ser UCG Ser	UAU Tyr UAC Tyr **UAA Stop** **UAG Stop**	UGU Cys UGC Cys **UGA Stop** UGG Trp	U C A G
C	CUU Leu CUC Leu CUA Leu CUG Leu	CCU Pro CCC Pro CCA Pro CCG Pro	CAU His CAC His CAA Gln CAG Gln	CGU Arg CGC Arg CGA Arg CGG Arg	U C A G
A	AUU Ile AUC Ile AUA Ile **AUG Met**	ACU Thr ACC Thr ACA Thr ACG Thr	AAU Asn AAC Asn AAA Lys AAG Lys	AGU Ser AGC Ser AGA Arg AGG Arg	U C A G
G	GUU Val GUC Val GUA Val GUG Val	GCU Ala GCC Ala GCA Ala GCG Ala	GAU Asp GAC Asp GAA Glu GAG Glu	GGU Gly GGC Gly GGA Gly GGG Gly	U C A G

3.7.1 Alternative Splicing

Normal splicing All exons included
Exon Skipping One exon skipped
Alternative 5′ splice site Start of exon changes (Exon is shorter or starts at different spot than desired)
Alternative 3′ splice site (end of exon changes, ends earlier or later than usual)
Intron retention Intron isn't removed

4 Sequence Alignment

4.1 DNA Errors

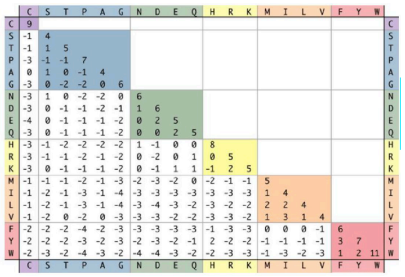
Substitution Flip one base for another (Mismatch Error)
Indel Insert/Delete base in sequence (Gap Error)
Transition Replace a Purine with a purine; replace pyrimidine with pyrimidine.
Transversion Convert Purine to pyrimidine, vice versa. More expensive than transitions
Transposon Delete/Insert whole region from sequence and insert to another

4.2 Similarity Heuristics

Hamming Distance Number of different positions within string

Ex: ATCTCA vs ATCACA is 1
Normalized Hamming Distance $\frac{\text{Hamming Distance}}{\text{len(Seq)}}$ (Assume sequences same length)
Edit Distance Minimum-weight series of edit operations transforming a into b

Ex: TGCAGT TCACAGT (Switch G to C, remove A, d= 2)
Substitution Matrix Cost of switching one base to another (since transitions and transversions cost different amounts)
BLOSUM62 Most classical amino acid substitution matrix



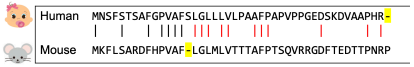
Levenshtein Distance Number of edits (indels + substitutions)

Dynamic Programming Reduce problems into subproblems to save results and reduce repeated computations.

- Requires optimal substructure (optimal solution contains optimal solutions to subproblems)
- Overlapping subproblems: Solution contains set of distinct subproblems repeated many times
- No Greedy Choice: Requires tracing back through many choices

4.3 Pairwise Sequence Alignment

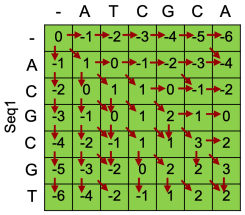
Pairwise Sequence Alignment Arrange two sequences so similar regions are aligned to each other.

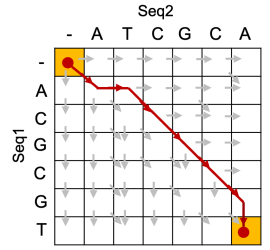


4.3.1 Global Alignment

Solves Global alignment (Overlap two entire sequences completely)
Time & Space Complexity: $O(mn)$
Needleman-Wunsch:

1 **Input**: Sequences seq1, seq2, match score (α), gap penalty (μ), mismatch penalty (σ)
2 **Output**: Alignment score, backtrack array
3 Initialize alignment + backtrack array of size seq1, seq2
4 Initialize top row and left column base cases
5 For every cell in matrix
6 if matching bases then add match score to score of left diagonal
7 if top/left cell + gap penalty is lowest score, add it
8 if left diagonal + mismatch penalty is lowest score, add it
9 put direction taken in backtrack array
10 output alignment score at bottom right cell, and use backtrack array to derive original alignment





1: **Input:** Sequences $seq1$ and $seq2$, match score(α), mismatch penalty (μ), gap penalty (σ)
2: **Output:** Alignment score and backtracking matrix
3:
4: $n \leftarrow$ length of $seq1$
5: $m \leftarrow$ length of $seq2$
6: Initialize score matrix S with dimensions $(n+1) \times (m+1)$
7: Initialize backtrack matrix B with dimensions $(n+1) \times (m+1)$
8:
9: **for** $i \leftarrow 0$ to n **do**
10: $S[i][0] \leftarrow i \times \sigma$, $B[i][0] \leftarrow$ 'UP'
11: **end for**
12: **for** $j \leftarrow 0$ to m **do**
13: $S[0][j] \leftarrow j \times \sigma$, $B[0][j] \leftarrow$ 'LEFT'
14: **end for**
15:
16: **for** $i \leftarrow 1$ to m **do**
17: **for** $j \leftarrow 1$ to n **do**
18: match $\leftarrow S[i-1][j-1] + (\alpha$ if $seq1[i-1] = seq2[j-1]$ else $\mu)$
19: delete $\leftarrow S[i-1][j] + \sigma$
20: insert $\leftarrow S[i][j-1] + \sigma$
21: $S[i][j] \leftarrow \max(\text{match}, \text{delete}, \text{insert})$
22: **if** $S[i][j] =$ match **then**
23: $B[i][j] \leftarrow$ 'DIAGONAL'
24: **else if** $S[i][j] =$ delete **then**
25: $B[i][j] \leftarrow$ 'UP'
26: **else**
27: $B[i][j] \leftarrow$ 'LEFT'
28: **end if**
29: **end for**
30: **end for**
31: **return** $S[n][m]$, B

1: **Input:** Sequences $seq1$ and $seq2$, Backtracking matrix B
2: **Output:** Aligned sequences and alignment score
3: **function** RecoverAlignment(i, j)
4: **if** $i \leftarrow 0$ and $j \leftarrow 0$ **then**
5: **return** '', ''
6: **end if**
7: **if** $B[i][j] \leftarrow$ 'DIAGONAL' **then**
8: $(\text{aligned\_seq1}, \text{aligned\_seq2}) \leftarrow \text{RecoverAlignment}(i-1, j-1)$
9: **return** $\text{aligned\_seq1} + seq1[i-1]$, $\text{aligned\_seq2} + seq2[j-1]$
10: **else if** $B[i][j] \leftarrow$ 'UP' **then**
11: $(\text{aligned\_seq1}, \text{aligned\_seq2}) \leftarrow \text{RecoverAlignment}(i-1, j)$
12: **return** $\text{aligned\_seq1} + seq1[i-1]$, aligned_seq2+'-'
13: **else if** $B[i][j] \leftarrow$ 'LEFT' **then**
14: $(\text{aligned\_seq1}, \text{aligned\_seq2}) \leftarrow \text{RecoverAlignment}(i, j-1)$
15: **return** aligned_seq1+'-', $\text{aligned\_seq2} + seq2[j-1]$
16: **end if**
17: **end function**
18:
19: $(\text{aligned\_seq1}, \text{aligned\_seq2}) \leftarrow \text{RecoverAlignment}(m, n)$
20: **return** aligned_seq1, aligned_seq2, $S[m][n]$

4.3.1.1 NCBI BLASTn Scoring scheme

Because whole regions get inserted/deleted, having n bases indeled is less costly than n indels. So we make gap opening a high cost, and continuing gap lesser cost.

Consant: Gap Penalty = Constant for all size n
Linear: Gap Penalty = Gap Length x Gap opening
Affine: Gap Penalty = Gap Opening + Gap Extension x (Gap Length - 1)
Convex: Gap Penalty <= Affine Gap Penalty

NCBI Blastn Scoring Matrix

Event	Score
Match	+2
Mismatch	–3
Gap opening	–5

Event	Score
Gap extension	–2

4.3.2 Local Alignment

Smith-Waterson Same as **Needleman-Wunsch** but we can start and end anywhere, and the goal is to find the lowest cost path overall.

0 is minimum, so initialize every cell as minimum of a continuation of path and 0, and the highest score is the ending of the alignment. To retrieve, we start at highest cell and go back until we reach 0.

4.3.3 Comparison

Global Alignment: Needleman–Wunsch Algorithm

- **Initialization**: Top-left, i.e. $M(0,0) = 0$
- **Iteration**:

$$M(i,j) = \max \begin{cases} M(i-1,j) + \text{Gap Pen} \\ M(i,j-1) + \text{Gap Pen} \\ M(i-1,j-1) + S(i,j) \end{cases}$$

- **Termination**: Bottom-right
- **Traceback**: Start from bottom-right and end at top-left

Local Alignment: Smith–Waterman Algorithm

- **Initialization**: Top row/left column ($M(0,j) = 0$ and $M(i,0) = 0$)
- **Iteration**:

$$M(i,j) = \max \begin{cases} 0 \\ M(i-1,j) + \text{Gap Pen} \\ M(i,j-1) + \text{Gap Pen} \\ M(i-1,j-1) + S(i,j) \end{cases}$$

- **Termination**: Anywhere
- **Traceback**: Start from the highest-scored cell and end when $M(i,j)$ is 0.

4.3.4 Database Search

k-mer Continuous block of k characters (def. 11 for nucl and 3 for prot)

High-Scoring Segment Pair Alignment found by BLAST word match

BLAST

1. Build sequence into **k-mers** (continuous block of k characters)
2. For each k-mer build index hash table of occurrences of all k-mers in target string
3. In construction of k-mers, allow some deviation along threshold of difference
4. Given a query, break it into k-mers and look up each k-mer as seed for sequence alignment
5. For each seed, extend seq until alignment falls below threshold

4.3.4.1 Variations

Gapped BLAST Extensions are in BLAST are allowed to use gaps. (So take gaps and continue aligning after the gaps)

Two-hit Seeding You can only extend a sequence if there is another sequence nearby it.

- Finding two-length words is more likely than a full word, so it's more sensitive.

4.3.4.2 Scoring

Raw Score Alignment score (S) of string

P value Probability of alignment score $\geq S$ given random strings

E value Expected number of alignments with score $\geq S$ given random strings

4.3.4.3 Alignment Scoring

- Used to compare alignments of same query-reference pair
- Used to compare alignments of diff query-reference pair
  - Determine whether (Q1, R1) or (Q2, R2) is more likely a homology
- Used to indicate confidence of alignment
  - Indicates whether alignment (Q1, R1) is actually homologous

4.3.4.3.1 Comparing Alignment of same query-reference pair

Define match (p), substitution (q), indel (r) probability. Probability is $p^m q^s r^i$, log-prob = $m \log(p) + s \log(q) + i(\log)r$ m, s, i are number of matches, substitutions, indels respectively.

Problems:

- Longer spotty seq's are penalized more than shorter perfect seq's
  - TGCAT_AG_GAT is very close to CCGTA_AG_AT, whereas TGCAGTAGGAT is scored worse than CCCACGAG-AT
- Repeated information is penalized (even though it's functionally the same)
  - TCAGT aligning to TCGGT is better than TCAGTTCAGT aligning to TCGGTTCGGT

4.3.4.3.2 Comparing Alignment of diff query-reference pair

Homology Model Assume alignment reflects evolution

Random Model Assume alignment spurred randomly

4.3.4.3.2.1 Homology Model

Probability: $\Pr(x, y \mid H)$

Assume p_{ab} is probability evolution gave char a in x and char b in y $P(x, y \mid H) = \prod_{i=1}^{n} p_{x_i y_i}$

4.3.4.3.2.2 Random Model

Probability: $\Pr(x, y \mid R)$

Assume p_a is probability to randomly get char a $P(x, y \mid R) = \prod_{i=1}^{n} p_{x_i} \prod_{j=1}^{n} p_{y_j}$

4.3.4.3.2.3 Likelihood Ratio

Likelihood Ratio: $P = \frac{\Pr(x,y \mid H)}{\Pr(x,y \mid R)} = \sum_i \log\left(\frac{p_{x_i y_i}}{p_{x_i} p_{y_i}}\right)$

Alignment score: $\log(P)$

4.3.4.3.2.4 Substitution Matrix

A matrix used to contain the unique probabilities of switching out one character for another.

BLOSUM62 is empirically found values on actual probabilities of switching out amino acids.

To calculate $\frac{p_{x_i y_i}}{p_{x_i} p_{y_i}}$ take a toy database of values, and calculate probability of getting pair (x, y) over all possible pairs in sequences (Number of pairs (x, y) over number of all pairs) and then divide that by probability of getting x and y individually (number of $\frac{x}{y}$ over all base pairs in list). Then substitution matrix only takes log values, so take the log.

How is the substitution matrix calculated?

- A toy database consisting conserved regions aligned into blocks:

```
AVQRLPECVAKPLWNVSNDLGLKPVLTYGDVCLTNCR
ACDTIPESVAAPLLKVSEALGLPPLATYAGLVLWNFC
PAEVLPRNLALPFVEVSRNLGLPPILVHSDLVLTNWT
```

- Summary: 3 rows, 37 columns, $3 \times 37 = 111$ amino acids, $\binom{3}{2} \times 37 = 111$ pairs
- For example: $s(I, L) = \log\left(\frac{p_{IL}}{p_I p_L}\right) = \log p_{IL} - \log p_I - \log p_L = \log \frac{3}{111} - \log \frac{2}{111} - \log \frac{21}{111}$
- In BLOSUM matrices these values are rounded to the nearest integer

4.3.4.3.2.5 Derivation of BLOSUM62

- Some protein families (like hemoglobin) are well-studied and will be overrepresented in our databases, so to create a matrix that properly reflects priors, we calculate similariy scores between sequences, and those with similarity scores greater than 62% are grouped together. The total weight of a group is 1, so if we have x, x', y where x, x' pass similarity threshold, but y doesn't, then x, x' each have weight 0.5 but y has weight 1. This is useful for creating a more representative protein database.

We can use substitution matrix to simplify calculating values $\sum_{x_i, y_i \in \text{ Sequences}} s(x_i, y_i) = \log(P)$

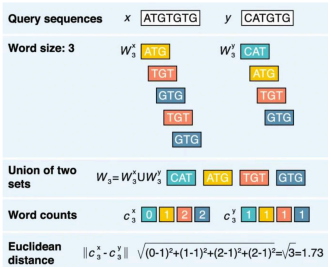
4.3.5 FASTA Format

> Title (Usually accession ID, Product, Organism) Sequence

- Sequences are standard IUPAC codes
- Lowercase allowed but mapped to uppercase
- Numbers not allowed in sequence

4.3.5.1 Alignment-Free Sequence Comparison

Given 2 sequences, chop them up into kmers. Then create a union between these two kmer lists. Then, create a word count of each kmer into a vector of the combined kmer list. Then take Euclidean distance between the two for a score.



4.3.5.2 Open Vocabulary Problem (Byte-Pair encoding)

To increase ability to scan sequence, take index of k-mers, sorted by frequency, and add that as a new "character". Iteratively, we reduce the number of tokens to scan.

4.4 Multi-sequence Alignment

Align k sequences

4.4.1 Sum-of-Pairs

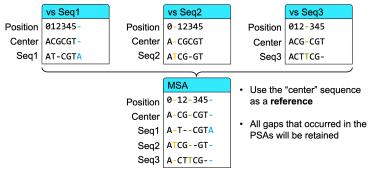
Do pairwise sequence alignment for every pair of sequences, and take the sum for a global alignment score, then optimize over that MSA for the alignment. Takes $O(n^2 k^2)$ time, because there are $\binom{k}{2} = \frac{1}{2}k(k-1)$ pairs of alignments

4.4.2 Centre-Star Method

Choose one sequence as center sequence by doing Pairwise Sequence Alignment on all pairs and find one sequence that has maximum of all alignment scores to be the center. This takes $O(k^2 n^2)$ time.

Then combine all the sequences by determining the gaps in each sequence at which position, and then insert them in every sequence. Ex: AC-GA, TC-AG, A-TCG => A-C–GA, T-C–AG, A-T–CG

This doesn't really concern alignment between non-centre sequences.



4.4.3 Progressive Alignment

Greedy algorithm that is a heuristic that doesn't visit all solutions but still gives good answer.

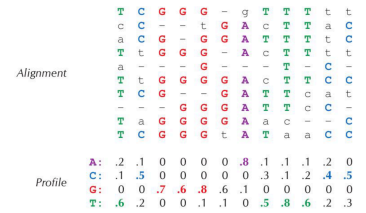
Do pairwise alignment on all sequence pairs, $O(k^2 n^2)$. Then build distance matrix among sequences via some heuristic. Then build a guided tree by some clustering algorithm and then select some starting sequence according to some optimal criteria (e.g. sequence most similar to rest), and align between node pairs, merging recursively for $k-1$ alignments.

4.4.3.1 Distance Matrix

Matrix with zero on diagonal, and non-negative values, symmetric, and satisfies triangle inequality. The values at coordinate i, j are just distances between sequence i and sequence j. Distance matrix is built from some heuristic: Hamming dist, Levenshtein dist, Substitution matrix-based dist.

4.4.4 Aligning new sequence

Then, if we have a new sequence to add, we gather our alignment of all the sequences, and capture the frequency of each possible character at each position for each character. Then the new "sequence" is the greatest frequency at each position, which you align to.



4.4.5 Applications of MSA

- Motif Finding: Detect conserved regions of a protein family as important motifs
- Structure Prediction: Alignment to protein family help to predict protein structure better
- Phylogeny: Phylogeny helps better examine evolution history among organisms

5 Phylogeny

MRCA Most Recent Common Ancestor (Root of Tree)

Internal Nodes Ancient hypothetical species

Leaf Nodes Extant species

Neighbour Two species that are most closely related on tree

Clade Group of organisms with common ancestor

Outgroup Set of organisms that are not in the "ingroup", and are distant in the tree.

Homolog Descendant of common ancestor

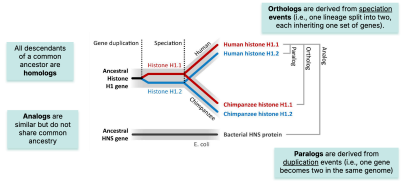
Analog Similar but don't share ancestry

Ortholog Speciation event, where common ancestors break into two species

Paralog One gene breaks into two separate versions in same genome

Phylogeny Construction of evolutionary tree by evolutionary relationship

Morphology Construction of evolutionary tree by physical featuers

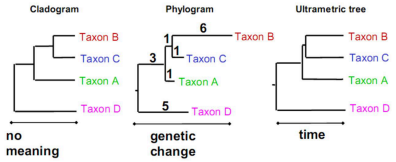


3 problems: **Topology**, **Root**, **Branch Lengths**

Cladogram Tree where branch length has no meaning

Phylogram Tree where branch length conveys genetic change

Ultrametric tree Tree where branch length conveys time



Hierarchical Clustering Arrange sequences based on pairwise distance in hierarchical manner

Well-behaved tree All nodes have equal distance from root to leaves (ultrametric), the distance is defined as "time". Also, the three-point condition is met, where for all distances, x, y, z, $d(x, y) \leq \max(d(x, z), d(z, y))$

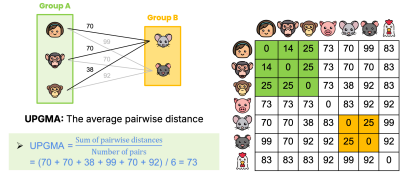
5.1 Distance-based methods

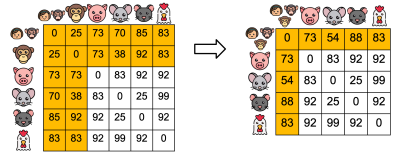
5.1.1 UPGMA

UPGMA Unweighted Pair Group Method with Arithmetic Mean

Linkage Criterion Distance metric between two groups (not just sequences, but 2 separate groups of sequences)

UPGMA (unweighted pair group method with arithmetic mean)





Note that distance from node to leaf is half of distance value in the matrix.

Given distance matrix, find closest pair of species, then combine them to a "new node" and recalculate the values for it and the remaining sequences by $\text{UPGMA} = \frac{\text{Sum of pairwise distances}}{\text{Number of pairs}}$ and continue until you're done.

This best describes topology and distance of species.

Ultrametric trees are represented by symmetric 0-diagonal matrices, where indices represent divergence events. Given ultrametric matrix, an ultrametric tree for that matrix exists if:

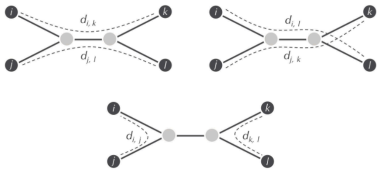
- There are n leaves, one for each row and column of matrix
- Each internal node is labeled by a time in D and has exactly two-children
- Along any path from root to a leaf, the divergence times at the internal nodes strictly decrease.
- For any triplet of nodes, the distances $D_{i,j}, D_{j,k}, D_{i,k}$ are either equal, or two are equal and the remaining one is smaller. (By 4-point theorem)

5.1.2 Neighbour Joining

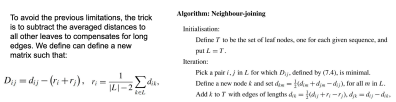
Additive Matrix There exists an additive tree fitting the matrix

Additive Tree Given symmetric $n \times n$ 0-diagonal matrix, for any neighbouring leaves i, j, with parent m, the distance from m to any other leaf k is $d_{k,m} = \frac{d_{i,k}+d_{j,k}-d_{j,i}}{2}$

Four-Point Theorem Distance matrix is additive iff two of the sums are equal and third is less than or equal to other sums:



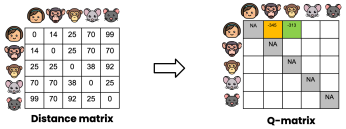
5.1.2.1 Procedure



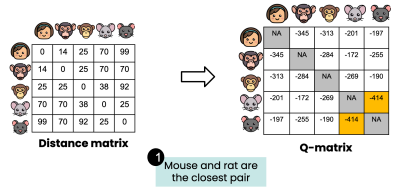
Procedure:

1. Create auxiliary Q-matrix where $Q(i,j) = (n-2)d_{i,j} - \sum_{k \in L} d_{i,k} - \sum_{k \in L} d_{j,k}$.
2. Merge closest pair wrt Q-matrix (lowest number in matrix)
3. Estimate branch length between chosen node and new node. $l(i,u) = \frac{1}{2}d(i,j) + \frac{1}{2(n-2)}\left[\sum_{k=1}^{n} d(i,k) - \sum_{k=1}^{n} d(j,k)\right]$. Note taht $l(i,u) + l(j,u) = d(i,j)$ Sometimes these branch lengths can be negative, so we can add a fixed constant to all branch lengths for non-negativity.
4. Update unrooted tree with new parent node between neighbours, and have respective lengths between them.
5. Update distance matrix so $d(u,k) = \frac{1}{2}[d(i,k) + d(j,k) - d(i,j)]$

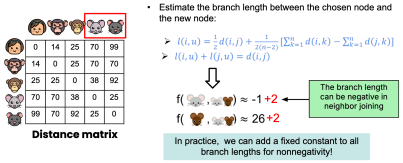
Step 1: Build a Q-matrix to **choose the closest pair to merge**

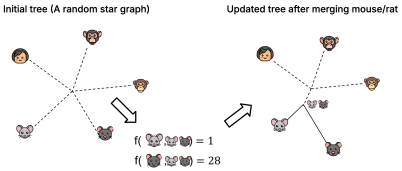


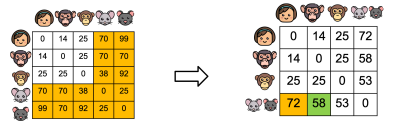
- The Q-metric considers the distance relative to other species
  - $Q(i,j) = (n-2)d(i,j) - \sum_{k=1}^{n} d(i,k) - \sum_{k=1}^{n} d(j,k)$
  - $= (5-2) \times 14 - (14+25+70+99) - (14+25+70+70) = -345$
  - $= (5-2) \times 25 - (14+25+70+99) - (25+25+38+92) = -313$



- Create a new internal node u:
- Estimate the branch length between the chosen node and the new node:
  - $l(i,u) = \frac{1}{2}d(i,j) + \frac{1}{2(n-2)}\left[\sum_{k=1}^{n} d(i,k) - \sum_{k=1}^{n} d(j,k)\right]$
  - $l(i,u) + l(j,u) = d(i,j)$







The distance is designed to be additive:

- $d(u,k) = \frac{1}{2}[d(i,k) + d(j,k) - d(i,j)]$
- $= \frac{1}{2}[70 + 99 - 25] = 72$
- $= \frac{1}{2}[70 + 70 - 25] = 58$

5.1.3 Jukes-Cantor

Jukes-Cantor is a tool to recompute distances based on back-mutations for more accurate phylogenetic trees.

Assumptions:

1. All nucleotide bases occur with equal probability
2. Each base has equal probability of mutating into any other with $r = 0.25$
3. Substitutions occur independently at each site over time

Let $p_{A(t)}, p_{C(t)}, p_{G(t)}, p_{T(t)}$ be probabilities that a given site contains nucleotide A, C, G, T at time t.

Since we assume all nucleotides are equally likely, $p_{X(t)}$ is probability that a site is $X(C, G, T)$ at time t

$\sum_{\delta \in A,C,G,T} p_\delta(t) = 1$ and if a nucleotide is A at time $t = 0$, then $p_{A(0)} = 1, p_{C(0)} = p_{G(0)} = ... = 0$

Define total mutation velocity away from A as α, then mutaiton velocity from A to C is $\frac{\alpha}{3}$, same with all other nucleotides.

For infinitesimal time step dt, $p_{A(t+dt)} =$ probability of staying A $= p_{A(t)} \cdot -p_{A(t)}\alpha dt + \left(p_{C(t)} + p_{G(t)} + p_{T(t)}\right) \cdot \left(\frac{\alpha}{3}\right)dt$

Similarly, $p_{X(t+dt)} =$ probability of becoming C, G, or T $= p_{X(t)} - p_{X(t)}\alpha dt + p_{A(t)}\left(\frac{\alpha}{3}\right)dt + 2p_{X(t)}\left(\frac{\alpha}{3}\right)dt$

Taking the limit as $dt \to 0$, $\frac{dp_A}{dt} = -\frac{4\alpha}{3}p_A + \frac{\alpha}{3}$ and $\frac{dp_X}{dt} = \frac{\alpha}{3}p_A - \frac{\alpha}{3}p_X$

Then we solve to get $p_{A(t)} = \frac{1}{4} + \frac{3}{4}e^{-\frac{4\alpha}{3}t}$ and $p_{X(t)} = \frac{1}{4} - \frac{1}{4}e^{-\frac{4\alpha}{3}t}$

Thus, the expected number of substitutions per site is: $d = -\frac{3}{4}\ln\left(1 - \frac{4}{3}p\right)$ which is our new distance function. $p = 3P_x$ is the observed fraction of nucleotide differences between sequences, and d is assumed to be αt to agree with p

5.2 Character-based Method

5.2.1 Maximum Parsimony

Score trees based on the minimum number of substitutions required to convey the found tree.

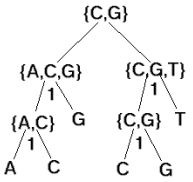
Parsimony Simplest explanation is usually best.

Pseudocode: Given tree and alignment column u, label interal nodes to minimize substitutions. **Initialization**: Set $C = 0$ and root node $k = 2N - 1$ **Iteration**: If k is a leaf, set $R_k = \{x_{k(u)}\}$ If k is an internal node with children i, j: If $R_i \cap R_j \neq \emptyset$, set $R_k = R_i \cap R_j$ Else $R_k = R_i \cup R_j$ and increment C

To score a tree using maximum parsimony we do the following:

1. Assign nucleotide to root node:
   - Choose arbitrarily from R_{2N-1}
2. For each internal node k, assign nucleotides recursively
   - if parent k has state r and $r \in R_i$, assign r to descendant i
   - Otherwise choose arbitrarily from R_j

Note that $x_{k(u)}$ means the u'th base pair for node k. Then, for internal nodes, you take the intersection as the list of possible values that wouldn't require a substitution. If there is nothing, then join them, so that you only require one substitution.



Note this doesn't build a tree, but scores it.

5.2.2 Maximum Likelihood

The likelihood of a tree T with branch lengths t and given sequence alignment is $L(T,t) = P(x_1, ..., x_n \mid T, t)$

We want to find the most likely tree given sequence alignments. Note, $L(x_1, x_2) = \sum_X \pi_X P(x_1 \mid X, t_1) P(x_2 \mid X, t_2)$

π_X is stationary probability of nucleotide X, X is possible ancestral states at root. $P(x_i \mid X, t_i)$ is probability of transition from X to x_i in time t_i

$L_{k(X)}$ means likelihood of subtree rooted at k, given it has state X $P(X \to Y, t)$ is probability that state X at a node changes to Y over time t π_x is stationary probability of nucleotide X at the root.

Pseudocode: **Initialization**: At each leaf node, define $L_{i(X)} = 1$ if $X = x_i$, 0 otherwise. **Recursion**: For each internal node with children i, j $L_{k(X)} = \sum_{Y,Z} P(X \to Y, t_i) L_{i(Y)} \cdot P(X \to Z, t_j) L_{j(Z)}$ **Final Step**: $L(T) = \sum_X \pi_X L_{\text{root}}(X)$

5.2.3 Bayesian Inference

Like maximum likelihood, based on probabilities, but tree topology is not single tree, but probability distribution of all trees. Uses Bayes' theorem to calculate posterior probability of trees based on prior probability of trees and observed data. Samples from probability distribution and updates model states iteratively

5.2.4 Tree Space Search

Stepwise Addition Start from minimum tree (3 taxa) and add taxa, then optimize topology.

Use starting tree A tree with all taxa but is less accurate. Either use a cheap method for construction or randomly create.

5.3 Summary

Methods	Optimality Criterion
Distance Methods	Distance
Maximum Parimsony	Parimsony
Maximum Likelihood	Likelihood

Note top-to-bottom is increasing accuracy and decreasing efficiency.

5.3.1 Newick Tree Format:

Newick Tree format is a text file like this:

```
(((monkey:0.01572,chimp:0.01038):0.27040,chicken:0.37215):0.12011,
(pig:0.08412,(rat:0.03239,mouse:0.03430):0.19210):0.06343,human:0.03265);
```

- () -- group (hierarchy)
- , -- siblings
- : -- branch length (optional)
- ; -- end of tree

- **For example:** ((A,B),C);

6 Genome Mapping

6.1 Sequencing

We get sequencing from either Sanger sequencing, Next Generation Sequencing, to produce DNA fragments.

Long (>10000 bp) Reads (Pacbio/Nanopore) typically have a high error rate, and short reads (Illumina) have a bigger computational task, but higher accuracy.

6.2 Assembly

The goal is to derive the original string from a collation of fragments. The original string is first copied multiple times, and then fragmented, so there's no guarantee that the fragments are one after another.

Coverage: Number of fragments overlapping for a specific position of the original sequence.

Note: If there is a lot of overlap between the end of one fragment and start of another, it is likely that they were overlapping in the actual genome.

6.2.1 Long Read

1. Get Reads
2. Build overlap graph
3. Bundle stretches of the overlap graph into **contigs**
4. Pick most likely nucleotide sequence for each contig
5. Output Contigs

Contig: Set of DNA segments that overlap in a way that provides contiguous representation of genomic region.

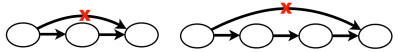
k-mer composition: Collection of all k-mer substrings of the Text, including repeats.

k-mer Prefix/Suffix: First/Last $k-1$ nucleotides of k-mer

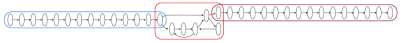
Then connect all k-mers of the text in a graph, where for given (Text) nodes with Suffix(Text) = Prefix(Text') should have an edge between them.

Then, to solve this problem, we want a Hamiltonian path (going through every node) to solve, so we can use heuristics to solve.

For instance, remove transitively-inferrible edges, edges skipping one or two nodes.



Then, create a contig for all non-branching stretches.



Then, derive a consensus by taking all reads tha tmake up a contig and line them up and take consensus (i.e. through majority vote).

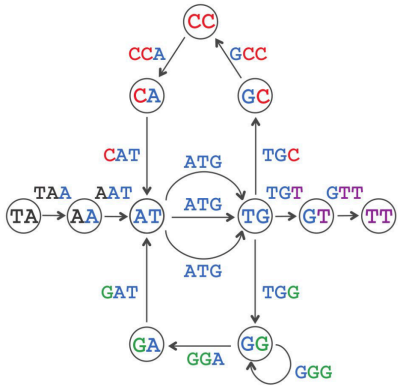
6.2.2 Short Read Assembly

1. Error Correction
2. Graph Construction
3. Graph Cleaning
4. Contig Assembly
5. Scaffolding
6. Gap Filling

Error Correction means preprocessing reads to find errors and only accepting exact matches and try to replace rare k-mers with common k-mers.

6.2.2.1 De Brujin

Graph Consruction: Given collection of k-mers P, define nodes as all unique $k-1$-mers that are prefixes or suffixes of k-mers in P and for each k-mer in P create a directed edge from its prefix to its suffix.



Euler's Theorem: Every balanced, strongly connected directed graph has a cycle that visits every edge exactly once.

Form a cycle by randomly walking in a graph, and not revisiting edges. While unexplored edges remain, select a node, newStart, in Cycle with unexplored edges and form a new cycle by traversing Cycle starting at newStart, then randomly walk to include new unexplored edges.

Note that you can turn an Eulerian path into an Eulerian cycle by adding an edge between two unbalanced nodes making it a balanced graph.

Graph Cleaning: Remove "sink" nodes in De Brujin graph and combine divergence structures that converge to a single node later through the graph.

Contig Assembly: Use Euler algorithm to identify strongly connected regions in the genome and use that path to determine the full contig.

6.2.3 Computational Problems in Sequencing Data

- Read Mapping/Alignments: Map/align reads/fragments back to known genome
- Variant Calling: Detect positions varying from reference population
- Genome Assembly: Reconstruct full chromosome from short/long sequencing reads/fragments

Seed-and-extend To find target regions on reference genome that are at most k mutations between read and target build alignment from seed regions instead of global/local alignment.

6.3 Indexing Data Structures

6.3.1 K-mer index

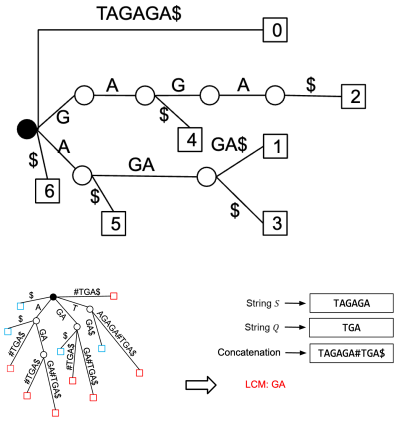
Generate list of words of length k in genome string. For each kmer build index table with all occurrences in the reference, and for each kmer of query, find its hits in the index table.

6.3.2 Suffix Tree

Suffix Substring S_i of s starting at position i.

- Start with full string S and empty root.
- Add each suffix successively and label leaf with position.
- Use existing edges where possible.
- Merge edges that have no branches and concatenate labels for better space efficiency.

Pattern Matching: To see if query string Q is substring of S, start at root and traverse tree according to query. Longest Common Substring: Build substring for concatenated string $S\#Q\$$ and label leaves depending on whether suffix belongs to S or Q. Common substrings, as internal nodes, are prefixes of two suffixes from both S and Q, so find node with longest path to the root.



6.3.3 Suffix Array

Sorted list of all suffixes of string S.

- Made by generating list of suffixes, sorting them in lexicographical order. $A[k]$ is start position of k-th least-sorted suffix.

```
def pattern_matching_with_suffix_array(S, Q, suffix_array):
    suffixes = [S[i:] for i in suffix_array]

    left, right = 0, len(suffixes)
    # First Binary Search for L_Q (First occurrence)
    while left < right:
        mid = (left + right) // 2
        if suffixes[mid] < Q:  # Move right if suffix is smaller
            left = mid + 1
        else:
            right = mid  # Move left otherwise
    first = left
    right = len(suffixes

    # Second Binary Search for U_Q (Last occurrence)
    while left < right:
        mid = (left + right) // 2
        if suffixes[mid].startswith(Q):  # If Q is a prefix, move right
            left = mid + 1
        else:  # Otherwise, move left
            right = mid
    last = right-1  # Adjust to return last valid match
    if first > last:
        return f'"{Q}" does not appear in S'
    else:
        return (first, last)  # Now properly inclusive
```

6.3.4 Burrows-Wheeler Transformation

Lossless string compression algorithm, made by generating all rotations of string S, sorting rotations lexicographically, and keeping only last column as output. Can be further reduced by run-length encoding.

Note that we rotate to the left, so TAGAGA$ becomes AGAGA$T

Three ways of inverting:

Sort and add Given Iteration i sort lexicographically, then add original BWT to beginning. Continue until you reach same length as input string, and take string with $ at the end. Each iteration takes $O(n \log n)$, so total complexity is $O(n^2 \log n)$

6.3.4.1 LF-Mapping

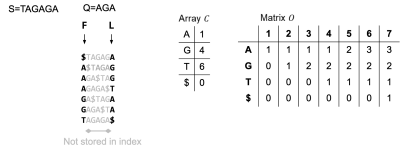
First-last property k-th occurrence of a symbol in First Column and k-th occurrence of symbol in Last Column correspond to the same position of this symbol in Text.

Last column is the BWT, first column is the sorted characters of BWT.

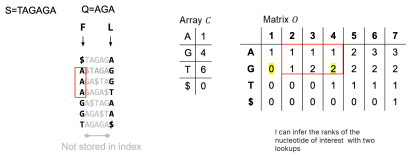
6.3.4.2 FM-Index Pattern Matching

Combines BWT for compression and suffix array for efficient search.

C array has $C[c]$ = total number of occurrences of characters $< c$ in F (sorted characters) O matrix has $O[c, k]$ = number of times c occurs in $L[1 : k]$ (Start from index 1)



To pattern match, do the following:



6.3.5 Comparison

	kmer index	suffix tree	suffix array	BWT & FM Index
Space	$O(R)$	$O(R)$ with pointers	$O(R)$	$O(R)$
Search Time	$O(q)$	$O(q)$	$O(q \log R)$	$O(q)$

7 Machine Learning

Dimensionality Reduction: Reduce higher-dimensional data into lower-dimensions to reduce the effect of noise and irrelevant features to create a more robust representation of data patterns.

Representation Learning: Find data representations in lower dimensions by finding patterns in large datasets. Goal is to find task-agnostic features from data without manual design.

Use case example: From Raw Data derive a feature representation. Then, from those features create a task-specific model to solve some problem. We lower our complexity by abstracting away the raw data.

Traditional Feature Engineering: Manually designed features tailored for specific tasks.

Traditional Feature Engineering vs Representation Learning: The features are either given (by domain knowledge) or found (by computation)

7.1 Deep Neural Networks

- Neural networks are made of multiple layers. Given input x, neural network is $f_\theta = f_L \circ ... \circ f_1$ where f_1 is input layer, f_L is output layer, and intermediate layers are hidden layers.

Feedforward Neural Networks: Layers of interconnected neurons, where each layer's output is passed to the next. Perfect for dense, unstructured data with fixed set of features.

$h^{(l)} = \sigma\left(W^{(l)} h^{(l-1)} + b^{(l)}\right)$

Activation Functions:

- ReLU: $\sigma(x) = \max(0, x)$
- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- Tanh: $\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Convolutional Neural Networks: Fits structured data with spatial bias (data points "near each other" are related). Each convolutional filter captures local features in a fixed input size.

$f_{i,j}^{(l)} = \sigma\left(\sum_u \sum_v W_{u,v}^{(l)} \cdot h_{i+u,j+v}^{(l-1)} + b^{(l)}\right)$ Basically each filter multiplies the kernel against the "kernel" for each window in the filter, and computes this for all pixels necessary.

Convolution Pooling: Pooling is the method of reducing the dimension of information for a region of information, to enhance translation invariance. Reduces computation and overfitting by **downsampling** feature maps.

Max Pooling: Return maximum value within a region
Average Pooling: Return average value within a region.

Convolutional Neural Networks Hyperparameters:
Kernel Size (k): Size of kernel
Stride (s): Step size for moving filter
Padding (p): Number of zero-padded pixels to input

Output Size: $\left\lfloor \frac{\text{Input Size} + 2p - k}{s} \right\rfloor + 1$

Padding Strategies:

- No padding ($p = 0$) Output smaller
- Same (padding to maintain size): $p = \frac{k-1}{2}$ if $s = 1$

	Inputs	Specialty	Advantages	Limitations
FNNs	Fixed Features	Unstructured Data	Simplicity	No spatial/ temporal relat.
CNNs	Structured Data	Spatial/ Local patterns	Spatial bias	Fixed input size
Transformers	Sequential Data	Temporal Dependencies	Long-range dependencies	Computational Cost

Probabilistic Interpretation of Supervised Learning:

- $f(x; \theta)$ is conditional probability of target y given input x: $p(y \mid x, \theta)$, same as approximating posterior distribution of y given x.

Universal Approximation Theorem: A sufficiently deep neural network with non-linear activation functions can approximate any continuous function arbitrarily well.

Supervised Learning Outputs: Sigmoid for binary classification, softmax for multi-class classification, identity for regression.

Loss functions in supervised learning:

- Negative Log Likelihood/Cross-entropy loss (Classification): $\mathcal{L} = -\sum_{i=1}^{N} \sum_{j=1}^{K} \tilde{y}_{ij} \log p(y_j \mid x_i, \theta)$ where $\tilde{y}_{ij}$ is one-hot encoded true lable and $p(y_j \mid x_i, \theta)$ is model's predicted probability.
- Mean Squared Error (Regression): $\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$

7.2 Optimization Methods

Stochastic Gradient Descent: Updates are based on gradient of loss for randomly selected mini-batch. $\theta \leftarrow \eta \nabla_\theta \mathcal{L}(\theta; \beta)$.

Momentum: Gradient Descent + considers past gradients to find smooth updates and prevent oscillations and navigating steep ravines and speeding up convergence.

$v_t = \beta v_{t-1} + (1 - \beta) \nabla_\theta \mathcal{L}\theta \; \theta \leftarrow \theta - \eta v_t$

v_t is velocity, β is momentum factor, and μ is learning rate.

Adam Optimizer: Combines momentum with adaptic learning rates for parameters. Best for sparse data.

$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_\theta \mathcal{L}\theta \; v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_\theta \mathcal{L}\theta)^2$
$\theta \leftarrow \theta - \eta \frac{m_t}{\sqrt{v_t} + \varepsilon}$

m_t is estimate of mean of gradients, v_t is estiamte of uncentered variance of gradients, β_1, β_2 are control decay rates.

AdamW Optimizer: Adam but with a weight regularization term. Best for large models (improves generalization through weight regularization term) $\theta \leftarrow \theta - \eta\left(\frac{m_t}{\sqrt{v_t} + \varepsilon} + \lambda\theta\right)$

Choosing Optimizer:

- SGD: Good for simpler models, but slow convergence in complex landscapes
- Momentum: Speeds up convergence by incorporating past gradients
- Adam: Combines momentum and adaptive learning rates, ideal for sparse data
- AdamW: Best for large models; improves generalization with decoupled weight decay

Best to start with Adam for most neural networks; for large-scale fine-tuned models, AdamW is better due to weight decay. Momentum-based SGD is common for image tasks and good generalization.

Transfer learning: After learning, we can use learned representations from intermediate layers to perform other tasks on smaller dataset, transfering knowledge learned from D, which is useful for leveraging knowledge from one task to improve performance on another. Good for domains with limited labelled data, like medical imaging. Often involves pre-training a model on a large dataset and then fine-tuning on the target task.

7.3 Normalization

Batch Normalization: Normalize input across inputs in batch to generalize inputs

$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \; y_i = \gamma \hat{x_i} + \mathrm{B}$

μ_B, σ_B are batch mean and variance, and the others are learnable parameters.

Layer Normalization: Normalize across features (within one sample, normalize features) to improve sequential model performance.

$\hat{x_j} = \frac{x_j - \mu_L}{\sqrt{\sigma_L^2 + \varepsilon}} \; y_j = \gamma \hat{x_j} + \mathrm{B}$

μ_B, σ_B are sample mean and variance.

8 Machine Learning in Genomics

Genomic Signature: Unique frequency of oligonucleotide chains within a genome

DNA Barcoding: Identifying a species by a characteristic sequence of a standard short section of DNA in genome.

Convolutional Neural Networks: Designed for structured data w/w spatial bias

Maxpooling layer: Convolutional layer that takes largest value in pool

8.1 DNA Barcoding

Identifying species based on short, standardized region of DNA by colelcting sample; extracting DNA from sample; amplifying specific gene region (barcode) using PCR. [For animals we use mitochondiral COI gene]. Sequence the barcode region. Compare sequence to reference database to identify the species.

8.2 Applications

Transcription Factors and RNA-binding proteins: Regulating factors of genes by binding to DNA/RNA

Splicing defects, genetic mutations, pathogenic variants, can all lead to improper genome sequencing, which is a hurdle for accurate DNA barcoding.

SpliceAI: Deep Learning model trained on GENCODE-annotated exon-intron boundaries; With large sequence window it is able to learn splicing patterns, and identify pathogenic splicing mutations.

DeepBind: Deep Learning Model using CNN to identify where

along gene sequences proteins (Transcription Factors) are most likely to bind. Then, with this model you can predict bind affinity for new sequences.

DeepVariant: CNN-based model trained on labeled sequencing data; converts aligned reads to images; predicts substitutions and indels with high precision and recall.

8.3 Evaluation

Training Set: 60-80% of data Validation: Used to tune hyperparameters (10-20% of data) Test: Used to assess performance (10-30% of data)

Type 1 Error: Predicted Positive, Actually Negative
Type 2 Error: Preidcted Negative, Actually Positive

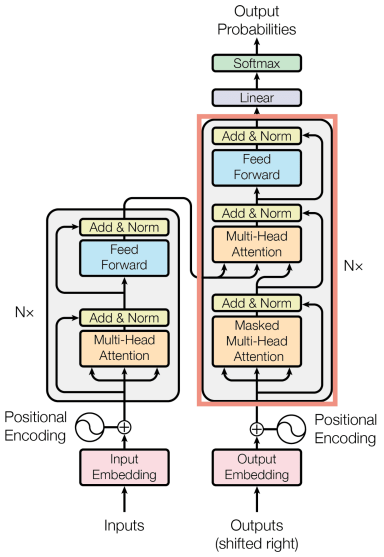
Precision: Sum of True Positive over all predicted positive
Recall (Sensitivity): Sum of True Positive over all Actually Positive
Specificity: Sum of True Negative over all Actually Negative

Accuracy: Sum of Correct estimations over all predictions

F1 Score: $\frac{1}{\frac{1}{\text{Recall}}+\frac{1}{\text{Precision}}}$

9 Transformers

Used for predicting sequential data



9.1 Input Embedding

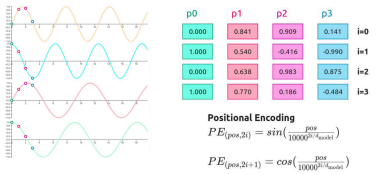
Input Embedding: Input tokens are mapped to dense vector representations using an embedding layer. $W_0 \in \mathbb{R}^{N_{\text{dict}} \times d}$ N_{dict}: Vocabulary Size. d is embedding dimensions.

W_0: Learnable lookup table, where embeddings learned iteratively.

9.2 Positional Embedding

Positional Embedding: To encode positional data, the vector is encoded based on its position within the string into a series of sin and cosine frequencies, which is beneficial because it is normalized and restricted, allowing invariance between varying length strings, but also ensuring positional value.

Positional Encoding formula: $PE_{\text{pos},2i} = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$ and $PE_{\text{pos},2i+1} = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$



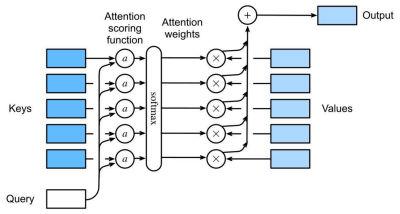
Helps model adapt and capture sequential patterns in data. Note that positional encoding can be learned as part of model's parameters.

9.3 Attention

Compared to fixed-weights from NNs, Attention dynamically computes weights based on input values.

Attention is conceptually a soft lookup table, retrieving relevant information based on learned similarity.

$\text{Attn}(q,(k_{1:m},v_{1:m})) = \sum_{i=1}^{m} \alpha_{i(q,k_{1:m})} v_i$ The attention weights are computed by query-key similarity.



q is query (current token input), k is previous set of inputs, v is previous set of computed encoded values.

Attention weights are derived from attention score function, which measures similarity between query and each key. Then attention weights are computed using softmax function.
Attention weight $\alpha_i = \alpha_i(q,k_{1:m}) = \text{softmax}_{i([a(q,k_1),\ldots,a(q,k_m)])}$ where a is attneition score function.

Generally, attention is represented as: $Z = \Phi(W(Q,K)V)$ where $Q \in \mathbb{R}^{n \times d_q}$ are queries, derived from X, describing what each input is "looking for"

$K \in \mathbb{R}^{m \times d_q}$ are keys, derived from X, describing what each input vector contains.

$V \in \mathbb{R}^{m \times d_v}$ are values, indicating how each input should be transmitted to the output.

Typically these values are found as linear projections of input: $Q = XW_q, K = XW_k, V = XW_v$

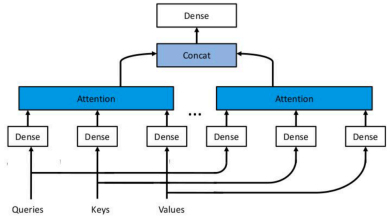
When computing attention over mini-batches of n input vectors, the attention-weighted output is: $\text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \in \mathbb{R}^{n \times d_v}$ where softmax is applied row-wise, for parallelism. (We scale by $\sqrt{d}$ to reduce exploding gradients.)

9.4 Multi-Head Attention

Instead of single attention function we can use multiple projections to learn different dependencies.

$T_i = \text{softmax}\left(\frac{XW^i_{q(XW^i_k)^T}}{\sqrt{d}}\right)XW^i_v$

Then, the output is $\text{MHA}(X) = \text{Concat}(T_1,\ldots,T_h)W_o$



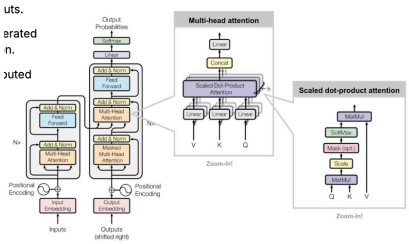
9.5 Residual Connections

To avoid vanishing gradients in deep networks, residual connections allow gradients to bypass layers. Ex: $f_{l'}(x) = F_{l(x)} + x$ allows the network to learn a residual correction instead of a full transformation, creating stable optimization and deeper networks. $f'_{l(x)} = F_{l(x)} + x$

9.6 Layer Normalization

$\hat{x_i} = \frac{x_i - \mu_L}{\sqrt{\sigma^2_L + \epsilon}}, y_i = \gamma\hat{x_i} + \beta$ where μ_L, σ_L are mean and variance of activations, and γ, β are learnable parameters.

Each Transformer block contains feedforward Multi-Layer-Perceptron: z = LayerNorm(x + MultiHeadAttention(x))
z = LayerNorm(z + MLP(z)) The MLP captures more complex dependencies and transformations.



10 DNA Foundation Models

Language Model: Predict next word based on context by finding context-aware word representations.
Pre-Trained Language Model: Deep Learning models trained on big datasets to learn linguistic patterns and representations
Large Language Model: Large-sized pretrained language model
Statistical Language Model: Probabilistic Model
Neural Language Model: Neural Networks to predict likelihood of sequences of words
Foundation Model: Deep Learning Model trained on large datasets to be applicable in many use cases.

LLMs have way more parameters; more emergent abilities; require ridiculous amounts of resources; and have more powerful features than regular Language Models

Benefits of Transformers are dynamic computation (information is context-specific), good memory usage, parallelizability.

Transformer Architecture Components:
- Encoder-only architectures use representations for other tasks. Discriminative so model can give predictions for embeddings.
- Decoder-only Generative model predicting next word / future one word at a time.

Unsupervised Representation Learning: Given unlabeled data
Self-Supervised Representation Learning: Given unlabeled data, generate pseudo-labels by designing a proxy task to produce targets based on patterns in data.

For example, given an image, you can "remove" certain sections, run your encoder + decoder to get a prediction of that removed section, and then compare the true removed section to the prediction. (No provided labels, but supervised learning occurs)

BERT vs GPT Transformer Architectures:

Masked Language Model: LM trained to predict masked/ missing words within text, using surrounding context to find structure.

Autoregressive LM: LM trained to predict next word in sequence based on previous words, "autoregressively" calculating by using previous predictions.

Discriminative Model: Separate data in to different classes

Generative Model: Generating new data points

	BERT	GPT
Training	Masked LMs	Autoregressive LMs
Model Type	Discriminative	Generative
Pretrain Task	Predict masked words	Predict next word
Direction	Bidirectional	Auto-regressive

Bidirectional model means "future" tokens affect previous token probability (DP), auto-regressive means only past affects future (Greedy)

10.1 DNA as language

- Full understanding of DNA isn't just nearby characters but functional annotation at many levels. Need to understand DNA at various granularities (protein language + codon language + regulatory DNA + RNA language models) all at once.

Mask Model Training: Mask a specific token in a word and train algorithm to predict that word.

10.2 Building Foundation Models

Instead of creating a custom task-specific model, we can create a general-purpose trained model that is pre-trained, and then fine-tune for the task we want with shorter computation required.

K-mer Tokenization: Split sequence into sequence of kmers, which are then mapped to an index.

Problem: K-mer size can grow exponentially, so instead we fine a new vocabulary based on training data via Byte-Pair Encoding.

Byte-Pair Encoding: Start with characters. Repeatedly merge the most common pair of symbols into one. Do this until you hit a desired vocab size.

Once we have a BPE, given a new text, we start with a set of characters, and assign merge rules to reach our BPE tokenization.

10.3 Applications

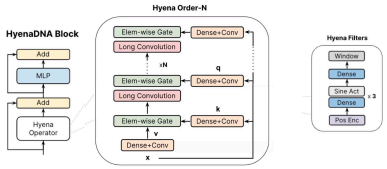
DNABert: Pre-Trained BERT for DNA sequences based on human references genome using overlapping k-mer tokenization. Goal: Wide-range applications (promoter region/ transcriptional factor binding site/splite side prediction). To overcome input length limit of DNABERT, you replace positional embeddings with attention with linear biases, so if tokens are far apart, decrease bias to increase attention to nearby vectors.

BarcodeBERT: Pre-trained BERT for DNA sequences based on Barcode data using non-overlapping k-mer tokenization for taxonomic classification.

DNA transformer design choices: Overlapping Kmers tokens, Non-overlapped, Byte-pair encoding.

10.4 Hyena Operators

To avoid the quadratic operation in tokens through Attention, they use Convolutions instead, in the Transformer-like-block. Transformers are $O(N^2)$ in time complexity, whereas convolutions are $O(Nk)$ Training process was to take sample from human genome, and use next token prediction on single nucleotide tokens. Hyena operators are good for regulatory element classification (short scan) or chromatin profiling, and species classification (long scan).



10.5 Eukaryotic DNA Terminology

Splicing: Post-transcriptional modification, removing specific introns to create a mature mRNA for protein synthesis.

Promoters: Regulatory sequence next to start codon enhancing transcription.

Enhancer/Silence: Far away sequences modifying frequency of transcription of DNA.

Insulator: Regulatory sequences ensuring enhancers and promoters don't promote other genes it's not supposed to.

Nucleosome: Region of DNA wrapped around histones.

10.6 Benchmarking tasks

Nucleotide-wide tasks: Check each base pair one by one for accuracy

Sequence-wide tasks: Check the entire sequence for correctness

Binned tasks: Checking if specific segments of DNA (as one block) are correct.

11 Genome Annotation

Structural Annotation: Identify locations of genes in a genome (aka gene prediction)
Functional Annotation: Determine functions of genes

11.1 Gene Prediction

Get set of genes and observe them, try to find what's common between them, and search the genome foor these patterns.

11.2 Structural Annotation

Given a set of genes, identify a pattern and search the genome for the pattern.

Key Components of Gene Structure:

- Untranslated Region: 5′ & 3′ regions of mRNA that are transcribed but not translated (for regulation of translation, mRNA localization, and stability.)
- Poly-A Tail: Adenisone nucleotides on 3′ end of eukaryotic mRNAs for stability, nuclear export, and greater translational efficiency.
- Intron: Non-coding Segments in coding region (removed in splicing)
- Exon: Coding sequence in mature mRNA.

Regulatory Elements in Genome Annotation:

- Transcriptional Factor Binding Site: DNA sequence where transcription factors bind. Crucial for regulating gene expression by binding or blocking transcriptional machinery.
- Promoter: Region upstream of transcription start site for RNA polymerase and associated factors to start transcrption.
- Enhancer: Can be far from gene (Works by looping DNA to interact with promoters, enhancing transcription)
- Donor (5′ Splice Site): Located at 5′ end of intron (Typically has GU dinucleotide)
- Acceptor (3′ Splice Site): Located at 3′ end of intron. (Typically has an AG dinucleotide)
- Branch Point (Adenine Nucleotide upstream of acceptor site used for lariat formation during intron removal)
- Lariat: Looping of DNA to clip Intron.

Splicing Mechanisms:

- Exon Skipping: Selective removal of 1+ exons
- Alternative 5′/3′ splice sites: Vary exact splice site
- Mutually exclusive exons: Only one exon from a pair is included in mature mRNA
- Intron Retention: As intron remains in mature mRNA, sometimes for regulatory function
- Regulatory Factors: Controlled by cis-elements [on same molecule as regulated gene] (Splicing enhancers/silencers) and trans-acting splicing factors (SR proteins, hnRNPs)

Spliceosome: Complex of small nuclear RNAs and proteins facilitating splicing

Gene Family: Set of genes originated from common evolutionary ancestor (Ex. Homology). To find what gene family a gene belongs to, a simple solution is BLAST, but it suffers from same shortcomings as gene prediction.

Open Reading Frame: Continous stretch of protein-encoding codons. Starts with ATG and ends with stop codon (TAA, TAG, TGA). There are 3 possible frames in a DNA strand, 6 in double-stranded. Note ORFs can be nested within a longer ORF, but they get blocked. Also we should exclude ORFs that are too short (300 bp)

We can predict introns by looking for a pattern: GT….A..AG

To find a gene we look for the following:

- ATG
- GT (Intron Start)
- A (Near Intron End)
- AG (Intron Stop)
- TAA/TAG/TGA (Exon Stop)

Sequence Motifs: Short, discrete sequence patterns indicating certain biologial functions.

Problems with hard-coded sequence motifs:

- They are too short, and these short sequences can occur randomly in genome.
- Exceptions (Some genes have alternative start codons, ORFS can be short)
- Errors in DNA sequencing
- Some features don't have discrete sequence motifs.

Gene Sequences are probabilistic:

- First 3 bases are 99% likely to be ATG
- First 6 bases upstream of coding region have 75% probability to be GACACC (Kozak Sequence)
- 90% prob of TATAAA sequence within 100 bp upstream of gene
- CG dimers are 50% higher frequency in 500 bp window before gene than genome avg.

11.3 Gene functional annotation

Once we've identified these structures, we can perform functional annotation (assign genes to gene families) Gene families: Set of genes that originate from common evolutionary ancestor. Given observed traits, group organisms on evolutionary relationships from these traits and have gene families depend just on gene, rather than whole genome.

11.4 Calculating Probabilistic Genome Frequency

When encountering a motif, if it is a gene, the probability of finding a relevant motif in downstream region is higher, if not, probability is lower. We can use this approach to determine if something is a gene.

Given aligned sequences, it's easy to compute a profiling matrix:

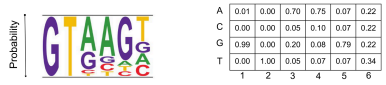
- Given a series of sequences, calculate the probability of X nucleotide at position I for all nucleotides and all positions. (ex. having nucleotide "A" at position 4)

The probability of observing sequence GTAAGC assuming it is the start of a donor site:

$$P(\text{GTAAGC}) = P(\text{G}) \times P(\text{T}|\text{G}) \times P(\text{A}|\text{GT}) \times P(\text{A}|\text{GTA}) \times P(\text{G}|\text{GTAA}) \times P(\text{C}|\text{GTAAG})$$

Assuming that every base is independent (simplification)

$$P(\text{GTAAGC}) = P(\text{G}) \times P(\text{T}|\text{G}) \times P(\text{A}|\text{GT}) \times P(\text{A}|\text{GTA}) \times P(\text{G}|\text{GTAA}) \times P(\text{C}|\text{GTAAG})$$
$$= P(\text{G}) \times P(\text{T}) \times P(\text{A}) \times P(\text{A}) \times P(\text{G}) \times P(\text{C})$$
$$= 0.99 \times 1.0 \times 0.7 \times 0.75 \times 0.79 \times 0.22 \approx \mathbf{0.09}$$



	1	2	3	4	5	6
A	0.01	0.00	0.70	0.75	0.07	0.22
C	0.00	0.00	0.05	0.10	0.07	0.22
G	0.99	0.00	0.20	0.08	0.79	0.22
T	0.00	1.00	0.05	0.07	0.07	0.34

Then, given a sequence, calculate the likelihood of getting the sequence given it is a gene (using profiling matrix). Then calculate the likelihood given the genes aligned randomly. This is the null hypothesis. Then, comparing the two likelihoods you can determine whether it is likely to be a gene or not.

Markov Model:
Markov Property: System's future state is influenced by present state, not past.

- Given base frequencies are linked to recent context, we use a Markov model to calculate probability of getting a nucleotide given the previous x nucleotides)

Assuming that every base is only dependent on the previous base

- First-order Markov model

$$P(\text{GTAAGC}) = P(\text{G}) \times P(\text{T}|\text{G}) \times P(\text{A}|\text{GT}) \times P(\text{A}|\text{GTA}) \times P(\text{G}|\text{GTAA}) \times P(\text{C}|\text{GTAAG})$$
$$= P(\text{G}) \times P(\text{T}|\text{G}) \times P(\text{A}|\text{T}) \times P(\text{A}|\text{A}) \times P(\text{G}|\text{A}) \times P(\text{C}|\text{G})$$

- Second-order Markov model

$$P(\text{GTAAGC}) = P(\text{GT}) \times P(\text{A}|\text{GT}) \times P(\text{A}|\text{TA}) \times P(\text{G}|\text{AA}) \times P(\text{C}|\text{AG})$$

A sequence of events, in which every next state is only dependent on its very previous state, is called a **Markov chain**

Markov Chain: Sequence of events, each dependent on previous state.

11.5 Simplifying Code to Gene Mask

We want to predict whether a site is "in a gene" or "non-gene", but that is a hidden state.

- If a hidden state changes between each other, that is a transition (non-gene => gene, gene => non-gene, gene stays gene)
- Hidden states can influence probability of observations (emission) [Ex. In genes we see more C and G than in non-genic regions]

Then each emission has a probability (eg. P(GI0)) and each transition has probability (P(0->1))

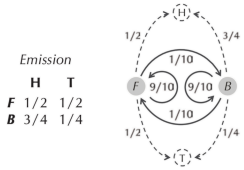
Hidden Markov Models:

1. Alphabet Σ of emitted symbols
2. Set of hidden states
3. A $|\text{States}| \times |\text{States}|$ transition matrix
4. A $|\text{States}| \times |\Sigma|$ emission matrix

For each state l, transition probabilities must sum to 1. Same with emission probabilities.

Visualization:

- Solid nodes represent hidden states. Every pair of states is connected by solid directed edge labeled with transition probability.
- Dashed nodes represent alphabet symbols. Each state is connected to every symbol node by dashed edge labeled with emission probability.



Emission

	H	T
F	1/2	1/2
B	3/4	1/4

Transmission

	F	***B***
F	9/10	1/10
B	1/10	9/10

Example: Given path $\pi = \pi_1 ... \pi_n$ and emitted string $x = x_1 ... x_n$, $\Pr(x, \pi) = \Pr(x \mid \pi) \cdot \Pr(\pi)$ and $\Pr(\pi) = \prod_{i=1}^{n} \Pr(\pi_{i-1} \to \pi_i)$. $\Pr(x|\pi) = \prod_{i=1}^{n} \Pr(x_i \mid \pi_i)$, therefore $\Pr(x, \pi) = \prod_{i=1}^{n} \text{emission}_{\pi_i}(x_i) \cdot \text{transition}_{\pi_{i-1}, \pi_i}$

Decoding Problem: Given emitted string from HMM, how do we find the hidden path that maximizes the probability.

Viterbi Graph:

- Create $|\text{States}|$ rows
- n columns (one for each symbol in x), where node (k, i) represents being in state k after emitting i^{th} symbol
- Each edge from $(l, i-1)$ to (k, i) has weight $\text{transition}_{l,k} \cdot \text{emission}_{k(x_i)}$
- Source node is created with π_0 as silent initial state, and a terminal sink node at the end with edge weight 1 for each node.

Viterbi Algorithm:

- $s_{k,i}$ is maximum product weight of path from source to node (k, i)
- Recurrence is $s_{k,i} = \max_{l \in \text{states}} \left(s_{l,i-1} \cdot \text{transition}\right)_{l,k} \cdot \text{emission}_{k(x_i)}$
- First column has $s_{\text{source}} = 1$
- $S_{\text{sink}} = \max_{k \in \text{states}} \{s_{k,n}\}$

Since maxing product is equal to maxing sum of logs, one can take logs of all weights and solve longest-path problem in resulting weighted DAG.

To find overall probability that HMM emits a particular string x, calculate Probability of emission given path for all possible paths in HMM.

HMMs can also be used for homology search, where an HMM profile is a probabilistic model including probabilities of sequence variants (nucleotides or amino acids)

12 Clustering

Good Clustering Principle: Every pair of points from same cluster should be closer to each other than any pair of points from different clusters.

Formal definition: Optimization given set of data points X, distance metrix $d(x_i, x_j)$ and number of clusters k the goal is to partition X into subsets $C = \{C_1, ..., C_k\}$ such taht objective function $\mathcal{L}(C)$ is optimized, so $C^* = \arg\min_C \mathcal{L}(C)$

12.1 K-Means

Given dataset, K-means clustering aims to partition observations such that within-cluster sum-of-squares is minimized: $\mathcal{L} = \sum_{i=1}^{K} \sum_{j=1}^{n_i} \left\| x_j^{(i)} - \mu_i \right\|^2$

Algorithm:

- Initialize centroids
- While not converged
- Update cluster assignments to closest cluster
- Update centroid

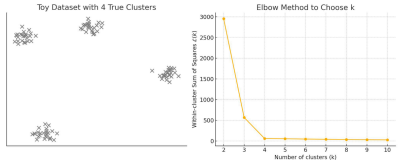
Algorithm 1 K-means algorithm to cluster $\mathbb{D} = \{\boldsymbol{x}_n\}_{n=1}^{N} \subset \mathbb{R}^d$

Require: $0 < K < N$ ▷ Expected number of clusters
$\boldsymbol{\mu}_i \sim \mathbb{D}$ ▷ Initialize centroids
while not convergence **do**
  $y_i \leftarrow \arg\min_i \left\| \boldsymbol{x}_j^{(i)} - \boldsymbol{\mu}_i \right\|^2$ ▷ Update the cluster assignments
  $\boldsymbol{\mu}_i \leftarrow \frac{1}{n_i} \sum_{j=1}^{N} \mathbf{1}[y_j = i] \cdot \boldsymbol{x}_j$ ▷ Update the centroids
end while

Robustness not guaranteed in KMeans, because of sensitivity to outliers

Elbow Method:

- Heuristic for selecting K
- Plot objective function against different values of K
- Optimal K is at point where adding another cluster results in minimal improvement.



Elbow Formulation:

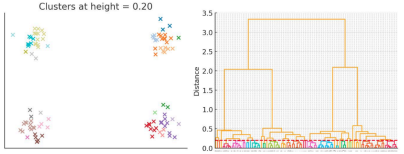
- Compute clustering for $1, 2, ..., K_{\max}$
- Plot clustering Loss vs number of clusters
- Optimal K at most noticeable "bend"

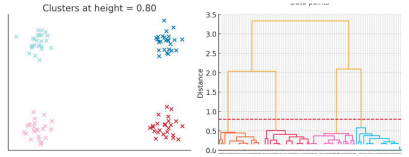
12.2 Hierarchical Clustering

- Determining clusters within clusters (ex phylogenetic class)

Dendograms:

- Horizontal cuts at different heights are different clusterings
- Higher cut (larger distance): Fewer, larger clusters
- Lower cut: More clusters, finer granularity
- Higher cut: Fewer, broader clusters





Linkage methods:

- Single Linkage: Merge clusters by minimum distance between clusters. $d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$
- Complete Linkage: Merge clusters by maximum distance between clusters. $d(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$
- Average Linkage: Merge clusters by average distance (UPGMA) $d(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y)$

12.3 Density Based Clustering

DBSCAN clusters points together based on density.

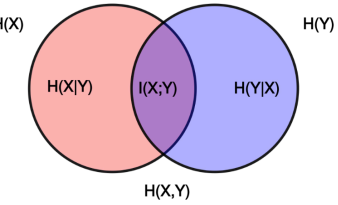
12.4 Neural Clustering

Using a deep neural network, we can learn lower-dimensional representations (then cluster the points based on those lower-dimensional representations) For instance, via autoencoder network.

12.4.1 Neural Information-Based Clustering

To train discriminative model without labels, we can do this through explicit maximization of mutual information between input and discrete output probability distribution.

Mutual Information: Measure of mutual information provided between two variables. How much you know about X given Y and vice versa.



$$\begin{aligned}
MI(y, x) &= \int\int p(y, x) \log \frac{p(x, y)}{p(x)p(y)} dydx \\
&= \int\int p(y \mid x)p(x) \log \frac{p(y \mid x)}{p(y)} dydx \\
&= \int\int p(y \mid x)p(x) \log p(y \mid x) dydx \\
&\quad - \int\int p(y \mid x)p(x) \log p(y) dydx \\
&= \mathbb{E}_{x \sim p(x)} \left[\sum_{j=1}^{K} p(y_j \mid x) \log p(y_j \mid x) \right] \\
&\quad - \mathbb{E}_{x \sim p(x)} \left[\sum_{j=1}^{K} p(y_j \mid x) \log p(y_j) \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} p(y_j \mid x_i) \log p(y_j \mid x_i) \\
&\quad - \sum_{j=1}^{K} \log p(y_i) \mathbb{E}_{x \sim p(x)} \left[p(y_j) | x \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} p(y_j \mid x_i) \log p(y_j \mid x_i) \\
&\quad - \sum_{j=1}^{K} \log \left(\frac{1}{N} \sum_{i=1}^{N} p(y_i \mid x_i) \right) \cdot \frac{1}{N} \sum_{i=1}^{N} p(y_j \mid x_i)
\end{aligned}$$

$$= -\overline{H(\sigma)} + H(\overline{\sigma})$$

Note that $p(y_j \mid x_i) = \sigma_{ij} = \varphi(x_i) = \text{softmax}\left(f_{\theta(x_i)}\right)$ So you can use the softmax outputs for this calculation.

13 RNA Sequencing

Microarray Technology: Tool to determine gene expression levels by comparing expression of specific RNA sequences in cells under different treatments.

Procedure:
1. The chip is a glass slide with thousands of tiny spots with DNA probes (short known sequences of specific genes)
2. mRNA is extracted from cells
3. mRNA is converted into cDNA (complementary DNA) and tagged with fluorescent dyes. cDNA is created by placing reverse transcriptase on mRNA transcript.
4. Labeled cDNA is washed over the chip, and whenever cDNA matches DNA probe, it will hybridize (stick to it)
5. Machine shines laser on chip, and determines gene activity by fluorescent tags.

Advantages:
1. Can focus on small regions, even few molecules/cell

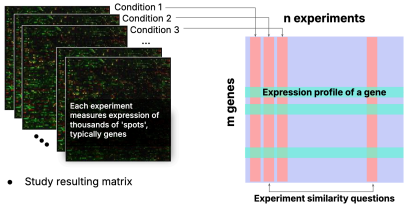
RNA-Seq Technology: More advanced version of RNA sequencing that doesn't tag to known sequences, but rather sequences actual RNA sequence.

Procedure:
1. Isolate mRNA
2. Remove DNA
3. Fragment RNA (slice into small pieces)
4. Reverse transcribe RNA into cDNA by Reverse transcriptase
5. Add ligate sequence adaptors (small tags to the end of the sequence)
6. Use PCR amplification to increase precision
7. Sequence cDNA using sequencer (like Illumina) to assign quality scores to individual reads

Expression Analysis Data Matrix: Each vector contains an expression level for each gene. These vectors are collected for each experiment/condition, and combined to form a matrix.

Each row in the matrix determines the **expression profile of a gene**.

Comparing two columns compares different experiment similarities.



13.0.1 Analysis

By **clustering** in the Expression Analysis Data Matrix, we can determine hidden relationships between genes and experiments.

By classification, we can extract features from data that best assign elements to well-defined classes.

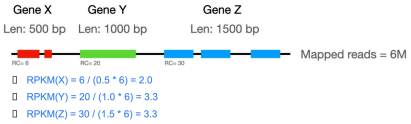
Differential Gene Expression: Genes are expressed differently depending on certain conditions (conditions can be Time, Physiology, Health, Tissue, Individual, Species)

13.0.2 Gene Expression Estimation Challenges

- Sequencing runs with more depth will have more reads mapping to each gene
- Longer Genes will have more reads mapping to them
- Normalization is necessary to do differential expression estimation

RPKM: Reads per Kilobase Per Million

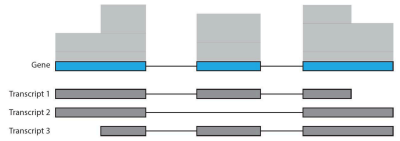
Procedure:
1. Find Read Count for gene
2. Find Total read count sum across all genes, and divide by 1 million
3. Divide gene read count by total read normalized read count
4. Divide by Transcript length divided by 1000



However RPKM is biased if a few genes are abnormally expressed, that affects normalization. Further, a single gene can have multiple isoforms (alternative splicing, different transcription start sites, etc), which means reads can map to shared exons. Then it is difficult to tell just from read expression which isoform gene was expressed most.

Instead of of finding gene expression, we can find transcript expression, and use the location of when they were expressed to find gene expression.

13.0.3 Transcript Quantification



Since each read is additive, we want to optimize weights for each transcript by minimizing:

$$\min_{w_1,\ldots,w_T \geq 0} \sum_{p \in P} \left(R_p - \sum_{t=1}^{T} w_t \cdot D_{t,p} \right)^2$$

P set of considered genomic positions. R_p: Observed read coverage (number of reads covering position p). $D_{t,p}$: Expected read coverage for transcript t at position p.

Quant: This algorithm is similar to unsupervised clustering algorithms, or EM in general, where we first optimize the transcript weights, and then optimize the profile weights repeatedly until we converge.

13.0.4 Statistical Analysis

P-Value: Probability of obtaining result at least as extreme as the one observed, given that null hypothesis is true.

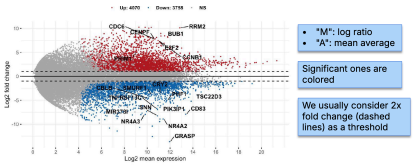
Null hypothesis H_0: Gene has same expression level in two groups

Alternative Hypothesis H_1: Gene has different expression levels

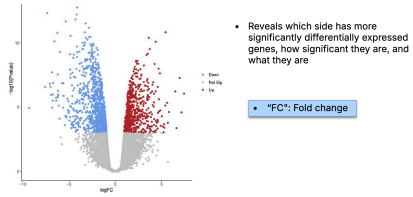
Small p-value rejects null hypothesis.

Statistical analysis of genes: We cannot run hypothesis testing on each gene individually, since we'd expect around $p\%$ to be wrong. Instead, we need to adjust p-values in multiple testing setting.

MA plot: Compare log ratio (how much gene changed between two conditions) and mean average (overall expression). Large changes given low expression ar eless significant, so MA plots reflect that uncertainty.



Volcano Plot: Compares log ratio to p-value (statistical significance of change). Ideally you'd have a large change tied to statistically significant change.



Simpson's paradox: Trend appears in groups of data but disappears when groups are combined.

14 Proteomics

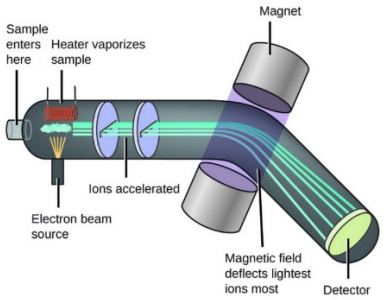
Types of protein structure:
Primary: Amino Acid Sequence
Secondary: hydrogen bonding of peptide backbone (alpha helix, beta pleated sheets)
Tertiary: 3D folding due to side chain interactions
Quaternary: Multiple Amino Acid Chains

Residue: Amino acid in a peptide chain
Peptide: Chain of amino acids
Oligopeptide: 2-20 residues
Polypeptide: 20+ residues

14.1 Main Problems

Identification: Determine which proteins are present in sample (from database) **Sequencing**: Determine amino acid sequence without database **Quantification**: determine expression level of protein under 2+ different conditions

14.2 Mass Spectrometry



Samples are ionized and then deflected by magnetic field. Final result on detector is dependent on mass-charge. Greater mass means smaller movement, greater charge means greater movement.

MS/MS (Tandem Mass Spectrometry): Two-stage mass spectrometry, that first separates protein by mass/charge ionization, and then fragments the proteins before running it through a mas spectrometry to analyze again. This seocnd stage is useful for understanding the composition of proteins.

Samples are detected by mass over charge. To differentiate between same ratios, proteins typically have multiple charge states, so you can identify proteins by the relative peaks of different charge mass/charge ratios.

G A S P V T C I L N D K Q E M H F R Y W
57 71 87 97 99 101 103 113 113 114 115 128 128 129 131 137 147 156 163 186

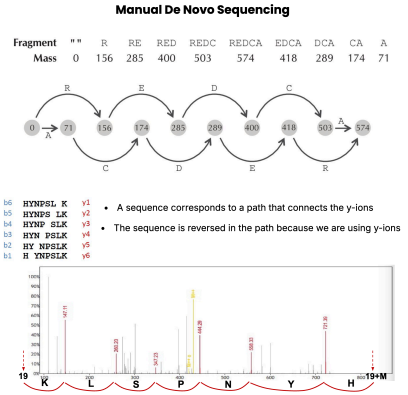
Name	3-letter code	1-letter code	Monoisotopic Mass	Average Mass	Composition
Alanine	Ala	A	71.03711	71.08	C_3H_5NO
Arginine	Arg	R	156.10111	156.2	$C_6H_{12}N_4O$
Asparagine	Asn	N	114.04293	114.1	$C_4H_6N_2O_2$
Aspartic Acid	Asp	D	115.02694	115.1	$C_4H_5NO_3$
Cysteine	Cys	C	103.00919	103.1	C_3H_5NOS
Glutamic Acid	Glu	E	129.04259	129.1	$C_5H_7NO_3$
Glutamine	Gln	Q	128.05858	128.1	$C_5H_8N_2O_2$
Glycine	Gly	G	57.02146	57.05	C_2H_3NO
Histidine	His	H	137.05891	137.1	$C_6H_7N_3O$
Isoleucine	Ile	I	113.08406	113.2	$C_6H_{11}NO$
Leucine	Leu	L	113.08406	113.2	$C_6H_{11}NO$
Lysine	Lys	K	128.09496	128.2	$C_6H_{12}N_2O$
Methionine	Met	M	131.04049	131.2	C_5H_9NOS
Phenyalanine	Phe	F	147.06841	147.2	C_9H_9NO
Proline	Pro	P	97.05276	97.12	C_5H_7NO
Serine	Ser	S	87.03203	87.08	$C_3H_5NO_2$
Threonine	Thr	T	101.04768	101.1	$C_4H_7NO_2$
Tryptophan	Trp	W	186.07931	186.2	$C_{11}H_{10}N_2O$
Tyrosine	Tyr	Y	163.06333	163.2	$C_9H_9NO_2$
Valine	Val	V	99.06841	99.13	C_5H_9NO

Note that we fragment peptides in MS, and derive the masses for each peptide chain to derive the entire sequence of the protein. For instance, given a peptide ANELLLNVK, we will fragment it into A, NELLLNVK, AN, ELLLNVK, ANE, LLLNVK, etc. We can then determine the peptide by daisy chaining in reverse.

14.3 MS Interpretations

2 ways to interpret MS/MS spectra: Search through existing protein databases or do de novo sequencing.

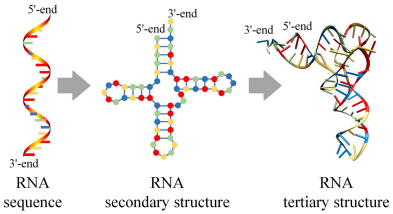
To search through the database, given a known protein sequence, use enzyme digestion rule to cut into peptides, and for each peptide compare with spectrum to see how well it matches the derived peptide fragments.



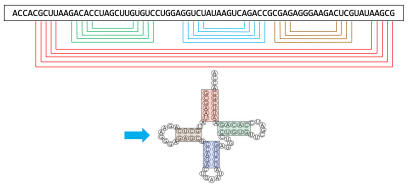
15 RNA Protein Folding

Condensed DNA is stored in a Chromatin chain. Each chromatin is a seires of nucleosomes (DNA coiled around 4 histones). In between histones are epigenetic marks (methyl) to influence gene expression.

RNA structure has three forms: Sequence (chain), Secondary Structure (loops within RNA), Tertiary structure (3D conformation)

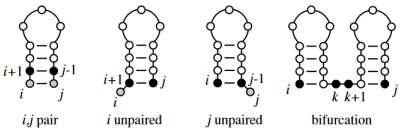


15.1 RNA Folding



Nussinov Algorithm: Construct matrix of RNA string by RNA string and initialize diagonal to 0, and lower diagonal to 0 as well. Each coordinate (i, j) in the matrix should contain the maximum score for sequence between S_i and S_j.

For each cell, we follow this structure:



$$M(i,j) = \max_{i \leq k < j} \begin{cases} M(i, k-1) + M(k+1, j-1) + \text{Score}(S_k, S_j) \\ M(i, j-1) \end{cases}$$

To traceback, simply go to top-right cell, and then find optimal structure by tracing arrows with high scores or diagonal arrows in the cell.