Sequence Alignment and Search

Christopher Lee, Matteo Pellegrini

Department of Chemistry and Biochemistry, UCLA

Announcements

- done with statistics!
- rest of the course focuses on various genomics and bioinformatics applications.
- congratulations on having climbed the two main learning curves in this class (Bayes Law, using the VM)!
- needed to get these out of the way at the beginning, as the same skills come up again and again in every genomics analysis.
- it's been a steep learning curve so far, so don't worry if it doesn't seem solid for you yet. You'll get lots of practice applying these on concrete genomics examples, so this will become "routine" for you.
- this week we focus on sequence alignment and homology search, a basic tool for a huge variety of genomics analyses.

Your Mission This Week

A dangerous bioterror agent (similar to anthrax) has been discovered in an envelope sent to your favorite senator!

- as we speak, the FBI crime lab is sequencing the genome of the organism found in the envelope, and scouring the world for all samples of bioweapon organisms to see if they can trace it.
- your mission this week is to write a computer program that can compare the new bioterror agent sequence against each known sample, score the similarity, find the differences if any and find the matching sample!

7 days to complete your Mission

By:

- **Wednesday**: complete crash course on what you need to know to solve this
- **Friday**: complete your mathematical solution of this problem.
- **Monday**: complete your analysis of the crime lab sequence data, by programming your solution and running it to align and score any pair of input sequences.

Mission Training

- Reading: Jones & Pevzner 6.1 - 6.9.
- Detailed project description uploaded to CCLE today. Implement "dynamic programming" (Viterbi) sequence alignment of a pair of sequences.
- Some example sequence data for testing your programs will be uploaded to CCLE.
- Please complete conceptual exercises training you on the theory details for how to implement your program by **Friday** discussion section (on Courselets).
- Today we drill down into your understanding of some details of the forward-backward algorithm, then apply the Viterbi algorithm to pairwise sequence alignment, first in HMM terms ("profile alignment") and second in standard "alignment matrix" terms.

Mini-language: Sequence Alignment Table

A **sequence alignment table** is a table showing how letters of different sequences are matched to each other, in which the rows represent individual sequences, and each column represents aligned letters inferred to be descended from a single letter in a common ancestor. For example for the sequences `TGA` and `GAT`, the alignment table:

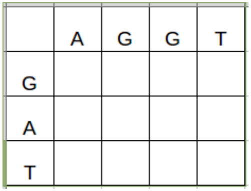
```
TGA-
-GAT
```

implies that the Gs are aligned, the As are aligned, but not the Ts. The dash character - represents a **gap** representing a deletion in one sequence of a letter that is present in the other sequence.

Mini-language: Sequence Alignment Matrix

A **sequence alignment matrix** is a table of cells that represents the space of all possible alignments of two sequences X and Y. By convention its horizontal dimension represents the consecutive letters of sequence X (i.e. its first column corresponds to letter X_1, and column n represents letter X_n), and its vertical dimension represents the consecutive letters of sequence Y (i.e. its first row corresponds to letter Y_1, and row m represents letter Y_m).

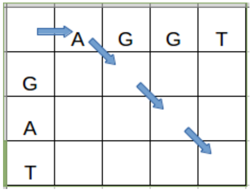
For example, for sequence `X=AGGT` and sequence `Y=GAT`, the matrix would look like:



	A	G	G	T
G				
A				
T				

Mini-language: Sequence Alignment Path

An **alignment path** is a sequence of moves from cell to cell in the matrix, that are either *diagonal* (aligning the two letters in the destination cell), *horizontal* (skipping that letter in the X sequence), or *vertical* (skipping that letter in the Y sequence).
E.g. the following alignment path skips the initial letter A, then aligns G with G, G with A, and T with T:



By convention, a horizontal move is symbolized by the *alignment operator* **x**, a vertical move by the alignment operator **y**, and a diagonal move by the alignment operator **m**.

Optimal Subpath?

An **optimal subpath** is the best possible (highest scoring) alignment path up to some specified cell in the alignment matrix. For example, for the pair of sequences `X=GAT` and `Y=AAG`, what is the optimal subpath up to cell (1,1)? Show all possible incoming moves into this cell, and the score for each subpath to this cell. Assume match=4, mismatch=-2, gap=-3.

Optimal Subpath? Answer

The three incoming moves to (1,1) are **m**, **x**, and **y**.
The possible subpaths to (1,1) are **m** (score -2), **xy** (-6), and **yx** (-6).
So **m** is the optimal subpath to (1, 1).

Simple Alignment with Substitution?

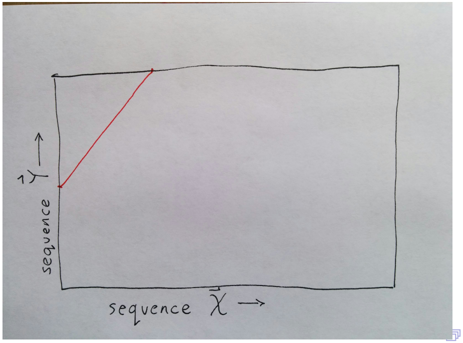
Say you want to find the best scoring alignment of two sequences, under the assumption that the only mutation that could occur within the homologous region is **substitution** of one letter by another (but this homologous region might be just a part of either sequence). Assume furthermore that you are given the scoring function $S(X_t, Y_u)$ that gives the correct log-odds score for the possible letter substitutions. E.g. given sequences (not aligned!):

```
DRGCGMMHDHOQMQLNOFWPLQVWCOERNG
GKOYWRNWTKOIWKCIEVSSWRDGVQQRRADMONHHDDPE
```

Propose an algorithm for finding the best-scoring alignment, and some measure of how the computational "work" for doing it will increase as a function of the average length L of the sequences.

Simple Alignment with Substitution? Answer

- We can "slide" one sequence letter by letter against the other sequence, and sum the scores for the aligned letters for a given "alignment offset". We then report the alignment with the highest score.
- This will require scoring every letter of the first sequence against every letter of the second sequence. If each sequence has L letters, then this will take $O(L^2)$ operations.



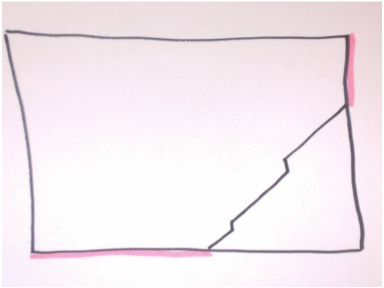
Alignment with Insertions and Deletions?

Say we expand the possible mutations to include not only single-letter substitution, but also **single-letter insertion/deletion** ("indel": an insertion in one sequence looks the same as a deletion in the other sequence; also known as "gaps").

- how would each of these mutations appear on the "alignment path" picture of sequence X vs. sequence Y we drew?
- how much harder do indels make the problem of evaluating *all possible alignments* of a pair of sequences of length L? Again, propose some measure of how the computational "work" for doing this (with gaps) will increase as a function of the average length L of the sequences.

Alignment with Insertions and Deletions? Answer

- Substitutions just move the alignment path diagonally (+1,+1) as we drew before.
- Indels move the alignment path either horizontally $\rightarrow$ (+1,0) or vertically $\uparrow$ (0,+1).



- We can gauge the computational work by the number of alignment paths we would have to compute. We can think of aligning a sequence of length L as (approximately) a sequence of three possible "moves" ($\nearrow, \rightarrow, \uparrow$), i.e. $O(3^L)$. Much harder!!

Seeking an Efficient Algorithm for Alignment with Gaps

- We need an algorithm that will make alignment much more efficient than the "exhaustive enumeration" limit!
- Exhaustive enumeration is necessary when the only way to tell whether a given alignment is the best is to compare it against *every* other possible alignment. This is called a "global optimality" condition i.e. $S(\alpha^*) \geq S(\alpha)$ for all possible alignments α.
- If we could find a way to translate this *global optimality* rule into a *local optimality* rule (i.e. we could tell whether *part* of an alignment path could be optimal or not), we could enormously accelerate the search.
- It turns out that our simple **additive** scoring function (i.e. that the total score is just the sum of separate scores; the score for one part of an alignment **does not depend** on what we choose for other parts of the alignment), enables us to do this!

Optimal Subpath Rule?

If you were given the *optimal subpath scores* to cells *(t-1,u)*, *(t-1,u-1)* and *(t,u-1)*, (which we will write as $S^*(t-1,u), S^*(t-1,u-1), S^*(t,u-1)$), can you suggest a rule for choosing the optimal path score to cell *(t,u)*, i.e. $S^*(t,u)$? As usual, assume the substitution score is *s(L1,L2)* and the gap penalty is *g*.

Optimal Subpath Rule? Answer

The only possible incoming edges to cell *(t,u)* are:

- move **m** from cell *(t-1,u-1)*;
- move **x** from cell *(t-1,u)*;
- move **y** from cell *(t,u-1)*;

Hence, if we know the optimal (maximum score) subpath to each of those three cells, we could find the optimal path to cell *(t,u)* by seeing which of those incoming edges has the highest total score, i.e.

$$\begin{aligned} S^*(t,u) = \max\{ & S^*(t-1,u-1) + s(X_t, Y_u), \\ & S^*(t-1,u) + g, \\ & S^*(t,u-1) + g\} \end{aligned}$$

Optimal Subpath Rule?

Could a **suboptimal** subpath to *(t-1,u-1)* be part of the **optimal** subpath to *(t,u)*? In other words, consider a suboptimal subpath P ending at *(t-1,u-1)* for which

$$S(P) < S(P^*)$$

where P^* is the optimal subpath ending at *(t-1,u-1)*. Could the path $P+$ **m** be the optimal subpath ending at *(t,u)*? If so, briefly explain why with an example; if not, why not.

Optimal Subpath Rule? Answer

No, this is not possible. The total score for this path will be

$$S(P+m) = S(P) + s(X_t, Y_u)$$

whereas the total score using path P^* will be

$$S(P^*+m) = S(P^*) + s(X_t, Y_u) > S(P) + s(X_t, Y_u)$$

This is because under our scoring rules the score for move **m** ($s(X_t, Y_u)$) will be the same regardless of whether we take subpath P or subpath P^*.

Dynamic Programming Viterbi Algorithm

Consider the following simple rule for finding the optimal subpath to a cell $P^*(t,u)$, assuming you know the optimal subpaths to its *predecessor cells* $P^*(t-1,u-1)$, $P^*(t-1,u)$, $P^*(t,u-1)$:

- calculate the total scores for the three subpaths passing through each predecessor, i.e.
  $S^*(t-1,u-1)+S(m)$,
  $S^*(t-1,u)+S(x)$,
  $S^*(t,u-1)+S(y)$
- choose the highest scoring of these subpaths as the optimal subpath $P^*(t,u)$ to cell (t,u).

A Simple Alignment Optimality Rule

- We want to find the optimal (highest scoring) alignment path $\alpha^*(X_t, Y_u)$ to endpoint (X_t, Y_u).
- Let π represent one of the three possible *predecessor* endpoints (X_{t-1}, Y_u), (X_t, Y_{u-1}), (X_{t-1}, Y_{u-1}).
- Assume the total score for a path is the *sum* of its subpaths, i.e. $S(\alpha(X_t, Y_u)) = S(\alpha(\pi)) + S(\pi \to (X_t, Y_u))$, where the last term is just the substitution score or gap penalty for that move.
- Say we *know* the optimal path to π: $S(\alpha^*(\pi)) \geq S(\alpha(\pi))\ \forall \alpha$.
- Now set $\alpha^*(X_t, Y_u)$ based on the π^* that maximizes the score:

$$S(\alpha^*(X_t, Y_u)) = \max_{\pi} [S(\alpha^*(\pi)) + S(\pi \to (X_t, Y_u))]$$

$$\geq S(\alpha^*(\pi)) + S(\pi \to (X_t, Y_u))$$

$$\geq S(\alpha(\pi)) + S(\pi \to (X_t, Y_u)) = S(\alpha(X_t, Y_u))\ \forall \alpha$$

Dynamic Programming Alignment Algorithm

- This maximization rule is sometimes called the **Viterbi Algorithm**, and its computational implementation is called **dynamic programming**.
- Since we can get $\alpha^*(X_1, Y_1)$ trivially, we can get $\alpha^*(X_2, Y_1)$ $\alpha^*(X_1, Y_2)$ etc. etc.
- we have to do this maximization at *every possible endpoint* (X_t, Y_u); this is referred to as "filling the alignment matrix".
- To complete it for a given endpoint, we just need to have completed it for its three *predecessor* endpoints.
- At each endpoint we write the **score** of the best path to that endpoint, and an **arrow** pointing to that best path (i.e. the predecessor that gave the maximum score).
- To get the full alignment path to a given endpoint, just *follow the arrows* (from that endpoint).

Viterbi Alignment Demo

Let's work through a simple example of filling out the "Viterbi alignment matrix" for aligning **CAT** vs. **ATA**, using the following scoring:

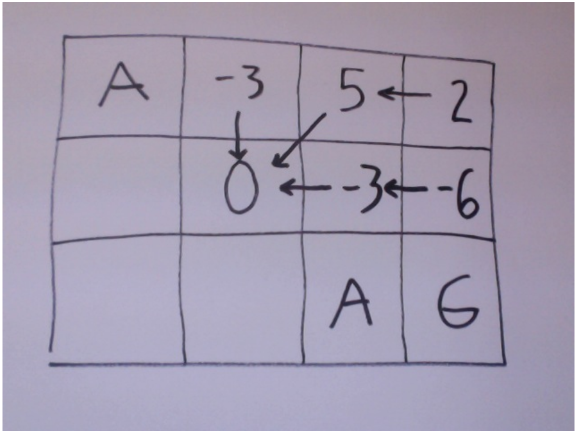
- exact match: +5
- substitution: -2
- gap (indel): -3

Let's leave an empty row before each sequence as the "origin" for the "alignment moves" for the first letter of each sequence. We explicitly write a zero score in this origin cell.

Viterbi Alignment

- Given the two sequences $\vec{X} = AGC\ldots$ and $\vec{Y} = ACC\ldots$
- Find the optimal global alignment of $\vec{X}^2$ to $\vec{Y}^1$ by filling out the corresponding Viterbi alignment matrix.
- Assume the following scores:
  - exact match: +5
  - substitution: -2
  - gap: -3

Viterbi Alignment Answer



Optimal Global Alignment:

```
AG
|
A-
```

Viterbi Alignment Errors

- Some people just copied the substitution scores for the letters X_t, Y_u into each cell. Remember that Viterbi considers all possible moves $\leftarrow, \swarrow, \downarrow$, and chooses the best one.

Announcements

- Instructions for the graduate term project will be posted today.
- I have office hour after class today.
- Since this Friday is Veteran's Day holiday, no discussion section. Please ask your project questions on the Stepik step it is about!

Project: How Impossible is This Mission?

- how hard is it to compare two sequences *without* insertions/deletions just using a naive algorithm?
- how much harder *with* insertions/deletions?
- is the computational complexity going to kill your mission?!?
- can you cut this down to size using a clever Viterbi implementation?
- for the answers, tune in to the initial training materials on Courselets (see link in CCLE Week 7)!

Simplest Possible Type of Alignment

- *global* alignment is defined as a path that begins from the origin (0,0) of the alignment matrix and ends at the opposite corner *(m,n)* of the matrix.
- means that it applies a gap penalty to *any* letters at the beginning (or end) of one sequence that are *not* aligned to the other sequence. I.e. the "scoring path" must "consume" *all* the letters of *both* sequences.
- Simplifies the alignment algorithm: we already know where the path must begin and end.
- We can also simplify gap scoring by asserting that any indel has the same score, regardless of whether it was preceded by an indel or not. That means we don’t need to keep separate matrices for the **I** and **D** states; we can just record the Viterbi maximum for all three moves on a *single matrix*.

Global Alignment Assembly

Say we are trying to assemble a large dataset of short read sequences from next-gen sequencing of a genome. Specifically, we align a candidate pair of read sequences using global alignment, and report them as aligned if the score is greater than zero.

- is this a good strategy for robustly detecting real alignments between reads, in terms of the false negative rate? I.e. is it likely to miss some real alignments?

Global Alignment Assembly Answer

- Global alignment assumes that the two sequences are entirely derived from a common ancestor, via fixed probabilities of single letter substitution or indel.
- But that is not the case for short reads: they each cover a random interval in the genome, so even if two reads overlap, on average only about half of one read will overlap the other. But global alignment will consider that an exponentially unlikely mutation event (a long sequence of single-letter indels).
- Concretely, this will result in a high false negative rate, because many read pairs with real alignments will get a negative score due to global alignment gap penalties for the non-overlapping ends.
- Strictly speaking, global alignment would only be appropriate for a pair of reads generated from the same genomic interval.

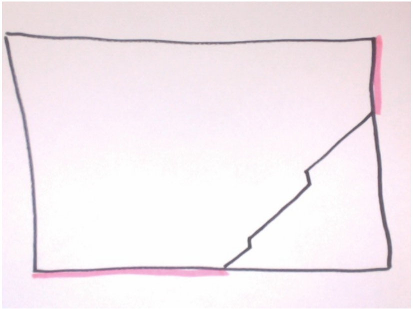
Assembly Alignment

Propose an alignment method appropriate for finding short read pairs with real alignment. In other words, where the alignment score is an appropriate guide to whether the pair of reads actually overlap (cover a shared genomic interval).

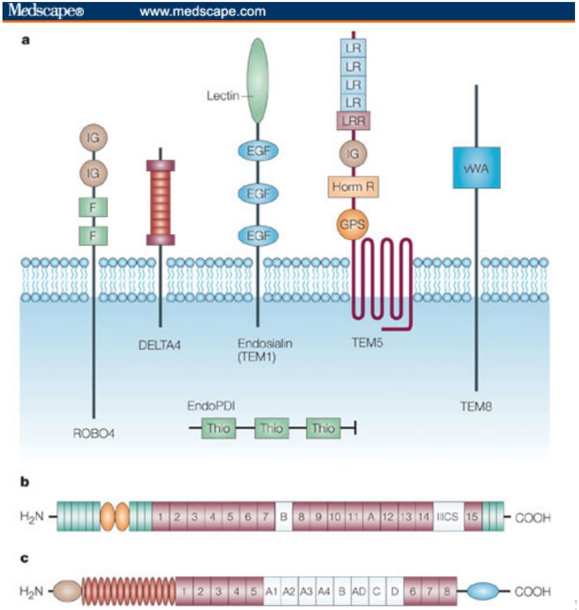
- focus on the question of where the alignment path should begin and end.
- you do not need to consider the question of how to derive the scoring parameters. Assume that appropriate parameters for short read assembly are already provided.

Assembly Alignment Answer

- overlap of two fragments of the genomic sequence should *not* charge gap penalties for the non-overlapping ends.
- thus the alignment scoring should only be applied over a path from the *beginning* of one sequence to the *end* of the other sequence (i.e. the portion that overlaps). I.e. from one edge of the matrix to the adjacent edge.



Multidomain Protein Architecture



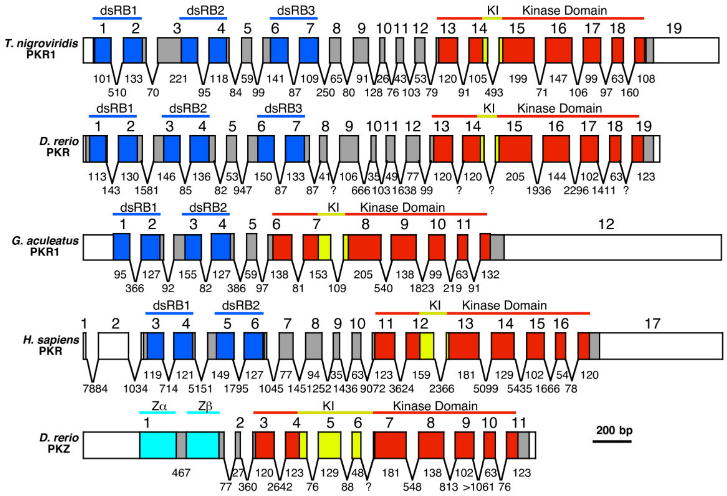
Domain Combinations

- Plant and animal protein architectures tend to contain multiple functional domains chained together in complex combinations.
- Each domain sequence is typically a member of a large family of related domain sequences found in different proteins (and in many cases multiple "copies" of the same domain in one protein).
- Discovering new domain types (families) and elucidating the domain architecture of all proteins are important research areas.

Local Alignment

- opposite extreme from global alignment: let the path begin and end *anywhere* in the matrix.
- specifically, introduce a new move "START" that is considered at each cell, and which simply has a total score of zero. I.e. *never* choose any move that results in a negative score -- START will always be preferred.
- to determine the Viterbi end point: after completing the matrix, simply search for the cell with the highest score. It is the endpoint of the optimal local alignment; backtrack as usual until you encounter a START move.
- Appropriate for finding the maximum *subinterval* of a pair of sequences that is convincingly homologous (under the mutation model provided by our scoring parameters).

Complex Gene Structure



Searching for regulatory sequences in introns

- You are studying introns in mammalian genes, looking for regulatory elements in orthologous introns.
- One important indicator of regulatory element function is *selection pressure against mutation*, i.e. finding regions of aligned introns from different mammals, that are mutated much less than the usual.
- This requires **accurate** intron alignments.
- You know that introns are poorly conserved and consequently hard to align accurately.

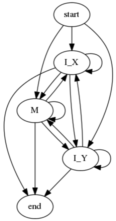
Intron Alignment Idea

- You have a database of matching *exons* in human and mouse (i.e. pairs of coding region segments in the human genome and mouse genome that align to each other optimally).
- A colleague suggests you use this matching exon information to find optimal alignments of matching introns as follows:
- take two consecutive human exons A, B (separated by an intron I) that are known to align to two consecutive mouse exons A’, B’ (separated by an intron I’).
- Now find the top-scoring alignment of the two introns I, I’ subject to the constraint that you already know the correct alignment of the exons on either side of the intron.

Intron Alignment

Assume that you know exactly the positions that are aligned at the end of exons A, A’ and at the beginning of exons B, B’. In other words, you know the letters immediately before the *first* letters of I, I’ are aligned, and you know the letters immediately after the *last* letters of I, I’ are aligned. Now you are given the sequences of introns I, I’ to align.

- What would be the easiest way to constrain our alignment paths to these start & end points, using our standard homogeneous 3-state alignment HMM?



- State the proposed constraint in precise terms of the allowed paths on the alignment graph.

Intron Alignment Answer

- The constraints are that letters X_0, Y_0 are aligned, and letters X_{m+1}, Y_{n+1} are aligned.
- We enforce the first constraint by only allowing transitions from the `start` state $\rightarrow s_i@X_1, Y_1$ (just like transitioning from $M@X_0, Y_0$).
- We enforce the second constraint by only allowing transitions to the `end` state from valid predecessors of $end@X_{m+1}, Y_{n+1}$, e.g. $M@X_m, Y_n$.
- This corresponds to a path that must begin at the origin corner of the alignment matrix and end at the opposite corner of the matrix.

This is called **global alignment**.

Global Viterbi Implementation

Describe how you would implement this as a Viterbi matrix algorithm:

- How does this constraint affect the *initialization* of the matrix?
- What moves are allowed during the algorithm?
- How does this constraint affect the *backtracking* stage of the algorithm?

Global Viterbi Implementation Answer

- we initialize the initial row and column of the alignment to zero (0).
- the three allowed moves are left, down, and diagonal.
- the backtracking *must* begin at the "end corner" (M,N).

How to Assess a Sequence Alignment?

Say somebody claims that two sequences are homologous (evolved from the same common ancestor) and proposes an **alignment** (match-up of the sequence letters that descended from the same ancestral nucleotide):

```
CATACTAGTAATCTGAAGTTGGAAAGAGGGCGAATCTCCT
CACACCGGGGTTGCGCAGCGGCGCTTATTTGGTCCCGTTC
```

How would you evaluate this claim? I.e. By what principles would you "score" the proposed alignment?

How to Assess a Sequence Alignment? Answer

- define some kind of *match metric*: for DNA, nucleotide identity is an obvious measure:

```
CATACTAGTAATCTGAAGTTGGAAAGAGGGCGAATCTCCT
|| ||  |   |  | ||  |     |    |   |
CACACCGGGGTTGCGCAGCGGCGCTTATTTGGTCCCGTTC
```

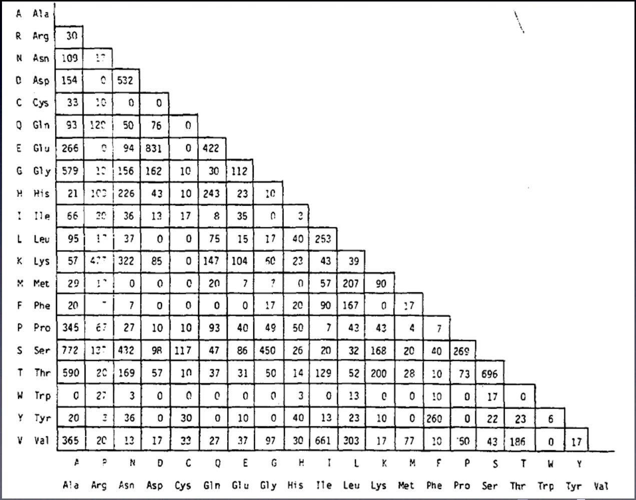
  13/40 =32.5% identity.

- Assess that relative to the *expected* value under "not-related" model (as our null hypothesis). E.g. For four equally likely nucleotides, we expect 25% identity just by random chance.
- Not very convincing. (Within this simplistic model, we could compute a p-value for getting the observed level of identity under our null hypothesis).

Amino Acid Substitutions

- Amino acid substitutions are not all identical
- Certain amino acids are more likely to substitute others
- e.g. hydrophobic amino aids tend to replace other hydrophobic amino acids
- How do we come up with more sophisticated substitution matrices?

Real Data



		A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A	Ala																				
R	Arg	30																			
N	Asn	109	17																		
D	Asp	154	0	532																	
C	Cys	33	10	0	0																
Q	Gln	93	120	50	76	0															
E	Glu	266	0	94	831	0	422														
G	Gly	579	10	156	162	10	30	112													
H	His	21	103	226	43	10	243	23	10												
I	Ile	66	20	36	13	17	8	35	0	3											
L	Leu	95	17	37	0	0	75	15	17	40	253										
K	Lys	57	477	322	85	0	147	104	60	23	43	39									
M	Met	29	17	0	0	0	20	7	7	0	57	207	90								
F	Phe	20	7	7	0	0	0	0	17	20	90	167	0	17							
P	Pro	345	67	27	10	10	93	40	49	50	7	43	43	4	7						
S	Ser	772	137	432	98	117	47	86	450	26	20	32	168	20	40	269					
T	Thr	590	20	169	57	10	37	31	50	14	129	52	200	28	10	73	696				
W	Trp	0	27	3	0	0	0	0	0	3	0	13	0	0	10	0	17	0			
Y	Tyr	20	3	36	0	30	0	10	0	40	13	23	10	0	260	0	22	23	6		
V	Val	365	20	13	17	33	27	37	97	30	661	303	17	77	10	50	43	186	0	17	

Alignment Scoring as a Hypothesis Test Problem

How can we best assess whether a given alignment (e.g. obtained from BLAST) is indicative of a real homology (i.e. two sequences actually descended from a common ancestor)?

- clearly should take into account the similarity observed between the two aligned sequences.
- clearly should also take into account whether that similarity is likely under a non-homologous model (e.g. aligning a pair of unrelated sequences).
- ideally, should reflect the actual probabilities of the the observed letters X_t, Y_u.
- can we formulate this using the hypothesis testing framework we learned in class?

Alignment Scoring

- Say you are given the likelihood $p(X_t, Y_u, M|\text{related})$ derived from "true alignments" of sequences that are known to be evolutionarily related to each other (i.e. descended from a common ancestor via mutation).
- Based on the probabilistic principles we have learned, propose a scoring function for assessing whether X_t, Y_u should be aligned (i.e. are evolutionarily **related**) to each other or not (i.e. are **unrelated**).
- How does your scoring function indicate they *should* be aligned vs. *should not* be aligned?

Alignment Scoring Answer

- Treat this as a hypothesis testing problem, where our hypothesis is the two positions descended (via mutation) from a common ancestor. We are given their likelihood under that model.
- So what's our NULL (competing) model? Propose that they are unrelated -- so just use independent likelihood model

$$p(X_t, Y_u|\text{unrelated}) = p(X_t)p(Y_u)$$

- Then we can formulate scoring function for these two competing hypotheses as simply their odds ratio:

$$\frac{p(X_t, Y_u, M|\text{related})}{p(X_t)p(Y_u)}$$

- an odds-ratio of greater than 1 favors *related*; odds-ratio less than 1 favors *unrelated*.

Alignment Scoring Errors

- Some people didn't think of this in probability terms, though the question emphasized that very clearly. Just remember that "hypothesis testing" should generally use either an odds ratio or a p-value.
- Some people proposed a p-value test, which is valid but would ignore the information $p(X_t, Y_u, M|\text{related})$ that the question asked you to take into account.
- Some people proposed the odds ratio test but were unsure what to use for the "unrelated" model. As in many previous problems we've worked, independence is the standard model that two variables are unrelated.

Alignment Log-Odds Scoring

To make the Viterbi algorithm on sequence alignment even easier, we typically use *log-odds scoring*, i.e. the score for aligning X_t, Y_u is

$$S(X_t, Y_u, M) = \log \frac{p(X_t, Y_u, M|\text{related})}{p(X_t)p(Y_u)}$$

- positive scores favor *related*; negative scores favor *unrelated*.

Then instead of the Viterbi computing the *product* of probabilities, we simply *add* the score for a new move to the total score for the previous Viterbi subpath. I.e.

$$V(M@X_t, Y_u) = S(X_t, Y_u, M) + \max_{\pi} V(\pi_{M@X_t, Y_u})$$

Blocks Substitution Matrices (BLOSUM)

- Like PAM, a log-odds scoring matrix.
- But instead of extrapolating from a single mutability matrix like PAM, to compute log-odds for different distances (e.g. PAM250), BLOSUM directly measures the mutation rates at different distances.
- e.g. BLOSUM65 measured between proteins that are 65% identical.

Why Different Matrices for different distances?

- 1% mutability data mostly represents neutral mutations in *orthologs* (i.e. those not culled by negative selection).
- 20% identity homologies mostly represent distantly related *paralogs*: only preserving the overall protein fold and maybe a few key catalytic sites. Not the same as applying the PAM1 matrix 80 times!

Insertion/Deletion Log Odds Scores?

Alignments of *related* sequence regions (descended from a common ancestor) should generally have a much lower fraction of insertions/deletions than of aligned letters (i.e. most of the letters should align). What contribution to the log-odds score would you expect passing through an insertion state to make?

Insertion/Deletion Log Odds Scores? Answer

We can think of this log odds ratio in terms of two components:

- *emission probability*: both the *related* and *unrelated* models would assign the same likelihood to the sequence letter, e.g.

$$p(X_t|I_X, related) = p(X_t|unrelated) = p(X_t)$$

  Hence the emission factors just cancel in the odds ratio and contribute nothing to the log odds score.
- *transition probability*: the question told us that $p(s_i = I_X|s_j, related) \ll 1$. By contrast no transition probability appears in the *unrelated* model (you can think of *unrelated* as a single-state HMM that transitions to itself with 100% probability). Hence transition will contribute a negative term to the log-odds score.

Overall, the log-odds score for an insertion state will be *negative*. This makes sense. Too many insertions/deletions is a sign of low sequence similarity.

Affine Gap Penalties

- "simple" gap penalty charges same penalty for any indel of one letter, regardless of what came before it.
- By contrast, "affine" gap penalty charges different penalty if the previous move was a "match move" (this is referred to as the "gap opening" penalty), versus if the previous move was also an indel ("gap extension" penalty).
- Typically there is a bigger penalty for gap opening than gap extension.
- Consequence: a single matrix is no longer sufficient for performing Viterbi alignment. Instead, we must go back to keeping separate matrices for the **M**, **D** and **I** "states".
- Concretely, "gap opening" is a transition from the **M** matrix to either the **D** or **I** matrices; "gap extension" is a move within the **D** matrix or within the **I** matrix.

BLAST

- Basic Local Alignment Sequence Tool
- Uses shortcut to compute alignments of a sequence against a database very quickly
- Typically takes about a minute to align a sequence against a database of one million sequences
- Accurately computes the statistical significance of the alignment

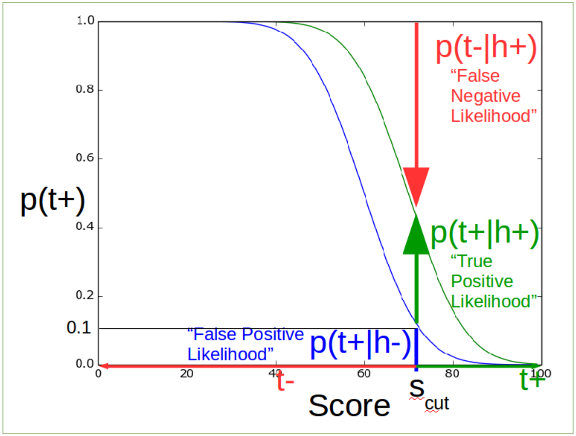
Alignment Score Hypothesis Test vs. Database?

Say we compute the optimal alignment score (using standard log-odds scoring) for a pair of sequences X, Y to be S_{obs}. Now we wish to assess this versus the null hypothesis h^- that sequence Y is an unrelated random sequence.

- draw our standard graph showing the false positive likelihood for obtaining this result under the null hypothesis.
- if we perform *multiple tests* by aligning sequence X against N sequences in a database, what is the expected number of false positives under the null hypothesis?

Alignment Score Hypothesis Test vs. Database? Answer

We graph the *survival curve* of the false positive likelihood $p(S \geq S_{obs}|h^-)$:



By the Bonferroni multiple test criterion, the expected number of false positives in N trials is $\beta = Np(S \geq S_{obs}|h^-)$.

BLAST Expectation Scores

BLAST "expectation scores" ("E value") are simply the expected number of false positives for our actual database search:

$$\beta = Np(S \geq S_{obs}|h^{-})$$

- takes into account both "how good a score" S_{obs} we got, and the size of the database we searched.
- computed in BLAST using a carefully parametrized theoretical model called the *extreme value distribution*.

Empirical False Positive Distribution

We can sample the empirical distribution of alignment scores by *simulating* the null hypothesis. Typically we take our null hypothesis to be

- **sequence** is randomized
- **composition** is unchanged. For example, searching a genome with high GC content, a random sequence with high GC content will automatically score higher. We do not want to be tricked by similar composition into thinking we observed **sequence homology**.

Therefore we can simulate the null hypothesis by **shuffling** our sequence (maintains composition, while randomizing sequence). For example, we can generate a sample of 1000 trials of the following:

- shuffle sequence X
- run the same database search on our shuffled sequence

Then β is just the number of hits with $S \geq S_{obs}$ in our sample, divided by 1000.

Scoring Problem: "Overfitting"

- our log-odds score is a valid hypothesis test when applied to a *single, predicted alignment*.
- but we are using it to *search* a vast database for maximum alignment score; subsequently using that maximized score as a *hypothesis test* is circular logic!
- another way of viewing this problem is as a case of *multiple testing*, i.e. we need to estimate the number of hits that we would expect to get this high a score purely by random chance when searching this big of a database (analogous to the Bonferoni correction).
- so we must *correct* for this maximization
- empirically, by shuffling and recomputing the score.
- theoretically, by the extreme value distribution (BLAST e-score).

BLAST Parameters

- **Identities** - **No. & % exact residue matches**
- **Positives** - **No. and % similar & ID matches**
- **Gaps** - **No. & % gaps introduced**
- **Score** - **Summed HSP score (S)**
- **Bit Score** - **a normalized score (S')**
- **Expect (E)** - **Expected # of chance HSP aligns**
- **P** - **Probability of getting a score > X**
- **T** - **Minimum word or k-tuple score** **(Threshold)**

BLAST - Rules of Thumb

- Expect (E-value) is equal to the number of BLAST alignments with a given Score that are expected to be seen simply due to chance
- Don’t trust a BLAST alignment with an Expect score > 0.01 (Grey zone is between 0.01 - 1)
- Expect and Score are related, but Expect contains more information. Note that %Identies is more useful than the bit Score
- Recall Doolittle’s Curve (%ID vs. Length, next slide) %ID > 30 - numres/50
- If uncertain about a hit, perform a PSI-BLAST search

Definitions

- Maximum Segment Pair (MSP): highest scoring pair of identical length segments from 2 sequences
- Word pair: segment pair of fixed length w
- T: minimum word pair score

Algorithm

- Pre-compute all words that score above T against the query sequence
- Search for these words in a database
- For each match, extend the alignment to generate the maximum segment pair
- Compute statistics of the resulting score

Step 1 : break up sequence into words

- ATCGTCTATTCCCGG
- w=4 letter words
  - ATCG
  - TCGT
  - CGTC
  - etc.

Step 2: find high scoring matches

- Start with word 1: ATCG
- score identities as +1, and mismatches a 0
- find all words that have a score of at least T=3
  - ATCA
  - ATCT
  - ATCC
  - etc.

Search for high scoring words in database

- Find all sequences in the database that contain a high scoring word
- For example:
- CCGGCT**ATCA**TCTATTCCCGGTTCG

Extend ALignments Around matching word

- CCGGCT**ATCA**TCTATTCCCGGTTCG
- **ATCG**TCTATTCCCGG
- The Yellow seqeunce alignmnet corresponds to the MSP wth score S

BLAST assessment

Explain the basic metric that BLAST provides for judging whether a given reported hit is likely to be a real homology. Be careful to define precisely what this metric estimates.

BLAST assessment Answer

BLAST reports an alignment score and also an E-value that estimates the statistical significance of that score. Specifically, the E-value estimates the number of hits from searching this database that would be expected by random chance with at least that alignment score. Note that the E-value can be bigger than 1 because it estimates the *number* of hits, not a probability. An E-value less than 0.01 is generally considered necessary for statistical significance.