Lecture 6

Gidon Rosalki

2025-11-04

1 Heuristics for Alignment

Often, the solution can be very expensive. Consider finding the subsequences in a sequence of length that is of the order of trillions. The proven best solution will still take a long time, and require large amounts of memory. A *heuristic* solution is not the provably best solution, but should provide a good enough solution, in much smaller amounts of required processing power. A heuristic target is a target, that is smaller than our overall objective, which by reaching these targets, we can often wind up with a "good enough" solution.

Let us consider some heuristic targets for aligning DNA:

1. Many non gapped stretches
2. Islands of identity

Consider, the two sequences s, t, of lengths n, m respectively. We want to check which words of length k appear in both, as efficiently as we can. There are lots of beautiful solutions in $O(n \log(n))$ from algorithms, but we could also simply create a hash table of the words. We can therefore check if any word appeared in the other sequence in $O(n)$ (yes, there are more problems, such as other collisions, but those are details, rather than the idea).

Consider the FASTA method, looking to heuristically match alignments of queries, vs the database. It follows the heuristic: "Good alignments have exact small matches". It uses the following algorithm:

1. Find promising matches (focusing on the top diagonals)
2. Re-score (using PAM), keeping the top scoring segments
3. Join segments using gaps
4. Finalise with DP, which is quick since most the work is already complete

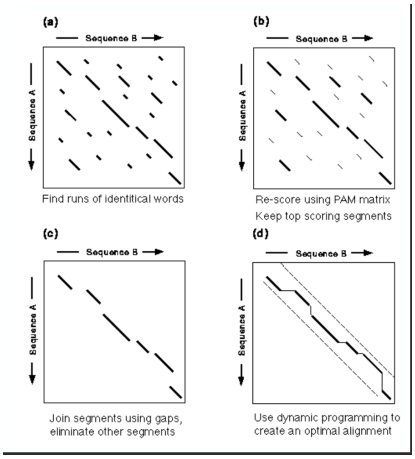


Figure 1: FASTA

However, it is plausible that we might have many islands of identity, but all with a low score, for example, if we have some very rare words, but having them align is much more meaningful. This would be missed by FASTA, and can result in poor alignments. If we instead change our heuristics to

1. Many non gapped stretches
2. High score

To resolve this, we have the BLAST algorithm. It passes over all the words in the sequence, and scores all the words with how worthwhile them aligning would be to the overall alignment. It instead follows the heuristic "Good alignments have multiple small hits" as follows:

1. Break query into words
2. Filter low scoring words
3. Store in hash table
4. Run other sequences through the hash table (allowing for mismatches)
5. Find high scoring pairs (HSPs)
6. Extend the sequence

2 Markov chains

Enables the Markov property

$$\mathbb{P}\left(X_{i+1}|X_0,\ldots,X_i\right) = \mathbb{P}\left(X_{i+1}|X_i\right)$$

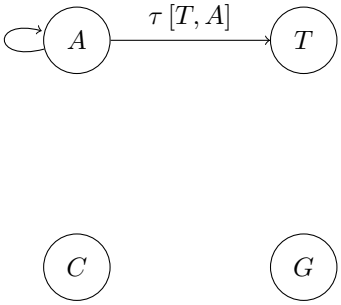
An example of this is a drunken walk. Each new step location is *only* dependent on the previous step, and no steps before that.
The uniform Markov property is that

$$\forall i,j\ \mathbb{P}\left(X_{i+1}=a|X_i=b\right) = \mathbb{P}\left(X_{j+1}=a|X_j=b\right) = \tau\left[a,b\right]$$

With these assumptions, we can write the following

$$\begin{aligned}\mathbb{P}\left(X_1\ldots X_n\right) &= \mathbb{P}\left(X_1\right)\mathbb{P}\left(X_2|X_1\right)\ldots\mathbb{P}\left(X_n|X_1\ldots X_{n-1}\right)\\ &= \mathbb{P}\left(X_1\right)\cdot\prod_{i=1}^{n-1}\mathbb{P}\left(X_{i+1}|X_i\right)\\ &= \mathbb{P}\left(X_1\right)\prod_{i=1}^{n}\tau\left[X_{i+1},X_i\right]\end{aligned}$$

A Markov Chain may often be expressed as a state diagram:



There are areas in the genome where the nucleotide C is followed by G. Areas where this occur are called *CpG islands*, and areas where it does not occur are called *CpG deserts*. These CpG sites generally imply the presence of a transcription start site. They are significant since the presence of the C followed by a G will be replicated in the reverse direction on the second strand of DNA, and are thus more likely to be preserved in replication.

We can use our knowledge of CpG sites, their likelihoods, and Markov chains, and we can thus create a classifier to decide in what part of the genome a particular sample is located. However, how do we choose what is the beginning / end of an area? Consider homework 1, where we used DP to find segments that fit together.