Helsinki University of Technology (HUT)

Espoo, Finland.

Tik-61.181 Bioinformatics
Fall 2000

EXERCISES

Raúl Lozano Mendoza, 55811K

EXERCISES 1

a)

To align these two sequences with non-probabilistic methods, we'll use the basic algorithm of dynamic programming. We have two sequences to align, s = AAAG and t = ACG, |s| = 4 and |t| = 3, so we can build an 5×4 array where entry (i, j) contains the similarity between s[1..i] and t[1..j]. The first row and column are initialized with multiple of the space penalty (-2 in our case). This is because there is only one alignment possible if one of the sequences is empty: Just add as many spaces as there are characters in the other sequence. The score of this alignment is –2k, where k is the length of the nonempty sequence.

We can compute the value for entry (i, j) looking at just three previous entries: those for (i-1, j), (i-1, j-1) and (i, j-1). The formula used is:

$$a[i,j] = \max\begin{cases} a[i,j-1]-2 \\ a[i-1,j-1]+p(i,j) \\ a[i,j-1]-2 \end{cases} \quad where\ p(i,j) = \begin{cases} 1 & s[i]=t[j] \\ -1 & s[i]\neq t[j] \end{cases}$$

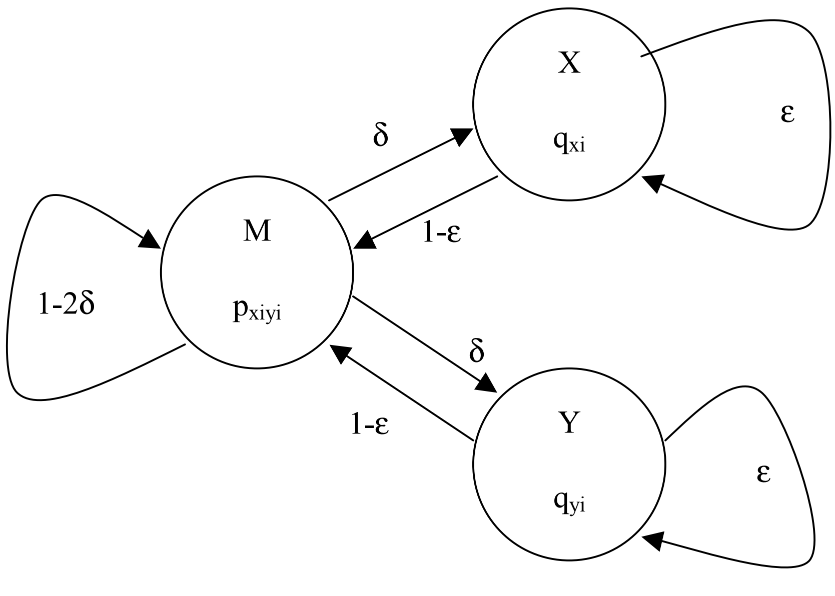
The resulting array is in the appendix of this document.

Looking at the resulting array, we have three possible alignments:

```
AAAG      AAAG      AAAG
A--CG     AC--G     --ACG
```

The optimal alignment is, with the optimal alignment algorithm, the second one.

To align these two sequences with pair HMMs we must do the following: first, we must define a model like this:



Where we have three states: M corresponding to a match, and two states corresponding to inserts, which we have named X and Y. We must also define the probabilities both for emissions of symbols from the states, and for transitions between states. For example, state M has emission probability distribution p_{ab} for emitting an aligned pair a:b, and states X and Y will have distributions q_a for emitting symbol *a* against a gap.

Second, we must use the Viterbi algorithm for pair HMMs to find the most probable path through the pair HMM, which is also the optimal alignment. The algorithm is:

Initialization: $v^M(0,0) = 1$; all other $v^\bullet(i,0)$, $v^\bullet(0,j)$ are set to 0.

Recurrence: i = 1..n, j = 1..m;

$$v^M(i,j) = p_{x_i y_j} \max\begin{cases}(1-2\delta)v^M(i-1,j-1),\\ (1-\varepsilon)v^X(i-1,j-1),\\ (1-\varepsilon)v^Y(i-1,j-1);\end{cases}$$

$$v^X(i,j) = q_{x_i} \max\begin{cases}\delta v^M(i-1,j),\\ \varepsilon v^X(i-1,j);\end{cases}$$

$$v^Y(i,j) = q_{y_j} \max\begin{cases}\delta v^M(i,j-1),\\ \varepsilon v^Y(i,j-1);\end{cases}$$

Termination:

$$v^E = \max(v^M(n,m), v^X(n,m), v^Y(n,m))$$

To find the best alignment, we keep pointers and trace back. To get the alignment itself we keep track of which residues are emitted at each step in the path during the trace back.

b)

The non-probabilistic approach for sequence alignment is the dynamic programming, in its simplest mode with the basic algorithm viewed above, or with more complex dynamic programming algorithms, that arises in the finite state automata (FSA) with multiple states. The FSA are the basis for a probabilistic interpretation of the gapped alignment process, by converting them into HMMs. One advantage of this approach is that we will be able to use the resulting probabilistic model to explore questions about the reliability of the alignment obtained by dynamic programming, and to explore alternative (suboptimal) alignments. We can also build more specialized probabilistic models out of simple pieces, to model more complex versions of sequence alignment.

For doing this conversion from FSA to pair HMM, we must give probabilities both for emissions of symbols from the states, and for transitions between states, which must satisfy the requirement that the probabilities for all the transitions leaving each state sum to one.

Another relation between FSA and HMM is that the algorithms for HMM are all based in dynamic programming also.

c)

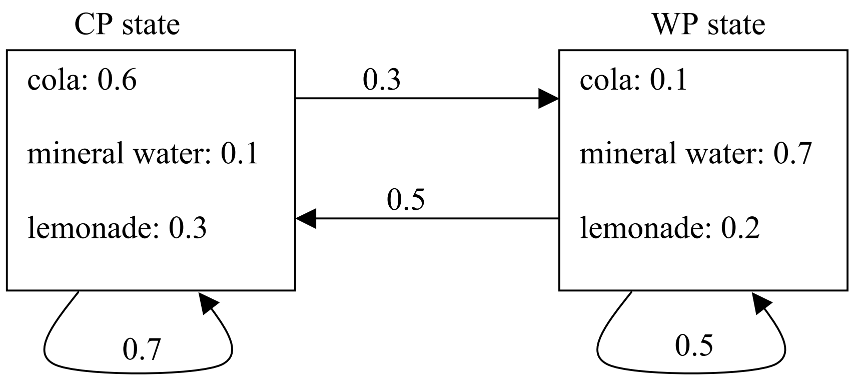
Possible uses of MC-models and HMMs are:

- Given a sequence, we would know if it belongs to a particular family, e.g., CpG islands, prokaryotic genes…
- Assuming the sequence does come from some family, we can say something about its internal structure, e.g., to identify α-helix or β-sheet regions in a protein sequence.
- Pair alignment.
- Multiple alignment.
- Database mining and classification of sequences and fragments, e.g., for protein families.
- Structural analysis and pattern discovery, e.g., features of secondary structure in proteins: hydrophobicity in α-helix.

The essential difference between a Markov chain and a hidden Markov model is that for a hidden Markov model there is not a one-to-one correspondence between the states and the symbols. It is impossible to tell what state the model is when x_i was generated just by looking at x_i. In opposite, with a Markov chain, we can know the state from which the symbol x_i is generated only looking at the symbol x_i.

d)

The state model would be like this:



The full probability of the sequence {cola, lemonade} must be calculated with the Forward algorithm or with the Backward algorithm. Using the Forward algorithm:

Initialization:

$$f_{WP}(cola) = e_{WP}(cola) \times \pi_{WP} = 0.1 \times 1 = 0.1$$
$$f_{CP}(cola) = e_{CP}(cola) \times \pi_{CP} = 0.6 \times 0 = 0$$

Recursion:

$$\begin{aligned} f_{WP}(lemonade) &= e_{WP}(lemonade)\times\left[f_{WP}(cola)\times a_{WP,WP} + f_{CP}(cola)\times a_{CP,WP}\right] = \\ &= 0.2\times(0.1\times0.5+0\times0.3) = 0.01 \\ f_{CP}(lemonade) &= e_{CP}(lemonade)\times\left[f_{WP}(cola)\times a_{WP,CP} + f_{CP}(cola)\times a_{CP,CP}\right] = \\ &= 0.3\times(0.1\times0.5+0\times0.7) = 0.015 \end{aligned}$$

Termination:

$$P(cola,lemonade) = f_{WP}(lemonade) + f_{CP}(lemonade) = 0.01+0.015 = 0.025$$

There are several possibilities for making this HMM to a (visible) Markov model. The first one would be to fix this crazy machine, then we would have a one state Markov model, and, of course, it would be visible. The second one would be to have a three states Markov model, with one state per each drink, in this case, we would know the states sequence just looking at the observations sequence, so it would be visible again.

f)

One advantage of probabilistic models is that, if data D correspond to samples from a model M, then, in the limit of an infinitely large amount of data, the likelihood takes its maximum value for M, i.e. P(D/M) > P(D/N), where N is any other model.

If the parameters of a pair HMM describe the statistics of pairs of related sequences well, then we should use this model with those parameter values for searching.

If we also have a model, R, that gives a good description of the generation of random sequence, then Bayesian model comparison with M and R is an appropriate procedure.

According to this philosophy, we should be using probabilistic models for searching.

But the probabilistic models have some problems that standard methods don't have:

- They don't compute the full probability of the pairs of sequences, summing over all alignments, but instead find the best match, or Viterbi path.
- Regarded as FSAs, their parameters may not be readily translated into probabilities.

This lead us to suggest that probabilistic models may underperform standard alignment methods if Viterbi is used for database searching, but if forward algorithm is used to provide a complete score independent of specific alignment, then the probabilistic models like pair HMM may improve upon the standard methods.

g)

A profile HMM is a certain type of HMM with a structure that in a natural way allows position-dependant gap penalties. A profile HMM can be obtained from a multiple alignment and can be used for searching a database for other members of the family in the alignment very much like standard profiles. We can use the same Viterbi algorithm to find the most probable path to our profile HMM.

A particular feature of protein family multiple alignments is that gaps used to line up with each other leaving solid groups without any gap. A first model for making comparisons with these solid blocks is to specify probabilities to the observation of amino acid *a* in position *i* ($e_i(a)$) comparing it to the probability under a random model. These probabilities behave like elements in a score matrix, that's why this method is called Position Specific Score Matrix (PSSM). This method is used for searching a match of the sequence x in a longer sequence. PSSMs are also useful to find conservation information in protein families, but they don't take care of gaps. The best approach to do this is to build a HMM with a repetitive structure of states, but with different probabilities in each position. A PSSM can be seen like a trivial HMM to which we can add features to take into account insertions and deletions.

h)

We have the following equation to calculate the computing time of an alignment of N sequences of length L:

$$t = (2L)^{N-2}$$

Now, we are using sequences of length 50, therefore, the equation is:

$$t = (10)^{2N-4}$$

What we want to calculate is how many sequences we can align in five billion years, we'll assume that 5 billion years are 5.000.000.000.000 years, and those are $15768*10^{16}$ seconds. Then, the resulting equation is:

$$15768\times10^{16} = (10)^{2N-4}$$

Now, we can take decimal logarithms in both sides and calculate N:

$$\log(15768\times10^{16}) = (2N-4)\times\log(10) = 2N-4$$
$$N = \frac{\log(15768\times10^{16})+4}{2} = 12{,}1 \approx 12$$

Resulting that in five billion years we are able to align 12 sequences of 50 bases each one.

i)

We are going to analyze two progressive alignment methods, one iterative refinement method and the multiple alignment with profile HMM method.

- Feng-Doolittle progressive alignment:

Pros:
  - Fast, thanks to the Fitch & Margoliash clustering algorithm, one of the fast clustering algorithms that build evolutionary trees from distance matrices.
  - Encouraging gaps to occur in the same columns in subsequent pairwise alignments.

Cons:
  - All the alignment is determined by pairwise alignment.
  - Once an aligned group has been built up, it uses position-specific information from the group's multiple alignment to align a new sequence in it.
  - Subalignments are 'frozen': once a group of sequences has been aligned, their alignment to each other cannot be changed at a later stage as more data arrive.
  - Align the whole sequences, regardless of what parts of the sequence are meaningfully aligneable or not.

- Profile alignment (CLUSTALW):

Pros:
  - More flexible than Feng-Doolittle.
  - Simple for linear gap scoring.
  - High accuracy.

Cons:
  - Subalignments are 'frozen': once a group of sequences has been aligned, their alignment to each other cannot be changed at a later stage as more data arrive.
  - Align the whole sequences, regardless of what parts of the sequence are meaningfully alignable or not.

- Iterative refinement (Barton-Sternberg):

Pros:
  - More flexible than progressive alignment.

Cons:
  - More complex than progressive alignment.

- Multiple alignment with profile HMM:

Pros:
  - Biologically realistic view of multiple alignment.
  - Taking into account which parts of the sequence are meaningfully alignable.

Cons:

- To estimate both a model and a multiple alignment from initially unaligned sequences is a hard task.

j)

When the continuous stretches of DNA are over a few hundred bases, the problem of sequencing it becomes unfeasible. However there are some methods for splitting a long DNA chain into random pieces and to produce enough copies of the pieces to sequence. Therefore, we can sequence fragments of them, but these methods leave us the problem of assembling the fragments, and that's the reason for the existence of methods for fragment assembly.

However, even using these methods to sequence DNA, we are restricted to a few tens of thousands of DNA base pairs whereas a chromosome is a DNA molecule with around 10^8 base pairs. This means that these methods only enable us to analyze a very small fraction of the chromosome, not allowing us to analyze bigger structures in the DNA molecule. In order to do this there are methods to create maps of entire chromosomes or of significant fractions of them, and they are called techniques for physical mapping of DNA.

l)

The restriction site mapping technique uses these kind of "fingerprints" that describes uniquely the information in a given fragment of the DNA molecule (restriction enzyme). These fingerprints are used to compare the fragments mainly by overlapping to accomplish the DNA mapping.

One restriction enzyme could recognize one or more restriction sites, however two different restriction enzymes cannot recognize the same restriction site. A restriction site is obtained from information on the length of the fragments; a fragment length is thus its fingerprint. Therefore the restriction site represents uniquely a particular restriction enzyme.

EXERCISES 2

a)

Obvious, with a reversal, we can put one number in any desired position, so if we have N numbers, we can choose the first and put it on the first position, and so on, therefore we are sorting the permutation one by one, but the last two numbers will be sorted only with one reversal, i.e., for sorting a permutation of N numbers, we need at most N-1 reversals.

We can see it with an example. Imagine a permutation of 7 numbers, the permutation could be like this:

2467531

We can do the following reversals for sorting it:

$\underline{2467531}$
$1\underline{357642}$
$12\underline{46753}$
$123\underline{5764}$
$1234\underline{675}$
$12345\underline{76}$
1234567

As we can see, for 7 numbers, we need 6 reversals, but it would be done with fewer reversals if the permutation was different.

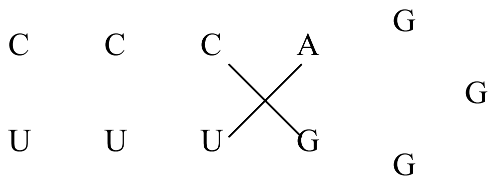
b)

We consider an RNA molecule as a string of *n* characters $R = r_1 r_2 \ldots r_n$ such that r_i belongs to the set {A, C, G, U}. The secondary structure of a RNA molecule is a collection S of pairs (r_i, r_j) of bases such that $1 \le i < j \le n$.

A knot exists when $(r_i, r_j) \in S$ and $(r_k, r_l) \in S$, and $i < k < j < l$.

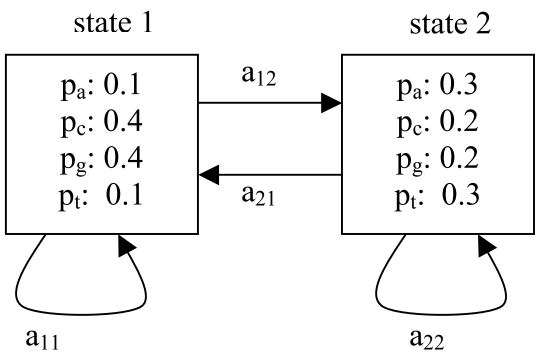
A knot may appear in any RNA sequence with the above property, for example:

If we have the RNA sequence: $R = r_1 \ldots r_{11}$ = CCCAGGGGUUU, we could have that (r_3, r_8) and $(r_4, r_9) \in S$, in this case we would have a knot in the structure:



c)

The model for this problem would be like this:



A expected length of a run of state 1s of 1000 means that for each 1000 iterations we leave the state 1 only once, so the probability of transit from state 1 to state 2 would be a_{12} = 1/1000, and the probability of remain in the state 1 is a_{11} = 1-(1/1000), because these two probabilities must sum to one. The same reasoning would be applied to the state 2, so the transition probabilities for the state 2 are: a_{21} = 1/100 and a_{22} = 1-(1/100).

We can write a stochastic regular grammar G = { T , N , P } that represents the same model, where:

T (terminal symbols) = {A, C, G, T}. These are the 4 bases of DNA.

N (non-terminal symbols) = {1, 2}. These are the 2 possible states.

P (production rules):

1 → A1 (0.1*(1-1/1000)
1 → C1 (0.4*(1-1/1000)
1 → G1 (0.4*(1-1/1000)
1 → T1 (0.1*(1-1/1000)

1 → A2 (0.3*1/1000)
1 → C2 (0.2*1/1000)
1 → G2 (0.2*1/1000)
1 → T2 (0.3*1/1000)

2 → A2 (0.3*(1-1/100)
2 → C2 (0.2*(1-1/100)
2 → G2 (0.2*(1-1/100)
2 → T2 (0.3*(1-1/100)

2 → A1 (0.1*1/100)
2 → C1 (0.4*1/100)
2 → G1 (0.4*1/100)
2 → T1 (0.1*1/100)

d)

The Nussinov folding algorithm, given a sequence x of length L with symbols $x_1,..,x_L$ is the following:

$$\delta(i,j) = \begin{cases} 1 & if\ x_i\ and\ x_j\ are\ base\ pair \\ 0 & if\ not \end{cases}$$

Now, we can modify it to find a maximally scoring structure where a base pair between residues a and b gets a score s(a,b):

$$\delta(i,j) = \begin{cases} s(i,j) & if\ x_i\ and\ x_j\ are\ base\ pair \\ 0 & if\ not \end{cases}$$

And apply the algorithm in the same way, and then at the end of the algorithm the score at the upper right corner of the matrix is the value for the maximum scored valid secondary RNA structure for the given sequence.

After that we can make the backtrace in the same way that with the Nussinov folding algorithm and we will obtain the structure with the highest score.

e)

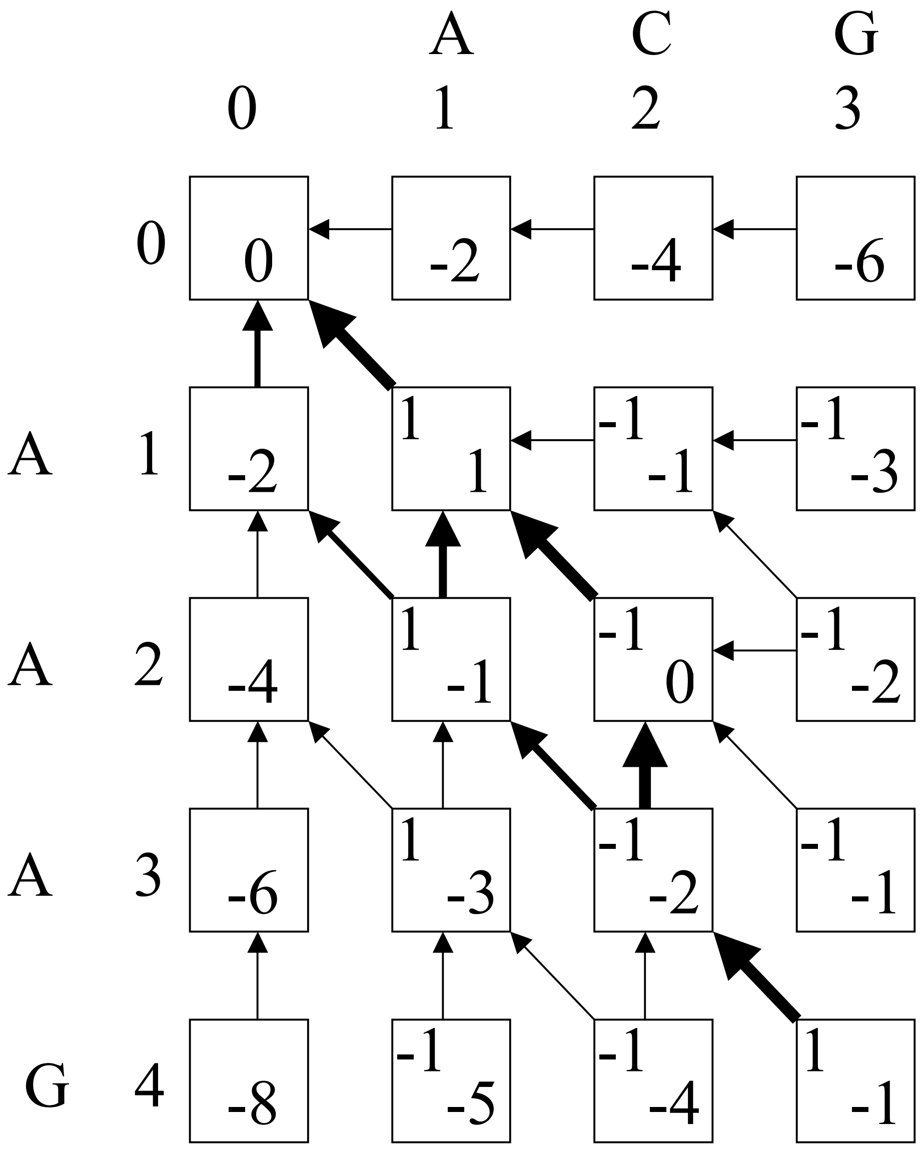
The author proposes to build a substitution matrix based in statistical descriptions of amino acid environments (WAC matrix), taking a measure of the tolerance of a microenvironment for one new amino acid based on the properties of the environment surrounding the lost amino acid.

The main difference with PAM matrices is that this is obtained by empirical methods from measures of the frequency of substitution of alternative residues in each position in some experiments. These substitution matrices are mostly used for general protein searching and homology measuring by pairwise alignment, but they have demonstrated also a good performance in detecting related protein sequences to a certain family.

APPENDIX

Exercises 1. a)

The resulting array of the dynamic programming algorithm is:


A C G
0 1 2 3
0: 0, -2, -4, -6
A 1: -2; 1 1; -1 -1; -1 -3
A 2: -4; 1 -1; -1 0; -1 -2
A 3: -6; 1 -3; -1 -2; -1 -1
G 4: -8; -1 -5; -1 -4; 1 -1

The marked paths are the possible alignments, the bigger arrow the more optimal path.