Tik-61.181 *"Special Course in Information Technology I"*

BIOINFORMATICS

EXERCISES

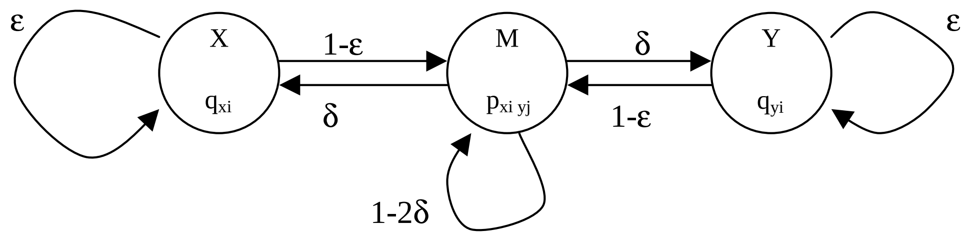
ANTONIO VILLENA BORREGO (55809H)
Antonio.Villena-Borrego@nokia.com

A) Explain in detail how you would align the following sequences with pair HMMs and with non-probabilistic methods:

AAAG
ACG

- HMMs:

In order to align these two sequences with HMMs we must first define a model:



where: $p_{x_i y_j}$ probability for emitting $x_i : y_j$ (a match)
q_{x_i} probability for emitting the symbol x_i and a gap (gap insertion)

Now, we have to apply Viterbi algorithm to find the optimal alignment. The algorithm, applied to our case would go as follows:

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 procedure.

- Non-probabilistic methods (Dynamic programming):

For resolving this exercise we have to build a 5×4 matrix which contains the similarity score between seq1 [1..i] and seq2[1..j]. Although a real scoring matrix could have been used (like “blosum50”,…), I choused the scoring system explained in S&M to simplify the calculations, where:

+1 match
-1 mismatch
-2 gap

Using a trace back procedure, we obtain all the possible alignments from the similarity matrix constructed (see Appendix 1):

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

Although the algorithm explained in the book would conclude that the second one in the most optimal alignment, the fact is that they all have the same score and all of them are optimal alignment.

OBS. The basic alignment algorithm explained in S&M ch.3 would always choose to go up when there are several different possibilities.

B) Analyse briefly the relation of the probabilistic approach and the non-probabilistic approach for sequence alignment.

The non-probabilistic approach for sequence alignment is the dynamic programming. The most complex dynamic programming algorithms are 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. We would also be able to explore for alternative (suboptimal) alignments as well as building more specialized probabilistic models out of simple pieces, which would serve 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, that satisfy 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) Give an example of the possible uses of the MC-model and HMMs in bioinformatics. Outline briefly the main differences between the two models.

There are several different the possible uses of the MC-model and HMMs in bioinformatics, from which I can outline the following:

- Pair wise and multiple alignments, which can help us identify whether two sequences comes from the same evolving branch (their origins are the same).
- Structural analysis of proteins.
- Pattern search, helping us to determine to which family a particular sequence belongs to.
- Classification of the sequence.

The essential difference between a Markov chain and a hidden Markov model is that for a hidden Markov model (HMM) there is not a one-to-one correspondence between the states and the symbols. This is due to the lack of memory property of the HMMs where it is impossible to discern, just by looking at x_i , what was the current state of the model when x_i was generated. The next state only depends of the current state of the system and not of the history of it.

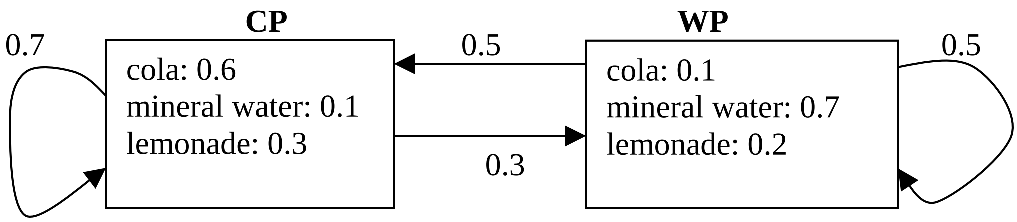
In opposite, with a Markov chain, we can know the state from which the symbol x_i is generated only looking at the symbol x_i, so we are able to reconstruct the evolution of the system just by looking at the sequence of states its gone through.

D) You are buying a drink from the notorious crazy coke machine while absent-mindedly solving some exercises. The machine can be in either of two states: cola preferring state (CP) and mineral water (WP) preferring state. When you put in a coin, the output of the machine can be described with the following probability matrix:

	cola	**mineral water**	**lemonade**
CP	**0.6**	**0.1**	**0.3**
WP	**0.1**	**0.7**	**0.2**

Draw the state model graph with transition probabilities $P(CP \to WP) = 0.3$ **and** $P(WP \to CP) = 0.5$ **and calculate the probability of seeing the output sequence {cola, lemonade} if the machine always starts off in the WP state. What kind of behaviour would make this HMM to a (visible) Markov model?**

The first thing that we have to do is to construct a model using a state model graph:



We then calculate the probability of the sequence {cola, lemonade} making use of the Forward algorithm:

Initialisation:

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

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

Recursion:

$$f_{WP}(lemonade) = e_{WP}(lemonade) \times \left(f_{WP}(cola) \times a_{WP,WP} + f_{CP}(cola) \times a_{CP,WP}\right) =$$
$$= 0.2 \times (0.1 \times 0.5 + 0 \times 0.3) = 0.2 \times 0.05 = 0.01$$

$$f_{CP}(lemonade) = e_{CP}(lemonade) \times \left(f_{WP}(cola) \times a_{WP,CP} + f_{CP}(cola) \times a_{CP,CP}\right) =$$
$$= 0.3 \times (0.1 \times 0.5 + 0 \times 0.7) = 0.3 \times 0.05 = 0.015$$

Termination:

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

To transform this HMM to a Markov model we have to make the model visible, we simply have to transform this model into one that only have three states (one for each drink). Then, by looking at the states, we are able to reconstruct the evolution of the system.

F) Compare briefly the FSAs and pairwise HMMs in searching.

One difference is that in probabilistic modelling we are able to sample the model with a very large amount of data, if this amount tends to infinite then the likelihood takes its maximum value for the model, and the parameters of this optimal model (emissions and transitions matrices) could be used to maximize the likelihood of the data.

Due to this, if the parameters of a pair HMM describe the statistics of pairs of related sequences well, then we should use that model with those parameter values for searching. We also could use the Bayesian model comparison with other model that gives a good description of the generation of random sequence.

We also find two weaknesses in applying current algorithms for these probabilistic models. The first one is that they do not compute the full probability of the pair of sequences, summing over all alignments, but instead find the best match, or Viterbi path, in this case, a model whose parameters match the data is not necessarily the best search model. The second one is that, regarded as FSAs, their parameters may not be readily translated into probabilities.

Therefore we conclude that probabilistic models may under perform standard alignment methods if Viterbi is used for database searching, but if the forward algorithm is use to provide a complete score independent of specific alignment, then pair HMMs may improve upon the standard methods.

G) Analyze the relation of the profile HMMs, HMMs, and PSSMs.

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) Assume we have a number of sequences that are 50 residues long, and that a pairwise comparison of two such sequences takes one second of CPU time on our computer. An alignment of four sequences takes $(2L)^{N-2}=10^{2N-4}=10^4$ seconds (a few hours). If we had unlimited memory and we were willing to wait for the answer until just before the sun burns out in five billion years, how many sequences could our computer align?

Considering that 1 billion years is 10^{12} years (British billion), and not 10^9 (American billion), the solution to this equation would be as follows:

$$(2L)^{N-2} \xrightarrow[L=50]{} (100)^{N-2} \longrightarrow 10^{2N-4} = 5 \text{ billion years}$$

$$(2N-4) * \log(10) = \log(5 \text{ billion years})$$

Taking into account that log(10) = 1 and that 5 billion years = $15768 * 10^{16}$ seconds

$$N = \left\lfloor \log(15768*10^{16}) + 4 \big/ 2 \right\rfloor = \lfloor 12{,}10 \rfloor = 12 \text{ sequences}$$

So in five billion years we are able to align 12 sequences, each one containing 50 bases.

I) Explain briefly the pros, cons, and the most useful applications of the different progressive alignment methods and profile HMMs in multiple alignment.

We find 4 different progressive aligning methods ranging from the easiest but most unrealistic one (Feng-Doolittle progressive alignment), to the most complex but more realistic profile HMMs.

Firstly, we find the Feng-Doolittle method, which is one of the fastest due to its clustering algorithm (Fitch & Margoliash), but it only uses pair wise alignment to determine the multiple alignments. This means that when a group of sequences has been aligned the method uses position-specific information from it to align the following sequence. Therefore when a sub-alignment has been built up, it does not change during the rest of the process.

Then, the CLUSTALW method (Profile alignment) has a higher accuracy and flexibility than the previous one, and it is very simple for linear gap scoring, but it has similar weak points, the sub-alignments that already have been done remains without changes to the end of the alignment, and it aligns always whole sequences, never taking into account what fragments of the sequence are alienable or not.

The Barton-Sternberg method (Iterative refinement) has the advantage that is more flexible than the progressive alignment methods, but its complexity is also bigger.

Profile HMMs for multiple alignments are, biologically, the most realistic algorithm to solve the problem. That is because this algorithm takes into account if the current fragment of the sequence is meaningfully alienable. On the other side, profile HMMS has the most difficult implementation algorithm due to the difficult construction of the model and the estimation of a multiple alignment from unaligned sequences.

J) Explain the reasons why there are methods for fragment assembly and physical mapping.

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.

K) Suppose we have the following fragments:

F_1: ATCCGTTGAAGCCGCGGGC
F_2: TTAACTCGAGG
F_3: CGGGCAGTACTTAA
F_4: CCCGACAGAGAT
F_5: CGACTCCCGACACA
F_6: CACAGATCCGTTGAAGCCGCGGG
F_7: TACTTAACTCGAG
F_8: CGCGGGCAGTACTT

And we know that the total length of the target molecule is about 55 pairs. Assemble these fragments and obtain a consensus sequence.

```
      1 . . . 5 . . . . 10 . . . . 15 . . . . 20 . . . . 25 . . . . 30 . . . . 35 . . . . 40 . . . . 45 . . . . 50
f1 →                 ATCCGTTGAAGCCGCGGGC
f2 →                                        TTAACTCGAGG
f3 ←                              CGGGCAGTACTTAA
f4 ←       CCCGACAGAGAT
f5 →  CGACTCCCGACACA
f6 →            CACAGATCCGTTGAAGCCGCGGG
f7 ←                                     TACTTAACTCGAG
f8 →                            CGCGGGCAGTACTT
      ----------------------------------------------------
X  =  CGACTCCCGACACAGATCCGTTGAAGCCGCGGGCAGTACTTAACTCGAGG
```

This answer has 50 bases (length (X) = 50), which is close to the given target length of 55 base pairs.

In the solution, we indicate by an arrow the chosen orientation:

$\rightarrow$ means fragment as it is
$\leftarrow$ means its reverse complement

L) Explain why two restriction sites for two different restriction enzymes cannot coincide.

We have to look into the meaning of "fingerprints", in which the definition of restriction sites is based to find the reason why two restriction sites for two different restriction enzymes cannot coincide.

A "fingerprint" describes a piece of information in a given fragment of the DNA molecule (restriction enzyme) in a unique form. This fingerprint is used to compare the fragments, mainly by overlapping, in order 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.

Set of exercises 2.

A) Prove that any un-oriented permutation over n elements can be sorted with at most n-1 reversals.

The demonstration is very simple. A reversal is used to 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

And the number of reversals would go as follows:

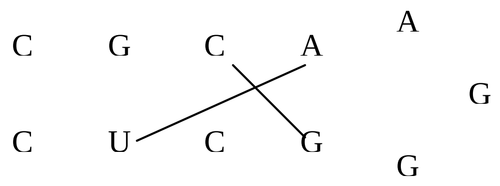
1. - $\underline{2467531}$
2. - 1$\underline{357642}$
3. - 12$\underline{46753}$
4. - 123$\underline{5764}$
5. - 1234$\underline{675}$
6. - 12345$\underline{76}$
7. - 1234567

So for 7 numbers, we've need 6 reversals, although it can be done with fewer reversals if the permutation was different.

B) Give an example of an RNA sequence where a knot may appear.

If we consider an RNA molecule as a string of n characters $R = r_1r_2 \ldots r_n$ such that r_i belongs to the set {A, C, G, U}, then the secondary structure would be a set S of pairs of bases (r_i, r_j) such that $1 \leq i < j \leq n$.

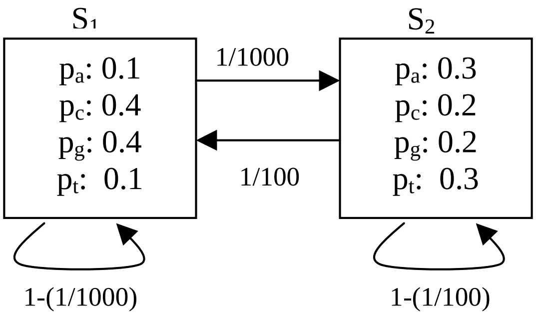
Considering that a knot exists when $(r_i, r_j) \in S$ and $(r_k, r_l) \in S$, and $i < k < j < l$., then a knot may appear when we have, for example, a RNA sequence $R = r_1 \ldots r_{11}$ = CGCAAGGGCUC, and we have that (r_3, r_8) and $(r_4, r_{10}) \in S$:



C) Consider a simple HMM that models two kinds of base composition in DNA. The model has two states fully interconnected by four state transitions. State 1 emits CG-rich sequence with probabilities (p_a, p_c, p_g, p_t) = {0.1, 0.4, 0.4, 0.1} and state 2 emits AT-rich sequence with probabilities (p_a, p_c, p_g, p_t) = {0.3, 0.2, 0.2, 0.3}.

a) Draw this HMM.
b) Set the transition probabilities so that the expected length of a run of state s1 is 1000 bases and the expected length of a run of state 2s is 100 bases.
***c)* Give the same model in stochastic regular grammar form with terminals, non-terminals, and production rules with their associated probabilities.**

The HMM could be represented as follows with the transition probabilities set to a_{12}=1/1000 and a_{21}=1/100:



Transforming this model into a stochastic regular grammar where the terminal symbols would be{A, C, G, T} and the non-terminals symbols {S_1, S_2}, having the following production rules:

$S_1 \rightarrow AS_2$ (0.3*1/1000)	$S_1 \rightarrow AS_1$ (0.1*(1-1/1000))
$S_1 \rightarrow CS_2$ (0.2*1/1000)	$S_1 \rightarrow CS_1$ (0.4*(1-1/1000))
$S_1 \rightarrow GS_2$ (0.2*1/1000)	$S_1 \rightarrow GS_1$ (0.4*(1-1/1000))
$S_1 \rightarrow TS_2$ (0.3*1/1000)	$S_1 \rightarrow TS_1$ (0.1*(1-1/1000))
$S_2 \rightarrow AS_1$ (0.1*1/100)	$S_2 \rightarrow AS_2$ (0.3*(1-1/100))
$S_2 \rightarrow CS_1$ (0.4*1/100)	$S_2 \rightarrow CS_2$ (0.2*(1-1/100))
$S_2 \rightarrow GS_1$ (0.4*1/100)	$S_2 \rightarrow GS_2$ (0.2*(1-1/100))
$S_2 \rightarrow TS_1$ (0.1*1/100)	$S_2 \rightarrow TS_2$ (0.3*(1-1/100))

We can clearly see in the grammar that the probabilities for all the transitions leaving each state sum to one.

D) Show that the Nussinov folding algorithm can be trivially extended to find a maximally scoring structure where a base pair between residues a and b gets a score s (a, b). (For instance, we might set s(G, C) = 3 and s(A, U) = 2 to better reflect the increased thermodynamic stability of GC pairs.)

Given a sequence x of length L with symbols $x_1,..,x_L$ then the algorithm goes:

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

If we modify the equations to find a maximally scoring structure where a base pair between residues a and b gets a score s (a, b):

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

Then we 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 back trace in the same way that with the Nussinov folding algorithm and we will obtain the structure with the highest score.

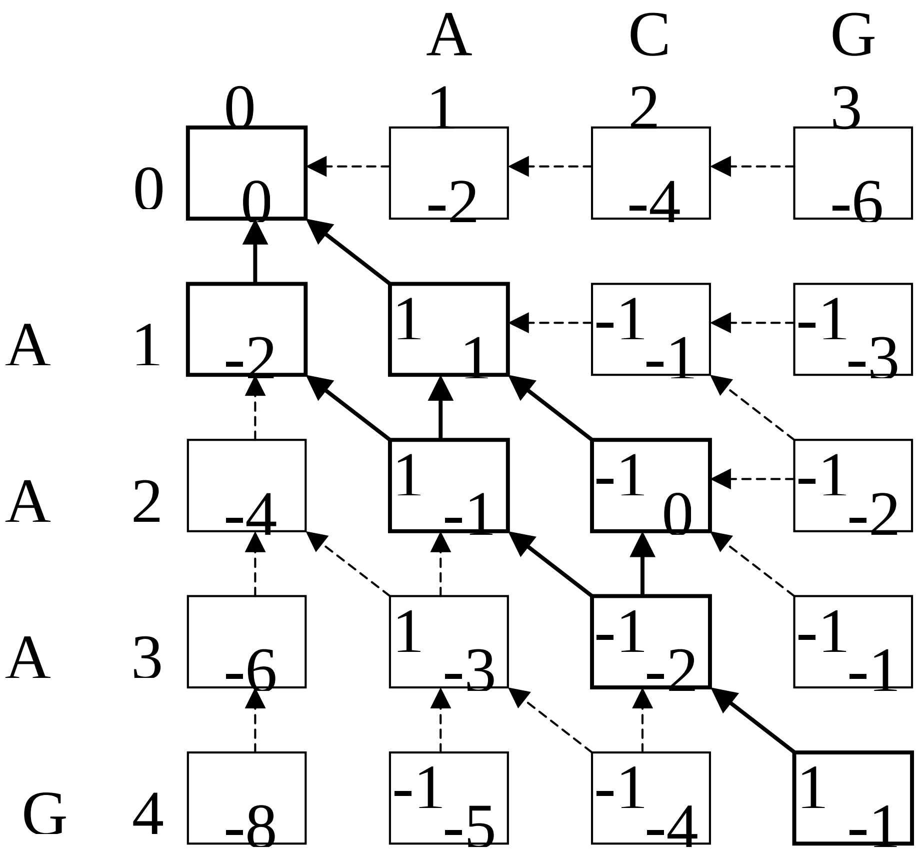
E) The authors [SSK, Ch. 11] propose a method for computing a substituting matrix for amino acids. Characterize the differences from the PAM matrices. Comment the proposal. Where would you use each?

The method proposed is to build a substitution matrix (WAC), based on statistical descriptions of the amino acid environments, by 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.

Other substitution matrices (score matrices) like PAM matrices or BLOSUM, mostly used for general protein searching and homology measuring by pairwise alignment, have been obtained by empirical methods. These empirical methods are taken from measures of the frequency of substitution of alternative residues in each position in some experiments. These matrices have demonstrated a good performance in detecting related protein sequences to a certain family.

Appendix 1.

The resulting similarity matrix obtain in exercise A) is like:



Where we can clearly see the three possible paths generated in the trace back procedure.

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