TRAINING HIDDEN MARKOV MODELS

Andrea Giansanti
Dipartimento di Fisica, Sapienza Università di Roma

Andrea.Giansanti@roma1.infn.it

CB_23_24_L35 , Rome 22 Dec 2023

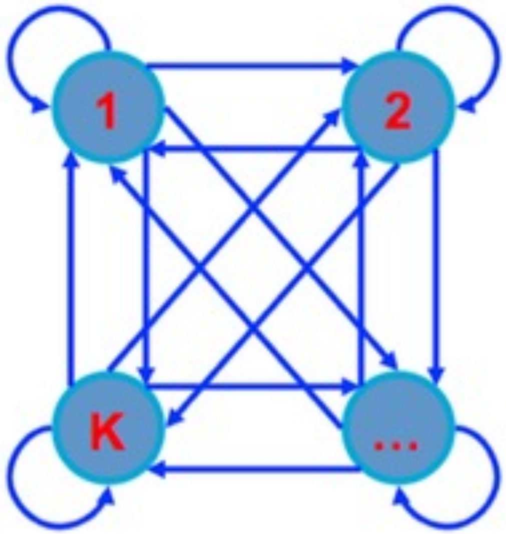
Dipartimento di Fisica

Sapienza
Università di Roma

- About letters, alphabets and states
- First order Markov models
- Higher order models
- Hidden Markov models
- Evaluation Problem
- Decoding Problem
- Learning problem

- Used in computational structural biology
- e.g. to associate to each family of functionally similar proteins
- a generative probabilistic model as a fingerprint

Refresh: definition of a HMM



Definition: A hidden Markov model (HMM)

- Alphabet $\Sigma = \{ b_1, b_2, \ldots, b_M \}$
- Set of states $Q = \{ 1, \ldots, K \}$
- Transition probabilities between any two states

a_{ij} = transition prob from state i to state j

$a_{i1} + \ldots + a_{iK} = 1$, for all states i = 1…K

- Start probabilities a_{0i}

$$a_{01} + \ldots + a_{0K} = 1$$

- Emission probabilities within each state

$e_i(b) = P(x_i = b \mid \pi_i = k)$

$e_i(b_1) + \ldots + e_i(b_M) = 1$, for all states i = 1…K

The three main questions on HMMs

1. Evaluation

GIVEN a HMM M, and a sequence x,

FIND Prob[x | M]

2. Decoding

GIVEN a HMM M, and a sequence x,

FIND the sequence π of states that maximizes P[x, π | M]

3. Learning

GIVEN a HMM M, with unspecified transition/emission probs., and a sequence x,

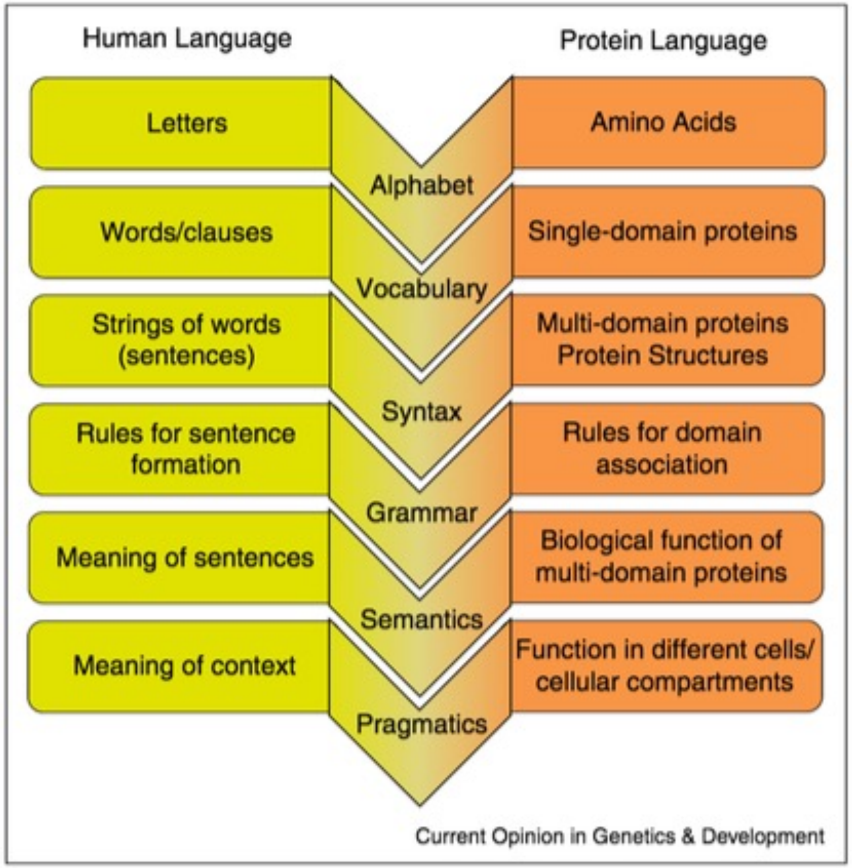
FIND parameters $\theta = (e_i(.), a_{ij})$ that maximize P[x | θ]

The vocabulary of proteins

Protein domains correspond to the words in the proteins language. Domains can be distinguished by their sequence (sequence profiles) or their structure (classification databases). Sequence profiles are most commonly represented using statistical models such as position specific scoring matrices (PSSM) and hidden Markov models (HMM). HMM-based methods include Pfam [9], EVEREST [10], SMART [11], and PANTHER [12]. PSSM-based methods include PRINTS [13], PROSITE [14] and ProDom [15]. Here we refer to sequence profiles as domains or words. Structure-based classifications including SCOP [16,17], SCOP2 [18] and CATH [19–21], as well as predicted domain structures as in SUPERFAMILY [22,23], Gene3D [24,25], ECOD [26] and COPS [27] are not considered here.

Sequence-based methods differ in coverage, level of curation and definition of families. Compilation databases like CDART [28–30] and InterPro [31] match all sets of profiles to all known sequences (Figure 2). This facilitates the classification of the protein universe into protein families by providing the locations of different domains along every sequence. Often two different sequence profiles match the same region of sequence leading to several domains, or words, for the same physical object. This can lead to confusion and such synonyms need to be recognized and possibly eliminated [32••].

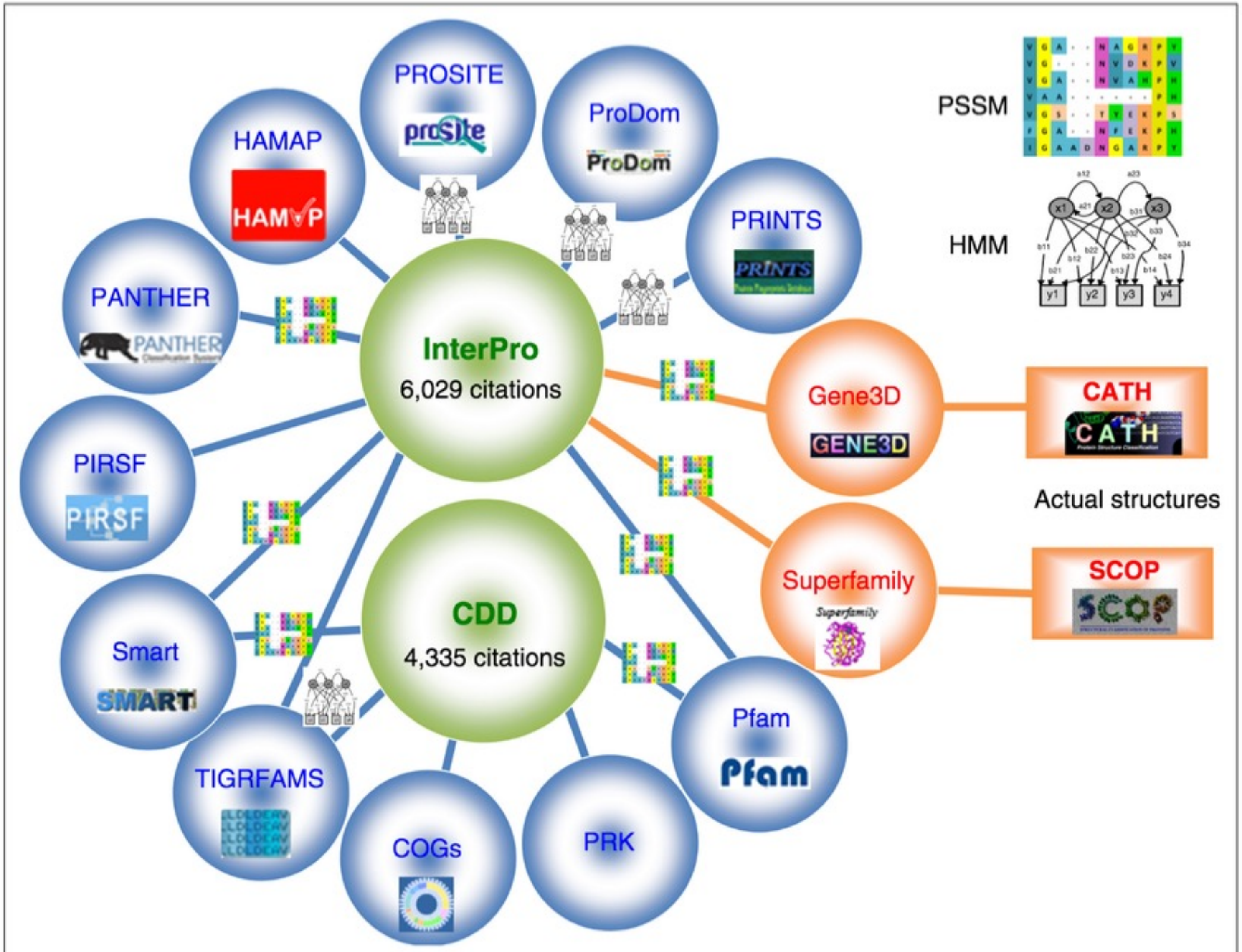
Figure 1


Human Language
Protein Language
Letters
Amino Acids
Alphabet
Words/clauses
Single-domain proteins
Vocabulary
Strings of words (sentences)
Multi-domain proteins Protein Structures
Syntax
Rules for sentence formation
Rules for domain association
Grammar
Meaning of sentences
Biological function of multi-domain proteins
Semantics
Meaning of context
Function in different cells/ cellular compartments
Pragmatics

Current Opinion in Genetics & Development

Analogy between human and proteins languages. In this comparison, the vocabulary (domains) of proteins is built from an alphabet of amino acids. The syntax principles enable domain association to form multi-domain architectures, a process governed by hierarchical rules (grammar), that determine the structure and hence the biological function (semantics) of proteins. In several languages, for example in English, a number of different classes of words exist (nouns, adjectives, verbs, adverbs, pronouns, conjunctions). Each class has its task in the language, that is, nouns name words, adjectives describe nouns, verbs are action words, conjunction connect words. Analogously, one can also distinguish different classes of domains with different tasks (motors, binding proteins, enzymes, signaling proteins, structural proteins, targeting proteins).

Figure 2



Current Opinion in Genetics & Development

Sequence profile databases can be sequence-based (blue circles) or structure-based (orange circles). Sequence-based profiles are derived by mainly two methods: HMMs (Hidden Markov Models) or PSSMs. (Position Sensitive Sequence Matrices). Structure-based profiles in Gene3D and superfamily are generated from HMMs built from actual structures coming from CATH and SCOP, respectively. Two main integrative resources, CDART and InterPro, are shown (green circles) with the databases they include.

CB_2324_L34

PROBABILISTIC TRAINING OF HIDDEN MARKOV MODELS

(See H&A chap 10.1

10.2.1 Likelihood ratios

10.2.2 Prior and posterior probabilities

10.2.3 choosing model parameters

OBS.1 The likelihood of observing n_a counts for a.acid a (with $a = 1, 2, \ldots, 20$) [i.e. the likelihood of having frequency $f_a = \frac{n_a}{n_{TOT}}$ in a type of proteins (e.g. TM helixes in membrane proteins)

$$L = \prod_{a=1}^{20} (P_a)^{n_a}$$ = this is the likelihood of having an "occupation" vettor $(n_1, n_2, \ldots, n_k, \ldots n_{20})$

The maximum likelihood estimate of
the parameters P_a is $\frac{n_a}{n_{TOT}}$
[see problem 10.2]

given a sequence with obsered occurreuces =
$\{n_a\}_{a=1}^{20}$, then the likelihood of the
sequence is $L = \prod_{a=1}^{20} (P_a)^{n_a}$

[this is an urn model with const. composition]

Let us optimize $\text{Log } L = \sum_a n_a \log P_a +$
$+ \lambda \left(1 - \sum_{a=1}^{20} P_a\right)$ maximizing with one
constraint to be obeyed, [P_as should be
probabilities.]
(method of Lagrange multipliers)

$$\frac{\partial \log(L)}{\partial p_\ell} = \sum_a \frac{\partial}{\partial p_\ell} n_a \text{Log } P_a + \lambda \frac{\partial}{\partial p_\ell}\left(1 - \sum_a P_a\right)$$

$1 \leq \ell \leq 20$

$$= \sum_a n_a \frac{\partial}{\partial p_\ell} \text{Log } P_a - \lambda \sum_a \frac{\partial}{\partial p_\ell} P_a$$

$$= \sum_a n_a \frac{1}{P_a} \frac{\partial}{\partial p_\ell} P_a$$

$$= \frac{n_\ell}{p_\ell} - \lambda = 0$$

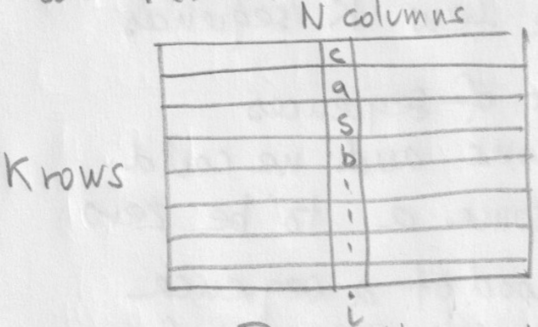
$$n_\ell = \lambda p_\ell \rightarrow \boxed{p_\ell = \frac{n_\ell}{\lambda} \quad \forall \ell}$$

$$\lambda \sum_\ell p_\ell = \sum_\ell n_\ell = n_{TOT} \longrightarrow p_\ell = \frac{n_\ell}{n_{TOT}}$$

$\lambda = n_{TOT}$

Consider now a PROFILE

In a nutshell: a protein profile is an (ungapped) K×N table



filled of letters taken from the amino-acid alphabet

define P_{ia} the probability of getting amino acid a at site i.

- From the previous argument: $P_{ia} = \frac{n_{ia}}{K}$ (ML estimate)

OBS. now, once we have TRAINED the model estimating P_{ia}s we can evaluate the likelihood of a seqence (not belonging to the profile) $S \equiv (x_1, x_2, x_3, \dots, x_N)$ with $x_i \in$ Alphabet of amino-acids

What is the L[S] Likelihood of S in the model? Answer

$$L = \prod_{i=1}^{N} P_{ix_i}$$

$P_{1x_1} \cdot P_{2x_2} \cdots P_{Nx_N}$

CONSIDER NOW THE PROBLEM OF PSEUDO COUNTS

Often defininig a classifier Profile we can use only a few K sequences

This finite number of sequences produces fluctuations and we could obsere n_{ia} for some a to be zero

Then the likelihood of a sequence with a at place i is ZERO! (WOW!)

TOO RIGID MODEL! So people invented The pseudo-count approach to set

$$P_{ia} = \frac{n_{ia} + A q_a}{K + A}$$

pseudo count

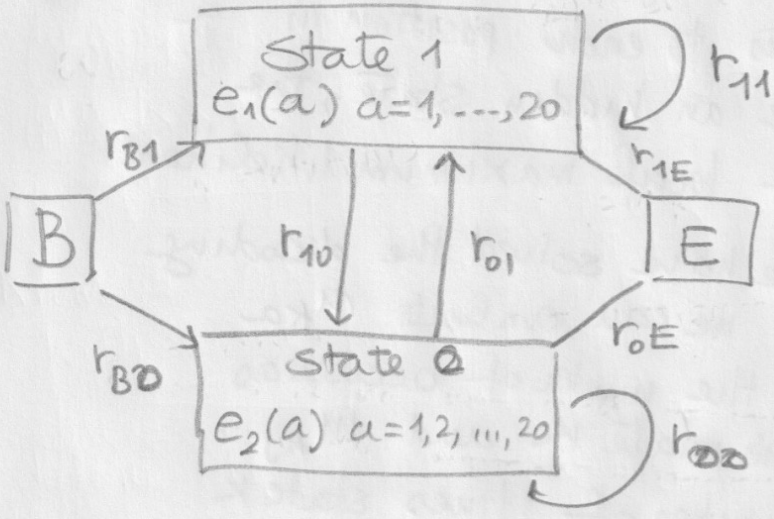
K and A integers

q_a are "prior" expected relative frequencies (e.g. the frequencies you find in the SWISS PROT database.)

[this is a bayesian approach]

K and A are the weights we put on THE DATA OBSERVED and on the PRIORS $\{q_a\}$

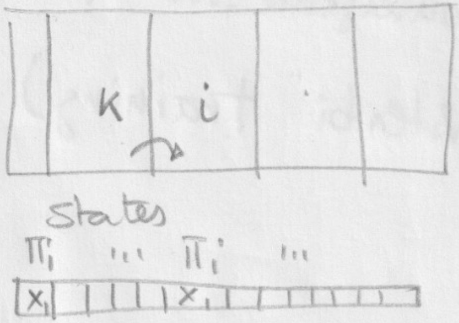
THE PARAMETERS OF an HMM (simplest)



TRAINING OF THE MODEL

$e_k(a)$ emission prob. in state k of symbol a $\simeq \frac{n_{ka}}{n_k^T}$

$r_{kj}\ell = m_{kj}/n_k^{TOT}$



The Viterbi algorithm associates to each position in a sequence an hidden state, the state of local maximum likelihood

Once we have solved the decoding problem we can evaluate n_{ka} n_{ka} the number of occurrences of a in state k and m_{kj} the number of times state k goes into state j.

We can then recalculate

$$e_k(a) \approx n_{ka} / n_k^{tot}$$

$$r_{kj} = m_{kj} / n_k^{tot}$$

and we get convergence ....

(This is the Viterbi training)

BOX 10.2
The Viterbi algorithm

The Viterbi algorithm calculates the most probable path of hidden states for a given sequence, $x_1 \ldots x_N$. Consider an HMM with emission probabilities $e_k(a)$ for emitting letter a when in hidden state k, and with transition probabilities r_{hk} between hidden states h and k. Let $v_k(i)$ be the likelihood of the most probable path for the first i letters in the sequence, given that the i^{th} letter is in state k. The algorithm is a type of dynamic programming routine, similar to those used for sequence alignment (see Chapter 6). We can initialize the algorithm by setting

$$v_k(1) = r_{Bk}e_k(x_1) \tag{10.26}$$

as there is only one path to each of the hidden states for the first letter of the sequence. For subsequent letters, $i = 2$ to N, we have the recursion relation

$$v_k(i) = \max_h(v_h(i-1)r_{hk}e_k(i)) \tag{10.27}$$

Finally, v_E, the likelihood of the best total path ending in the end state, is:

$$v_E = \max_h(v_h(N)r_{hE}) \tag{10.28}$$

Just as with sequence alignment, we need to store an array of pointers for back-tracking, i.e., we can define variables $B_k(i)$ that are set equal to the hidden state h that gave the maximum term in step (10.27). In this way, we can trace the path through the model back from the end to the beginning.

The time taken by the Viterbi algorithm is $O(KN)$, where K is the number of hidden states summed over at each step of the recursion. Thus, it is a polynomial time algorithm, even though the number of possible paths through the model scales as K^N.

Baum-Welsh training
(maximization of expectation)
based on forward-backward algorithm)

Guess initial values of e_k and $r_{kk'}$

$P(\pi_i = k)$ probability that point i in the sequence is in state k

$$\overline{n_{ka}} = \sum_{\substack{\text{sites where} \\ x_i = a}} P(\pi_i = k) \quad (10,35)$$

The probability that position i in the sequence is in state k while $i+1$ is in state j see (10,36)

$$\overline{m_{kj}} = \sum_i P(\pi_i = k, \pi_{i+1} = j)$$

expectation part

now use $\overline{n_{ka}}$ e $\overline{m_{kj}}$ in

$$e_k(a) \approx \frac{\overline{n_{ka}}}{n_k^{tot}} \qquad r_{kj} \text{ (a } \frac{\overline{m_{kj}}}{n_k^{tot}}$$

After the cycle one gets stable parameters - that are the result of the TRAINING