Applied Probabilistic Machine Learning

Hidden Markov models

Hugues Richard *RichardH@rki.de*
Ivan Tunov *Ivan.Tunov@student.hpi.uni-potsdam.de*

MF1 - Genome Competence Center
DACS

Robert Koch Institute (RKI)
Hasso Plattner Institute (HPI)

12 December 2024

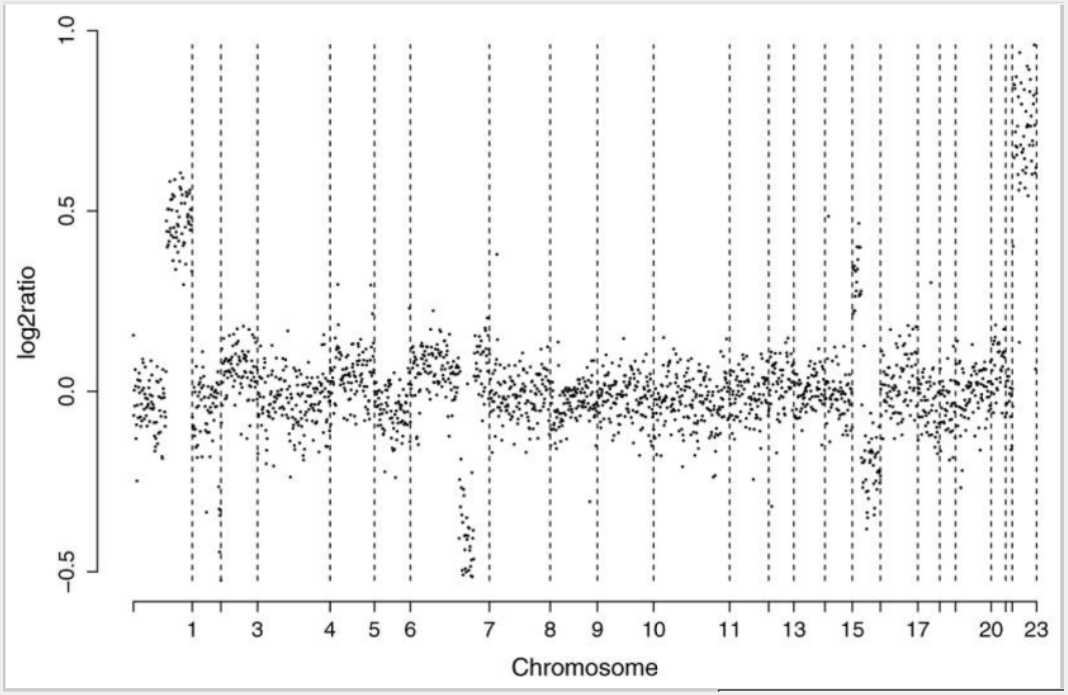
Learning goals

- Understand what are Hidden Markov Models (HMM)
  - From mixture models to HMM
  - Motivating examples
- Work a toy example with HMM: the occasionally dishonest casino
- Computing quantities for HMMs
  - Learn to compute the probability of a sequence generated by an HMM
  - Reconstruct the most likely hidden path
  - Learn to estimate the parameters of an HMM with EM algorithm
- Various Examples of HMMs in biology

Introduction

Finding signal in sequential data - example 1

- Array CGH data: Quantify DNA relative copy number
  - Compare 2 samples DNA count along the chromosome $\mathbf{x}_t = \log_2 \frac{n_t^1}{n_t^2}$
  - Question: What are the relative copy numbers at each positions on the chromosome? $n_t^i \in \{0, 1, 2, 3, 4\}$



- We want to estimate Y underlying ratio $Y \in \{0, 1/2, 1, 1.5, 2, \ldots\}$

First solution: mixture models

- We hypothesize a mixture with the observed log-ratio normally distributed given the expected ratio y:

$$\mathbf{x}_t \mid Y_t = y \sim \mathcal{N}(\mu = \log_2(y), \sigma_y^2)$$

- We fix the possible values y and we estimate the σ_y using EM.
- Allocation can be done with the bayes rule $\mathbb{P}(y \mid \mathbf{x}_t)$.

First solution: mixture models

- We hypothesize a mixture with the observed log-ratio normally distributed given the expected ratio y:

$$\mathbf{x}_t \mid Y_t = y \sim \mathcal{N}(\mu = \log_2(y), \sigma_y^2)$$

- We fix the possible values y and we estimate the σ_y using EM.
- Allocation can be done with the bayes rule $\mathbb{P}(y \mid \mathbf{x}_t)$.
- **Problem**: we expect longer regions with the same value y, here position are independent.

First solution: mixture models

- We hypothesize a mixture with the observed log-ratio normally distributed given the expected ratio y:

$$\mathbf{x}_t \mid Y_t = y \sim \mathcal{N}(\mu = \log_2(y), \sigma_y^2)$$

- We fix the possible values y and we estimate the σ_y using EM.
- Allocation can be done with the bayes rule $\mathbb{P}(y \mid \mathbf{x}_t)$.
- **Problem**: we expect longer regions with the same value y, here
  - Add a dependency on the sequence of the Y_t

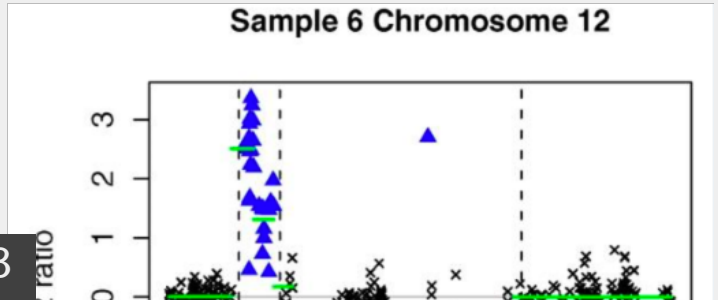
position are independent.

First solution: mixture models

- We hypothesize a mixture with the observed log-ratio normally distributed given the expected ratio y:

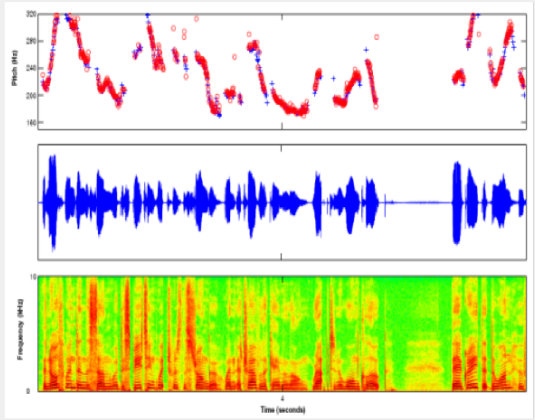
$$\mathbf{x}_t \mid Y_t = y \sim \mathcal{N}(\mu = \log_2(y), \sigma_y^2)$$

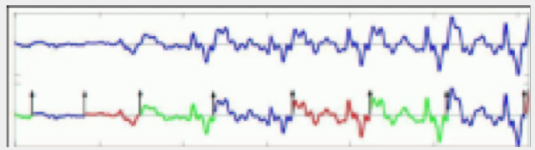
- We fix the possible values y and we estimate the σ_y using EM.
- Allocation can be done with the bayes rule $\mathbb{P}(y \mid \mathbf{x}_t)$.
- **Problem**: we expect longer regions with the same value y, here position are independent.
  - Add a dependency on the sequence of the Y_t
  - HMM: Markovian dependency (more chance to stay with the same value)
  - bayes rule will reconstructs long ranges with similar value Y_t



Other motivations - 1

- Speech recognition:
  - We *understand* multiple levels of abstraction:
    - words ($y_t \in \mathcal{D}$) $\rightarrow$ phonemes ($x_t \in \mathbb{R}$)
    - phonemes ($y_t \in \mathbb{R}$) $\rightarrow$ soundwaves ($\mathbf{x}_t \in \Sigma$)
  - Infer phonemes from soundwave and words from phonemes





S. Roweis 2004

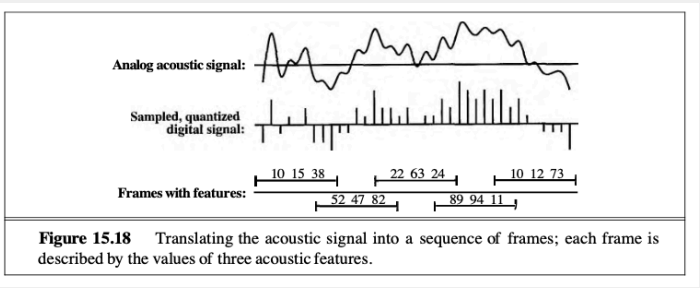


Figure 15.18 Translating the acoustic signal into a sequence of frames; each frame is described by the values of three acoustic features.

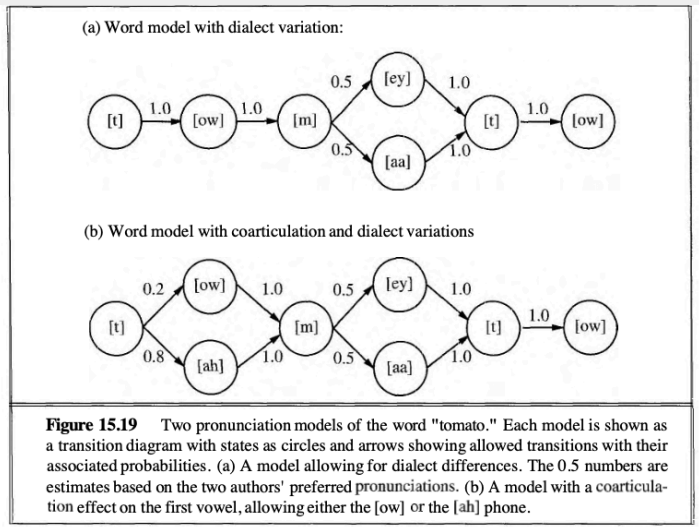


Figure 15.19 Two pronunciation models of the word "tomato." Each model is shown as a transition diagram with states as circles and arrows showing allowed transitions with their associated probabilities. (a) A model allowing for dialect differences. The 0.5 numbers are estimates based on the two authors' preferred pronunciations. (b) A model with a coarticulation effect on the first vowel, allowing either the [ow] or the [ah] phone.

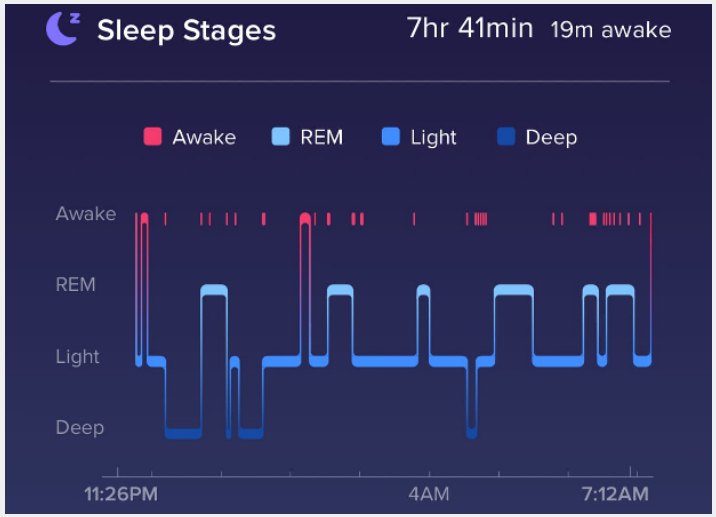
[Russell and Norvig, 2003]

Other motivations - 2

- Sleep pattern tracking:
  - Monitor movement of the hand (6D) + optionally heart rate

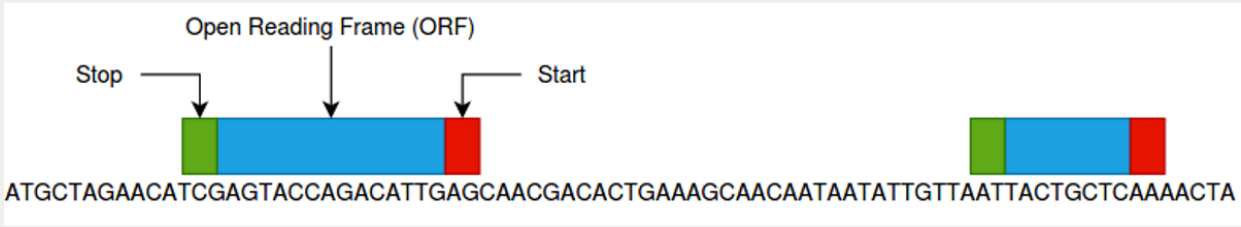
$$\mathbf{x}_t \in \mathbb{R}^6$$

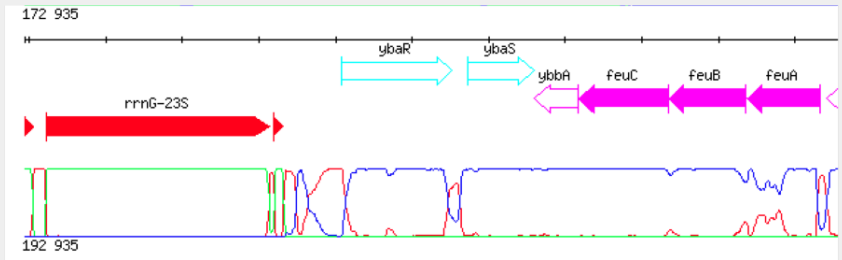
  - infer sleep stages



Other motivations - 2

- Gene Annotation
  - A (prokaryotic) gene is a start codon `ATG` followed by codons and finishing with a stop codon `TAA, TAG, TGA`.
- Can we annotate the genes?
  - Observations $\mathbf{x}_t \in \{A, C, G, T\}$
  - We want to constrain open reading frames (ORFs)
  - We want to infer $y_t \in$ {`stop`, `start`, `coding`, `non coding`}.





Hidden Markov Models

- We observe a sequence of events $\mathbf{x}_t$ that we want to annotate in categories
  - I do not move during deep sleep, I move more in light sleep...
  - Introduce a latent variable Y_t and say it's a Markov chain!

$$\begin{array}{ccccccccc} \ldots & \to & Y_{t-1} & \to & Y_t & \to & Y_{t+1} & \to & \ldots \\ & & \downarrow & & \downarrow & & \downarrow & & \\ & & X_{t-1} & & X_t & & X_{t+1} & & \end{array}$$

Sleep stage, copy number, part of gene

coordinates, intensity ratio, DNA sequence

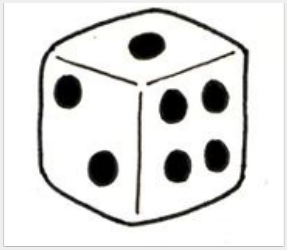
- The Markov chain on Y_t allows to control in two ways:
  - Smoothing the signal
    - Copy number variants are usually ℓ bp long.
    - Deep sleep stage lasts around t minutes
  - Integrating hard constraints
    - A gene always ends with a stop codon

An HMM toy example

The occasionally dishonest casino

- Let's imagine a casino where the croupier alternates between 2 dices:

Fair dice



$p_{123456} = \frac{1}{6}$

Loaded dice

$p_6 = \frac{1}{2}, p_{12345} = 0.1$

- We start with the fair dice
- We randomly change dice at each turn
  - Keep the loaded dice with probability 0.9
  - Keep the fair dice with probability 0.95

HMM PARAMETERS

- We need to know:

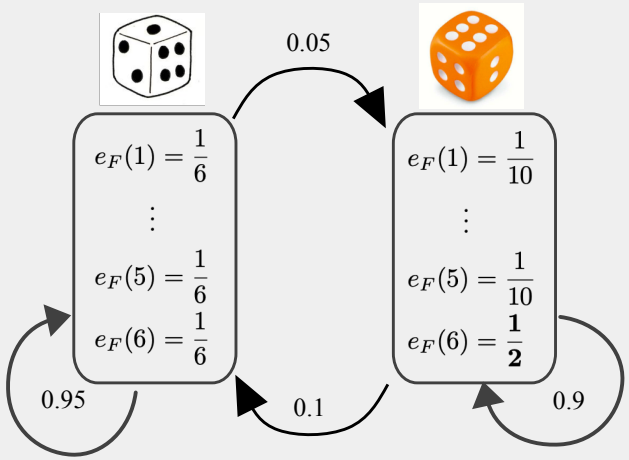
- Starting distribution π (here $\pi_F = 1$)
- The Markov Chain Y_t of transitions between dices, taking values in $\{F, L\}$

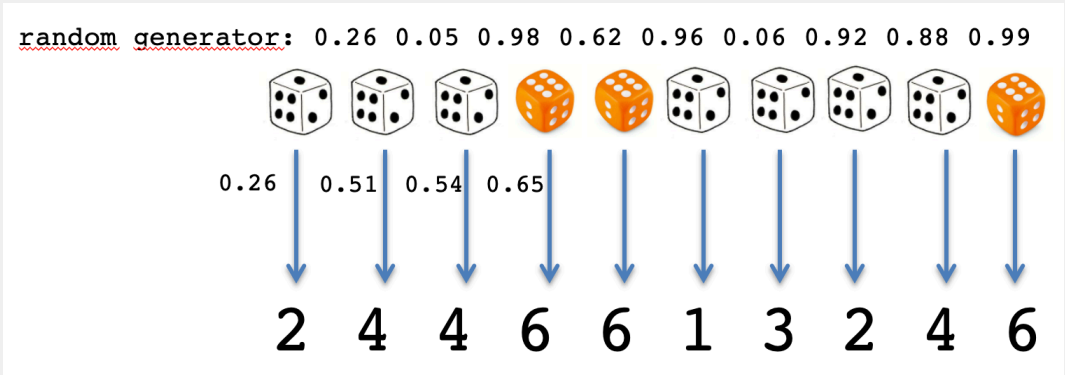
$$\alpha_{uv} = \mathbb{P}(Y_{t+1} = v \mid Y_t = u)$$

- The dice conditional observations X_t, taking values in $\{1, 2, \ldots, 6\}$:

$$e_u[i] = \mathbb{P}(X_t = i \mid Y_t = u)$$

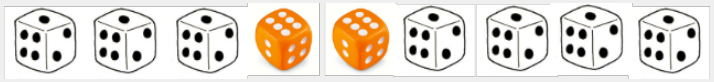
- Generating a dice sequence:





Place your bets: Probability of a sequence

- When the sequence of dice types (Hidden states) is known:



2 4 4 6 6 1 3 2 4

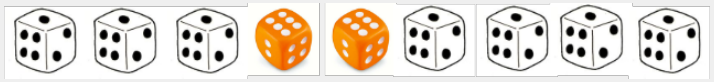
$$\mathbb{P}(\text{🎲🎲🎲🎲🎲🎲🎲🎲🎲}, 244661324) = \mathbb{P}(2 \mid \text{🎲})\mathbb{P}(\text{🎲} \mid \text{🎲})\mathbb{P}(4 \mid \text{🎲})\mathbb{P}(\text{🎲} \mid \text{🎲})$$
$$\cdot\, \mathbb{P}(4 \mid \text{🎲})\mathbb{P}(\text{🎲} \mid \text{🎲})\mathbb{P}(6 \mid \text{🎲})$$
$$\cdot\, \mathbb{P}(\text{🎲} \mid \text{🎲}) \ldots$$

The *full* joint probability factorises easily:

$$\mathbb{P}(x_{1:T}, y_{1:T}) = \pi[y_1] e_{y_1}[x_1] \cdot \prod_{t=2}^{T} \alpha[y_{t-1}, y_t] e_{y_t}[x_t]$$

Place your bets: Probability of a sequence

- When the sequence of dice types (Hidden states) is known:



2 4 4 6 6 1 3 2 4

$$\mathbb{P}(\text{🎲🎲🎲🎲🎲🎲🎲🎲🎲}, 244661324) = \mathbb{P}(2 \mid \text{🎲})\mathbb{P}(\text{🎲} \mid \text{🎲})\mathbb{P}(4 \mid \text{🎲})\mathbb{P}(\text{🎲} \mid \text{🎲})$$
$$\cdot\, \mathbb{P}(4 \mid \text{🎲})\mathbb{P}(\text{🎲} \mid \text{🎲})\mathbb{P}(6 \mid \text{🎲})$$
$$\cdot\, \mathbb{P}(\text{🎲} \mid \text{🎲}) \ldots$$

The *full* joint probability factorises easily:

$$\mathbb{P}(x_{1:T}, y_{1:T}) = \pi[y_1] e_{y_1}[x_1] \cdot \prod_{t=2}^{T} \alpha[y_{t-1}, y_t] e_{y_t}[x_t]$$

- When the hidden states are unknown we have two main questions:
  - What is the probability of the sequence? $\mathbb{P}(x_{1:T})$
  - What is the most likely hidden state sequence?

    $\arg\max_{y_1,\ldots,y_T} \mathbb{P}(x_{1:T}, y_{1:T})$ or $\mathbb{P}(y_t \mid x_{1:T})$

Probability of a sequence

- Let's compute $\mathbb{P}(2663)$
- Naive solution: Enumerate all states

$$\mathbb{P}(2663) = \mathbb{P}(2663, \square\square\square\square) + \mathbb{P}(2663, \square\square\square\blacksquare) + \mathbb{P}(2663, \square\square\blacksquare\square) + \mathbb{P}(2663, \square\square\blacksquare\blacksquare) + \ldots + \mathbb{P}(2663, \square\blacksquare\blacksquare\blacksquare)$$

$$= \sum_{y_{1:4} \in \{F,L\}^4} \mathbb{P}(2663, y_1, y_2, y_3, y_4)$$

$\rightarrow$ the number of terms in the sum is increasing exponentially ($|\Sigma|^T$)

- Second solution: use the Markov chain hypothesis to factorise the probabilities

Decomposing the likelihood

- We can decompose the likelihood according to the arrival state

$$\mathbb{P}(2663) = \mathbb{P}(2663, y_4 = \square) + \mathbb{P}(2663, y_4 = \square)$$

- Let's decompose once more on y_3:

$$\mathbb{P}(2663, y_4 = \square) = \underbrace{\mathbb{P}(2663, y_4 = \square, y_3 = \square)}_{A} + \underbrace{\mathbb{P}(2663, y_4 = \square, y_3 = \square)}_{B}$$

$$A = \mathbb{P}(x_4 = 6 \mid \cancel{266}, y_4 = \square, \cancel{y_3 = \square})\, \underbrace{\mathbb{P}(266, y_4 = \square, y_3 = \square)}_{\mathbb{P}(\square \mid \square, \cancel{266})\mathbb{P}(266, y_3 = \square)}$$

$$\mathbb{P}(2663, y_4 = \square) = \mathbb{P}(3 \mid \square) \cdot \alpha_{\square,\square} \cdot \mathbb{P}(266, y_3 = \square) + \mathbb{P}(5 \mid \square) \cdot \alpha_{\square,\square} \cdot \mathbb{P}(266, y_3 = \square)$$

- We introduce a *forward* variable f:

$$f_v(t) = \mathbb{P}(x_{1:t}, y_t = v) = e_v[x_t] \sum_{u \in \Sigma} \alpha_{u,v} f_u(t-1)$$

- We can compute the likelihood by filling a $|\Sigma| \times T$ table.

The forward algorithm

t	1	2	3	4
x_t	2	6	6	3
$f_F(t)$	$1/6$	$1/6 \cdot 0.95 \cdot 1/6$ 0.26389	$0.26389 \cdot 0.95 \cdot 1/6 +$ $0.004167 \cdot 0.1 \cdot 1/6$	
$f_L(t)$	0	$1/2 \cdot 0.05 \cdot 1/6$ 0.004167	$0.26389 \cdot 0.05 \cdot 1/6 +$ $0.004167 \cdot 0.9 \cdot 1/2$	

$$f_{\text{⚀}}(2) = e_{\text{⚀}}[2]\left(\alpha_{\text{⚀},\text{⚀}} f_{\text{⚀}}(1) + \alpha_{\text{⚅},\text{⚀}} f_{\text{⚅}}(1)\right) = \mathbb{P}(6 \mid \text{⚀})\,(0.95 \cdot 1/6 + 0.9 \cdot 0)$$

$$f_{\text{⚅}}(2) = e_{\text{⚅}}[2]\left(\alpha_{\text{⚀},\text{⚅}} f_{\text{⚀}}(1) + \alpha_{\text{⚅},\text{⚅}} f_{\text{⚅}}(1)\right) = \mathbb{P}(6 \mid \text{⚅})\,(0.05 \cdot 1/6 + 0.1 \cdot 0)$$

- Forward algorithm:
  - Initialisation: $f_v(1) = \pi(v) e_v[x_1]$
  - For t in $2 : T$:
    - $f_v(t) = e_v[x_t] \cdot \sum_u \alpha_{u,v} f_u[t-1]$
  - Termination: $\mathbb{P}(x_{1:T}) = \sum_v f_v(T)$
  - complexity in $O(|\Sigma| \times T)$

Practical informations

- Practical points for the forward algorithm [Durbin et al., 1998]:
  - the forward values $f_v(t)$ decrease exponentially with t:
    - f_v is a sum over previous values $\rightarrow$ approximation errors.
    - Solution: store *normalized* values $\tilde{f}$, together with the normalisation factor $\tilde{f}_v(t) = \frac{f_v(t)}{\prod_{i=1}^{t} s_i} = \mathbb{P}(y_t = v \mid x_{1:t})$
  - New update formulas:

$$\tilde{f}_v(t+1) = \frac{1}{s_{t+1}} e_v[x_{t+1}] \sum_u \alpha_{u,v} \tilde{f}_u(t)$$

    We normalize s_{t+1} such that $\tilde{f}_v(t+1)$ sums to 1:

$$s_{t+1} = \sum_v e_v[x_{t+1}] \sum_u \alpha_{u,v} \tilde{f}_u(t)$$

  - In practice we even keep the log values $\log(s_t)$.
- An HMM with hidden states Σ and values on $\mathcal{O}$ can be seen as a Markov chain on $\Sigma \times \mathcal{O}$ with transition matrix A:

$$A[(u,a),(v,b)] = \alpha_{u,v} \cdot e_v[b]$$

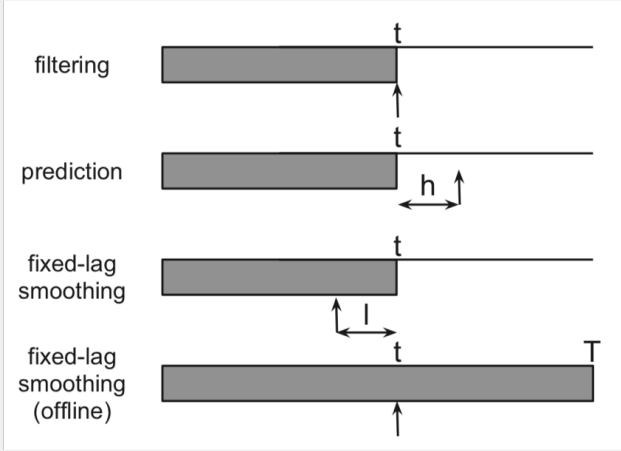
Prediction using HMM

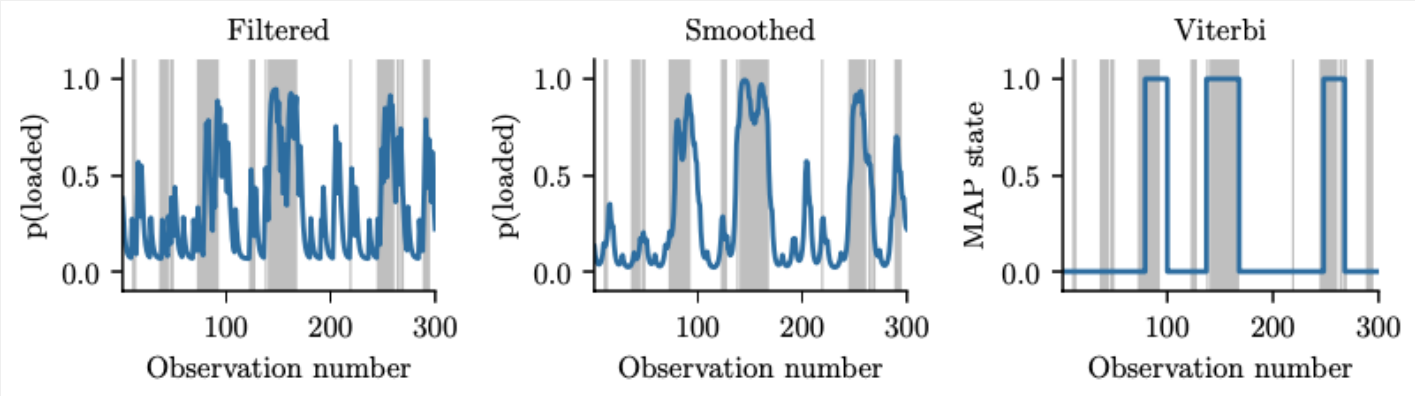
The 3 main questions with an HMM

- **Evaluation**: What is the probability of a sequence?
- **Decoding**: What is the most probable hidden state allocation? Different questions (and algorithms):
  - Most likely state sequence: $\arg\max_{y_{1:T}} \mathbb{P}(x_{1:T}, y_{1:T})$ (Viterbi)
  - Posteriors quantities for y_t:
    - filtering: $\mathbb{P}(y_t = u \mid x_{1:t}) \propto f_u(t)$ (Forward)
    - prediction: $\mathbb{P}(y_{t+h} = v \mid x_{1:t}) \propto \sum_u f_u(t)\alpha^h(u, v)$
    - smoothing: $\mathbb{P}(y_t = u \mid x_{1:T})$ (Forward-Backward)
    - fixed-lag smoothing: $\mathbb{P}(y_{t-\ell} = u \mid x_{1:t})$
- **Estimation**: How to estimate parameters?
  - hidden states observed: easy.
  - If the hidden states are not observed, resort to the EM algorithm (Baum-Welch).

Decoding quantities

- Viterbi: $\arg\max_{y_{1:T}} \mathbb{P}(x_{1:T}, y_{1:T})$
- Posteriors quantities for y_t:
  - filtering: $\mathbb{P}(y_t = u \mid x_{1:t}) \propto f_u(t)$ (forward)
  - prediction: $\mathbb{P}(y_{t+h} = v \mid x_{1:t}) \propto \sum_u f_u(t)\alpha^h(u, v)$
  - smoothing: $\mathbb{P}(y_t = u \mid x_{1:T})$ (forward-backward)
  - fixed-lag smoothing: $\mathbb{P}(y_{t-\ell} = u \mid x_{1:t})$





[Murphy, 2023] figure 9.3

Viterbi algorithm

- Goal: compute the hidden state sequence $y^\star$ such that: $y^\star = \arg\max_{y_{1:T}} \mathbb{P}(x_{1:T}, y_{1:T})$
- Again we will use two recurrence variables:
  - **Max values:** $\delta_u(t) \in]0;1[$ the highest joint probability up to position t, given we arrive on state u:

$$\delta_u(t) = \max_{y_{1:t-1}} \mathbb{P}(y_{1:t-1}, y_t = u, x_{1:t})$$

  - **Backward pointers:** $\psi_u(t) \in \Sigma$ the state at $t-1$ that lead to obtain the score $\delta_u(t)$:

$$\Psi_u(t) = \arg\max_{u \in \Sigma} \left(\delta_u(t-1) \cdot \alpha(u, v)\right)$$

Viterbi algorithm

Initialisation

$$\delta_u(1) = \pi_u e_u(x_1)$$

Recurrence

$$\delta_v(t) = \left[\max_{u \in \Sigma} \delta_u(t-1) \cdot \alpha(u,v)\right] \cdot e_v(x_t)$$

$$\Psi_v(t) = \arg\max_{u \in \Sigma} \delta_u(t-1) \cdot \alpha(u,v)$$

Terminaison

$$S^\star = \max_u \delta_u(T)$$

Backtracking

$$y_T^\star = \arg\max_{u \in \Sigma} \delta_u(T)$$

$$y_t^\star = \Psi_{t+1}(y_{t+1}^\star)$$

(To avoid vanishing quantities, the algorithm works in practice with $\log \delta_u(t)$ and products replaced by sums).

Viterbi on the Casino example

```
Rolls     3151162464466442453211316311641521336251445436316566266566666
Die       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLL
Viterbi   FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL

Rolls     6511664531326512456366646316366631623264552352666666251516 31
Die       LLLLLLFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLFFFFFFFFF
Viterbi   LLLLLLFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF

Rolls     2225554416665665635643243641315134651463534111264146262533 56
Die       FFFFFFFFLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Viterbi   FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL

Rolls     3661636664662325344136616611632525624622552652522664353533 36
Die       LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi   LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Rolls     2331216253644144323351632436336655624666626326666123552452 42
Die       FFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
Viterbi   FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLFFFFFFFFFF
```

[Durbin et al., 1998]

Viterbi example

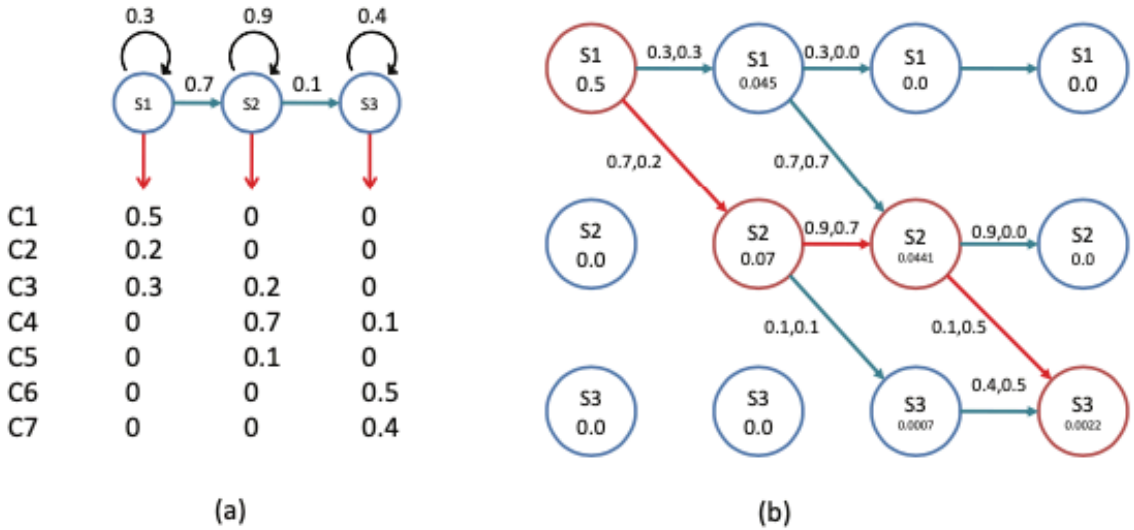


Figure 9.6: Illustration of Viterbi decoding in a simple HMM for speech recognition. (a) A 3-state HMM for a single phone. We are visualizing the state transition diagram. We assume the observations have been vector quantized into 7 possible symbols, $C_1, \ldots, C_7$. Each state S_1, S_2, S_3 has a different distribution over these symbols. Adapted from Figure 15.20 of [RN02]. (b) Illustration of the Viterbi algorithm applied to this model, with data sequence C_1, C_3, C_4, C_6. The columns represent time, and the rows represent states. The numbers inside the circles represent the $\delta_t(j)$ value for that state. An arrow from state i at $t-1$ to state j at t is annotated with two numbers: the first is the probability of the $i \to j$ transition, and the second is the probability of generating observation $\boldsymbol{y}_t$ from state j. The red lines/circles represent the most probable sequence of states. Adapted from Figure 24.27 of [RN95].

[Murphy, 2023] figure 9.6

FORWARD-BACKWARD ALGORITHM FOR STATE POSTERIOR

- Let's decompose the posterior $\mathbb{P}(Y_t = u \mid x_{1:T})$ in order to highlight Markov conditional independencies.
- We note that $\mathbb{P}(Y_t = u \mid x_{1:T}) \propto \mathbb{P}(x_{1:T}, Y_t = u)$ and work with the joint

$$\mathbb{P}(x_{1:T}, Y_t = u) = \underbrace{\mathbb{P}(x_{t+1:T} \mid Y_t = u, \cancel{x_{1:t}})}_{\triangleq b_u(t)} \cdot \underbrace{\mathbb{P}(x_{1:t}, Y_t = u)}_{f_u(t)}$$

b is called the backward variable, and can be updated recursively

$$b_u(t) = \sum_{v \in \Sigma} \alpha_{u,v} e_v(x_{t+1}) b_v(t+1)$$

- The Forward-Backward algorithm computes the posterior $\mathbb{P}(Y_t = u \mid x_{1:T})$ in a forward and a backward pass on the sequence.
  - $\mathbb{P}(Y_t = u \mid x_{1:T})$ is obtained by normalizing $f_u(t) \cdot b_u(t)$ over u.

Estimating Parameters

Parameter estimation with known states

- The log-likelihood of the complete data writes (states Σ, observations $\mathcal{O}$):

$$\begin{aligned}\log \mathbb{P}(x_{1:T}, y_{1:T}; \alpha, (e_u)_u) &= \log \pi_{y_1} + \sum_{t=2}^{T} \log \alpha[y_{t-1}, y_t] + \sum_{t=2}^{T} \log e_{y_t}[x_t] \\ &= \log \pi_{y_1} + \sum_{u \in \Sigma} \sum_{v \in \Sigma} N_{u,v}(\bullet) \log \alpha[u, v] \\ &\quad + \sum_{u \in \Sigma} \sum_{a \in \mathcal{O}} N_u(a) \log e_u[a]\end{aligned}$$

- The maximum likelihood estimators are a generalisation of Markov chains:

$$\hat{\alpha}(u, v) = \frac{N_{u,v}(\bullet)}{N_{u,\bullet}(\bullet)} \qquad \hat{e}_u(a) = \frac{N_u(a)}{N_u(\bullet)}$$

- When the data is incomplete, we need an iterative procedure: the EM algorithm!

Baum-Welch: EM for Hidden Markov Models

Reminder, we generate a sequence of parameters $\Theta^{(1)}, \ldots, \Theta^{(m)}$

- While $|\ell^{(m)} - \ell^{(m-1)}| > \varepsilon$:
  - **E-Step:** Compute the posterior probabilities $Q_u(t)$ and $Q_{u,v}(t)$ for the hidden variables (using forward/backward algorithm):

    $$Q_u(t) = \mathbb{P}(Y_t = u \mid x_{1:T}, \Theta^{(m)}), Q_{u,v}(t) = \mathbb{P}(Y_{t-1} = u, Y_t = v \mid x_{1:T}, \Theta^{(m)})$$

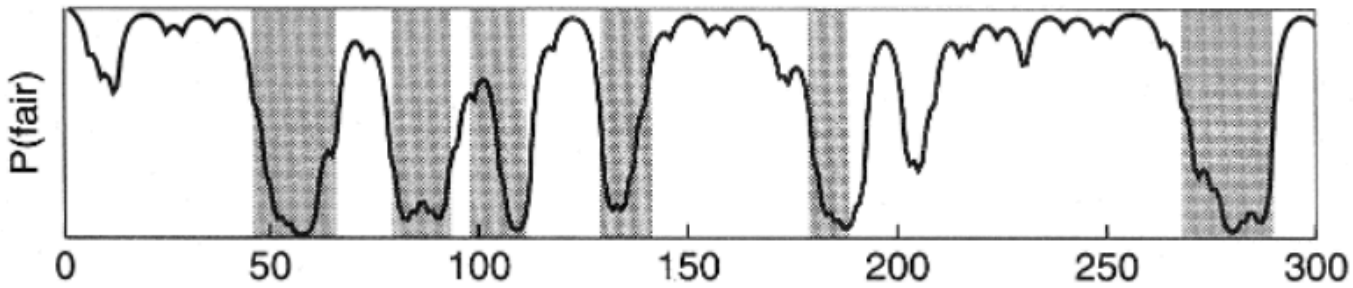
  - **M-Step:** Maximize the conditional likelihood to reestimate parameters

    $$\hat{\alpha}^{(m)}[u, v] = \frac{\sum_{t=2}^{T} Q_{u,v}(t)}{\sum_{t=1}^{T-1} Q_u(t)} \qquad \hat{e}_u^{(m)}[a] = \frac{\sum_{t=1}^{T} Q_u(t) \cdot \mathbb{I}_{\{y_t = a\}}}{\sum_{t=1}^{T} Q_u(t)}$$

  - Compute log-likelihood $\ell^{(m+1)}$ using the forward algorithm $\sum_u f_u(T)$.
- Note: It is possible to run a "hard assignment" version of EM using Viterbi at the E-step (similar to K-means for clustering), see project 4.

Learning goals

- Understand what are Hidden Markov Models (HMM)
  - Make inference/annotation over sequential data with short term dependency and implicit constraints.
- Computing quantities for HMMs
  - Decoding and smoothing allow to make statement about the most likely hidden state of the sequence.
  - Using the EM algorithm, we can **at the same time** estimate the parameters and perform decoding



X-axis = no. of rolls, Y-axis = p (die is fair)
The shaded areas show the rolls generated by loaded die.

More examples of HMMs in biology

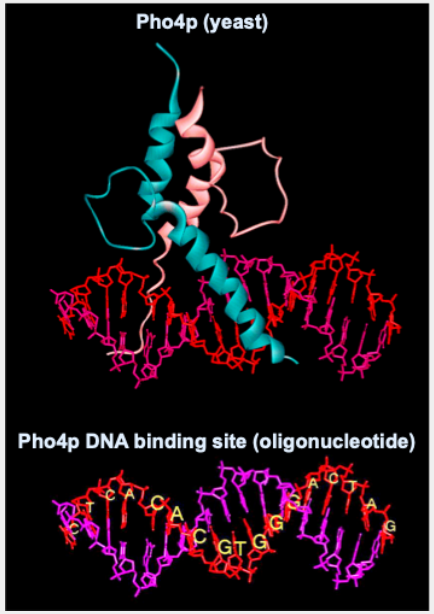
Applications of HMMs to biological sequences

- Modeling/predicting Transcription Factor binding sites
- Gene prediction
- Pairwise sequence alignment
- Profile Hidden Markov models

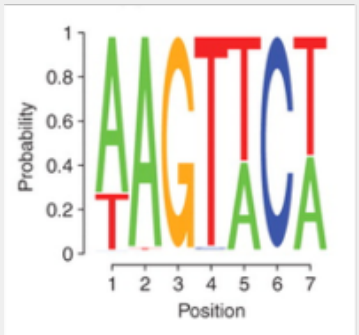
Transcription Factor Binding sites

- Motifs upstream of the coding sequence

Gene	Site Name	Sequence	Affinity
PHO5	UASp2	---aCtCaCA**CACGTGGG**ACTAGC-	high
PHO84	Site D	---TTTCCA**GCACGTGGG**GCGGA--	high
PHO81	UAS	----TTATG**GCACGTGCG**AATAA--	high
PHO8	Proximal	GTGATCGCT**GCACGTGGC**CCGA---	high
group 1	consensus	---------**gCACGTGgg**-------	high
PHO5	UASp1	--TAAATTA**GCACGTTTT**CGC----	medium
PHO84	Site E	----AATA**CGCACGTTTT**TAATCTA	medium
group 2	consensus	--------**cgCACGTTtt**-------	medium
Degenerate consensus		---------**GCACGTKKk**-------	high-med

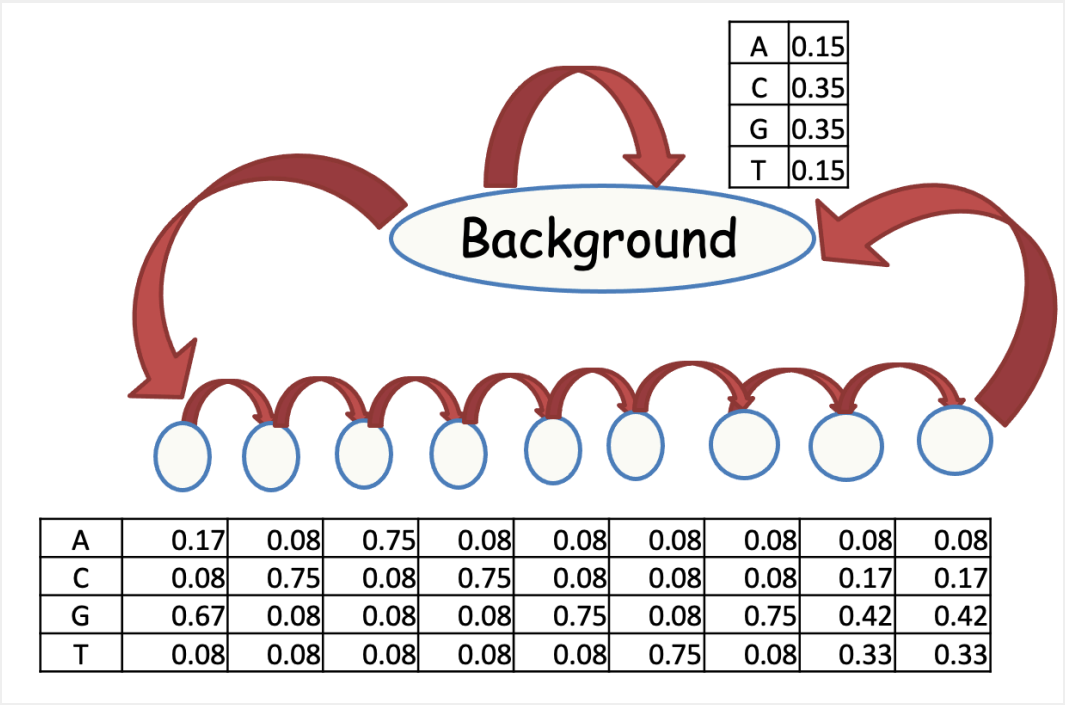


- Usually represented by a Position Weight Matrix (PWM)



Position Weight Matrices and HMM

- A background states models the average probabilities
- A succession of states with different distributions for each position in the motif.

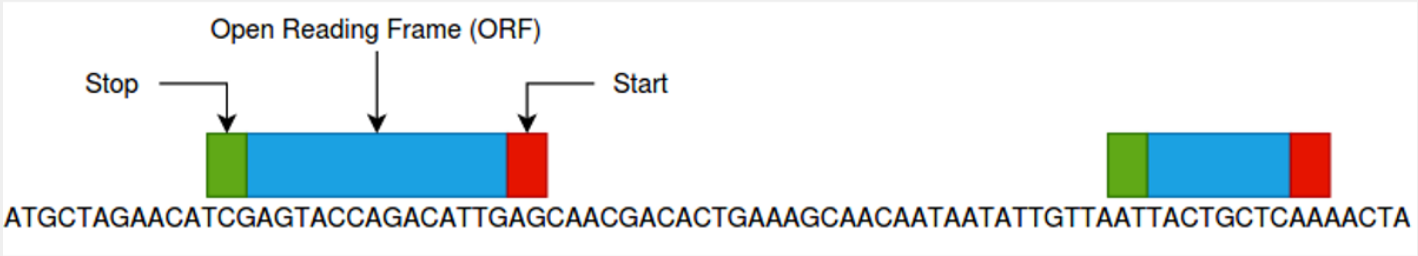


A	0.15
C	0.35
G	0.35
T	0.15

A	0.17	0.08	0.75	0.08	0.08	0.08	0.08	0.08	0.08
C	0.08	0.75	0.08	0.75	0.08	0.08	0.08	0.17	0.17
G	0.67	0.08	0.08	0.08	0.75	0.08	0.75	0.42	0.42
T	0.08	0.08	0.08	0.08	0.08	0.75	0.08	0.33	0.33

Gene Prediction (prokaryotes)

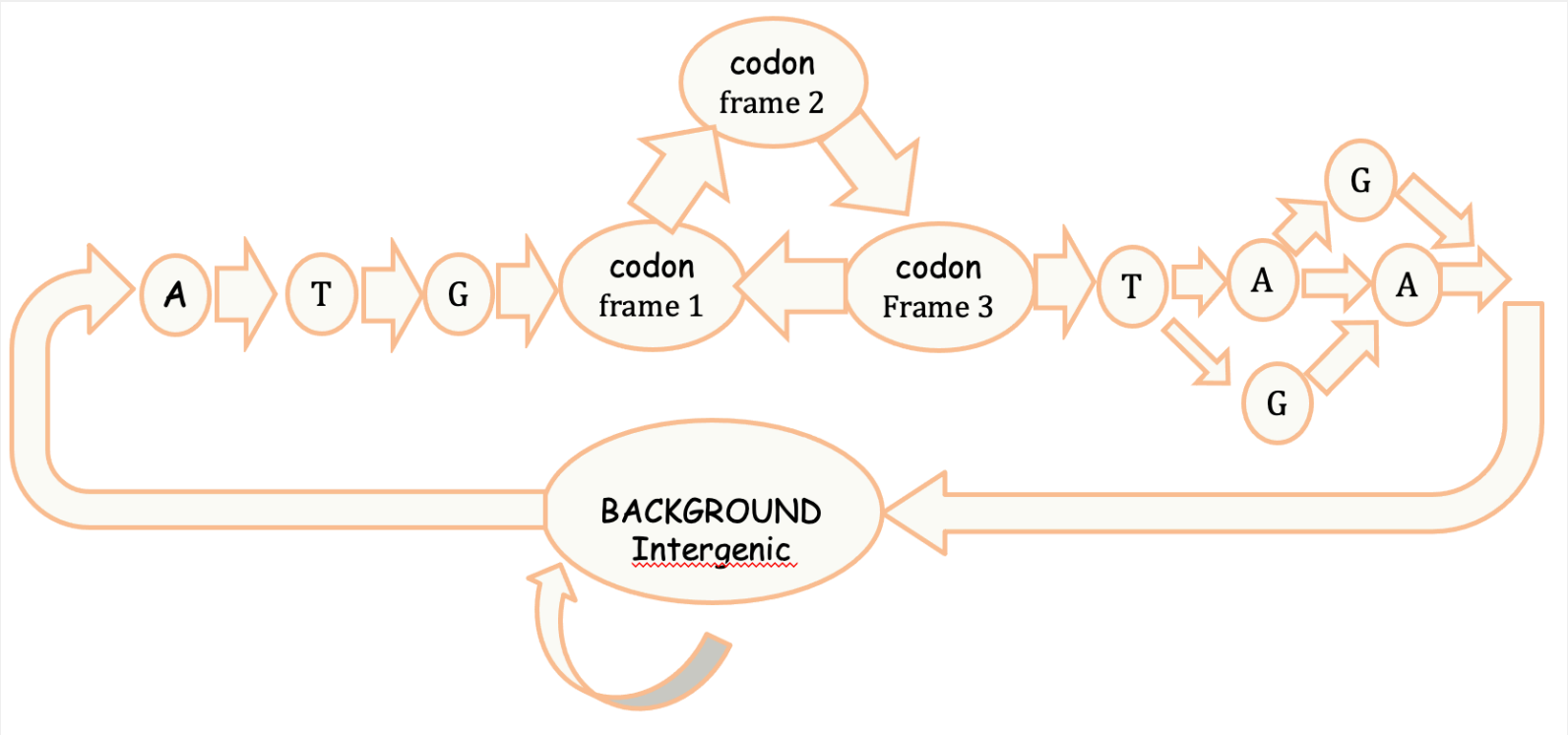
- A gene is a sequence that starts with a start codon and stops with a stop codon.
  - Start: ATG mainly
  - Stop: TAA, TAG, TGA.



- First techniques to predict genes from sequenced genomes:
  - extract all maximal Open Reading Frames
  - Compute a likelihood ratio between codons frequencies in coding and non coding regions

Gene Prediction - 1

- First model: one strand, start and stop codons.



Gene Prediction - 2

- Adding forward and reverse strand.

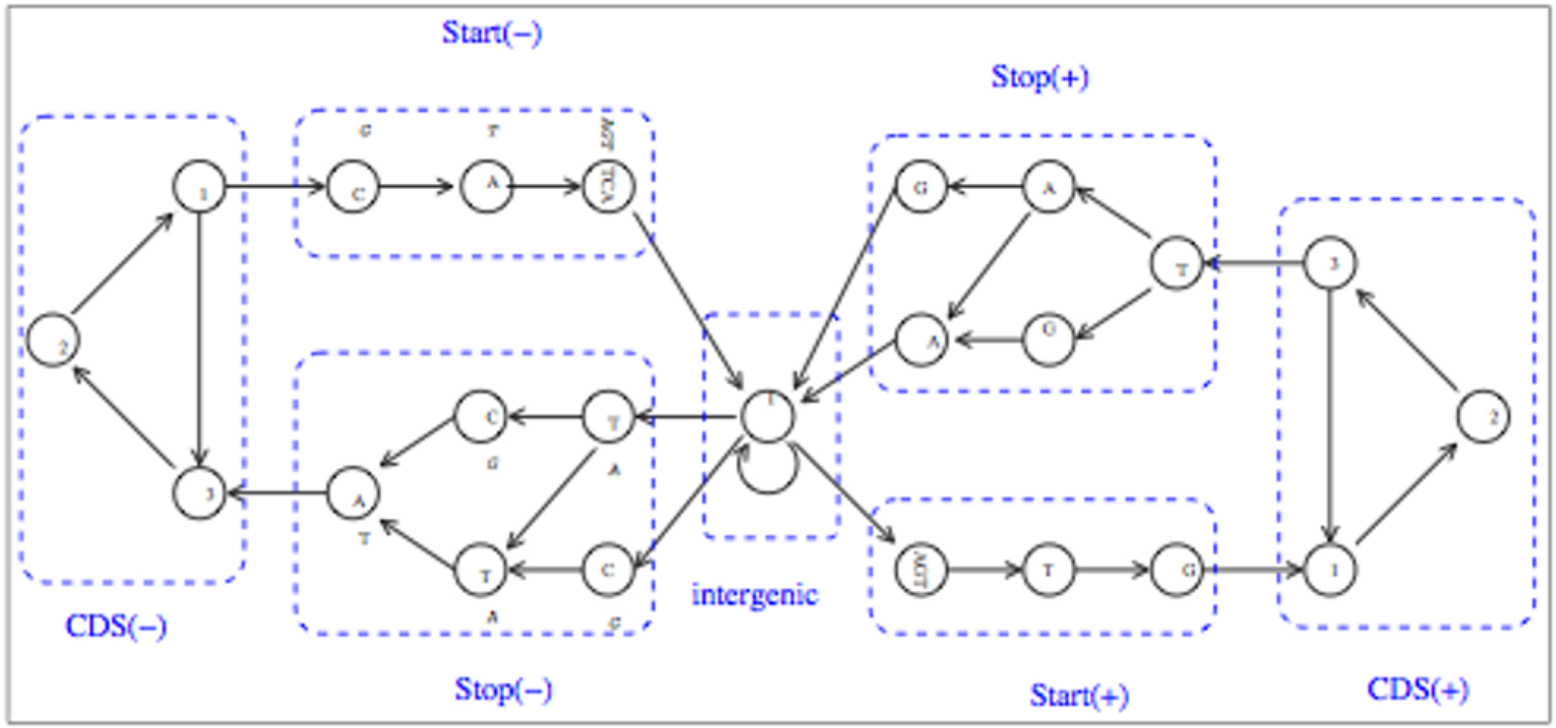
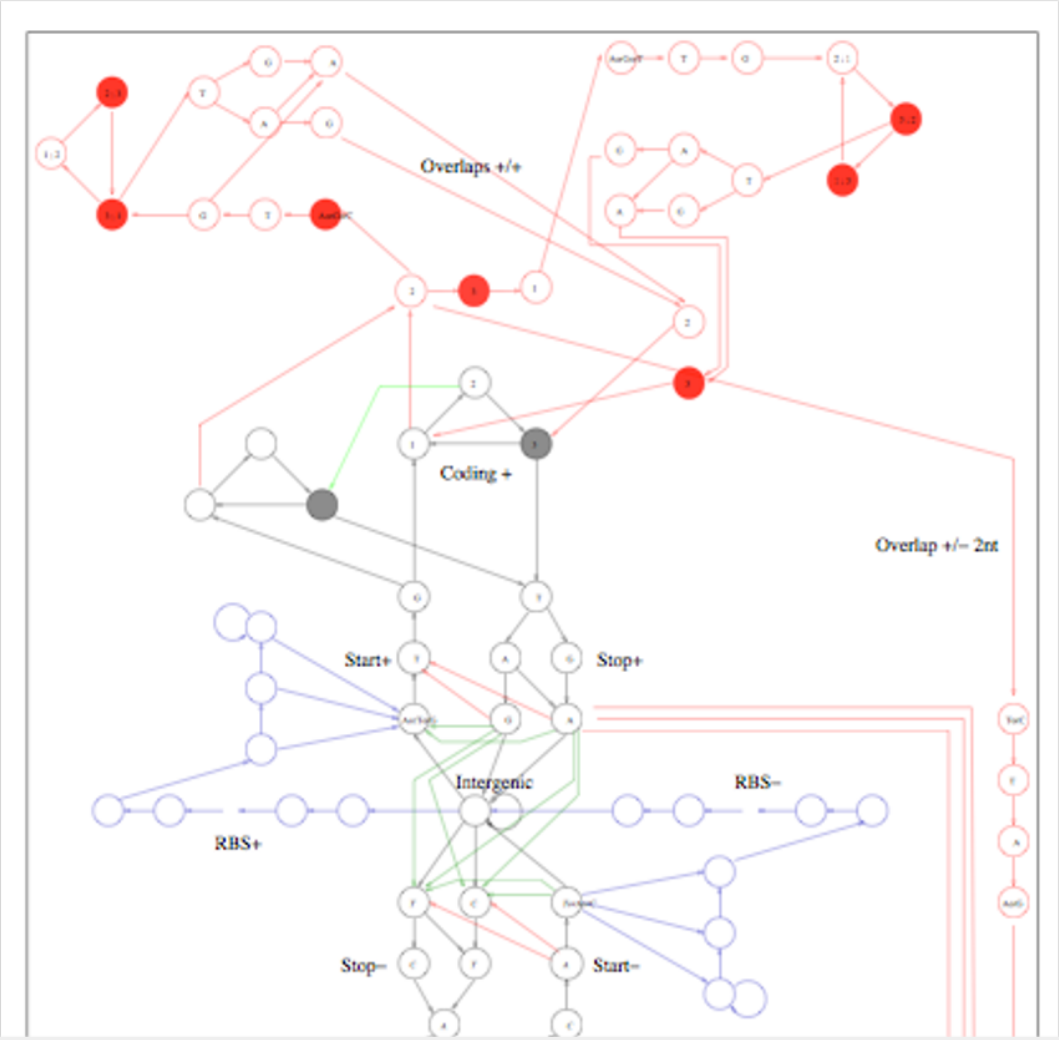


Figure 1: Example of a simple HMM dedicated to bacterial coding sequences detection

Gene Prediction - 3

- More complicated: overlapping genes and Ribosome Binding sites



SHoW, P. Nicolas

References I


Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998).
***Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*.**
Cambridge University Press.


Murphy, K. P. (2023).
***Probabilistic Machine Learning: Advanced Topics*.**
MIT Press.


Russell, S. and Norvig, P. (2003).
***Artificial Intelligence: A Modern Approach*.**
Prentice Hall, 2 edition.