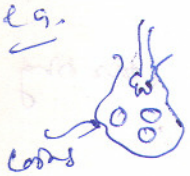
Gene Prediction:

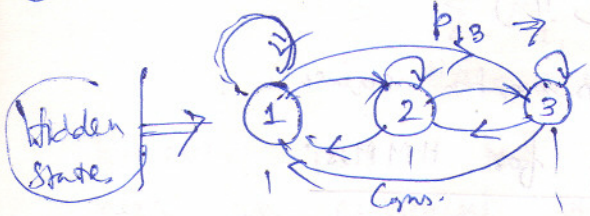
Hidden Markov Models (HMM)

e.g.



Method.

Rob. (1) Draw a coin randomly
(2) Toss it



P_{13} ⇒ Prob. of going to state 3 from 1.
⇒ transition prob.

$P_1(H)$, $P_2(H)$, $P_3(H)$ } ⟶ Individual biases.
$P_1(T)$ $P_2(T)$ $P_3(T)$ }

H/T H/T H/T ⟶ O/P symbols.

O/P sequence depends very much on the individual biases (Prob. of an O/P symbol given a state), the transition probabilities (Prob. of going to state i from j) between various states, as well as on which state is chosen to begin the observations (Prob. of state i as initial state).

- Notations:

$S = \{1, 2, \cdots, N\}$ ⇒ set of N discrete states.

$V = \{v_1, v_2, \cdots, v_M\}$ ⇒ set of M distinct observation symbols.

i_t = state at time t.

O_t = observation symbol at time t.

$A = \{a_{ij}\}$ ⇒ set of transition prob. i.e.
$a_{ij} = P(i_{t+1} = j \mid i_t = i)$
Assume a_{ij}'s are independent of time (or time invariant).

$B = \{ b_j(k) \mid b_j(k) = P(v_k \text{ at } t \mid i_t = j) \}$,

Prob. of observation symbol v_k at state j

$\pi = \{ \pi_i \mid \pi_i = P(i_1 = i) \}$, Prob that the starting state is i.

$\therefore$ HMM $\Rightarrow \lambda = (A, B, \pi)$

$O = O_1, O_2, \cdots, O_T$ as observation sequence.

- The Three problems of for HMMs:

Prob. 1. Given the model, $\lambda = (A, B, \pi)$ how do we compute $P(O|\lambda)$ i.e. the prob. of occurrence of the observation sequence $O = O_1, O_2, \cdots, O_T$.

Prob. 2: Given $\lambda = (A, B, \pi)$, how do we choose a state sequence $I = i_1, i_2, \cdots, i_T$ so that $P(O, I|\lambda)$ is maximised.

$P(O, I|\lambda)$ = Joint prob. of $(\langle O_1, i_1 \rangle, \langle O_2, i_2 \rangle \cdots \langle O_T, i_T \rangle)$ given λ.

Prob. 3.: How do we derive an HMM model $\lambda = (A, B, \pi)$ so that $P(O|\lambda)$ (or $P(O, I|\lambda)$) is maximized.

The first two are analysis problems & the last one is a synthesis problem.

• Prob. 1: $P(I|\lambda) = \pi_{i_1} a_{i_1 i_2} a_{i_2 i_3} \cdots a_{i_{T-1} i_T}$

$$P(O|I,\lambda) = b_{i_1}(O_1)\, b_{i_2}(O_2) \cdots b_{i_T}(O_T)$$

$$\therefore \quad P(O|\lambda) = \sum_{\forall I} P(O|I,\lambda)\, P(I|\lambda)$$

$$= \sum_{\forall I = i_1 i_2 \cdots i_T} \pi_{i_1} b_{i_1}(O_1)\, a_{i_1 i_2} b_{i_2}(O_2) \cdots a_{i_{T-1} i_T} b_{i_T}(O_T)$$

However, $|I| = N^T$. which is a large no. We need an effiecient way to compute $P(O|\lambda)$.

Time Order: $\approx 2TN^T$.

Forward Procedure:

Let, $\alpha_t(i) = P(O_1, O_2, \cdots, O_t, i_t = i \,|\, \lambda)$

Compute $\alpha_t(i)$ inductively:

1. $\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N \Rightarrow N$ times multiplies
2. for $t = 1, 2, \cdots, T-1, \quad 1 \le j \le N,$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\, a_{ij}\right] b_j(O_{t+1}) \Rightarrow N+1 \text{ multiplies}$$

$\Rightarrow$ for $\forall j$, $N(N+1)$ multiplies

$\forall t, j$, $(T-1)N(N+1)$.

$$\therefore \quad P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Time Complexity: $N + N(N+1)(T-1) \approx O(N^2 T)$.

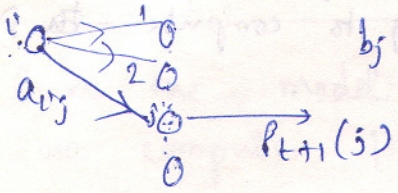
Backward Procedure:

$$\beta_t(i) = P(O_{t+1}\, O_{t+2} \cdots O_T \mid i_t = i, \lambda).$$

1. $\beta_T(i) = 1, \quad 1 \le i \le N$

2. for $t = T-1, T-2, \ldots, 1, \quad 1 \le i \le N$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)$$

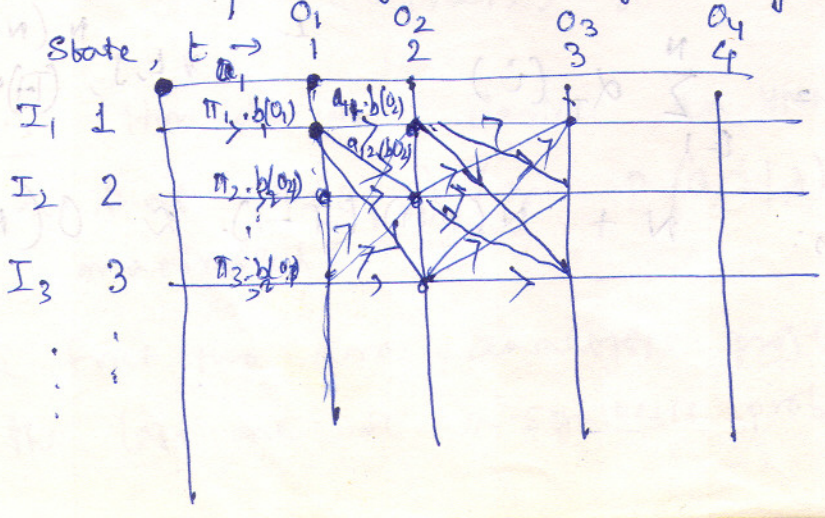


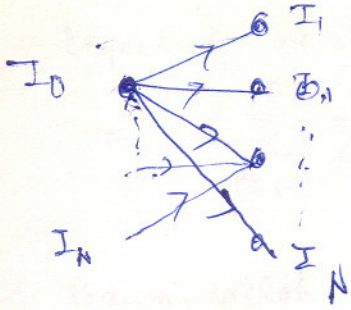
3. $$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i\, b_i(O_1)\, \beta_1(i).$$

Prob. 2. Given O, λ,
Find I to maximise $P(O, I \mid \lambda)$.

Viterbi Algorithm (Viterbi, 1967)

Essentially a Dynamic Programming approach.





At a node

$\delta(I_i, O_t)$ = maximum prob. ~~$P(O_1 O_2 \ldots O_t,$~~ - score vs instant.

$\therefore$ at (I_i, O_{t+1})

$$= \max_{\forall j} \left\{ \delta(I_j, O_t) \cdot a_{ji} \cdot b_i(O_{t+1}) \right\}$$

1. Initial score: $\delta(I_i, O_1) = \pi_i \, b_i(O_1)$, $\forall 1 \le i \le N$

2. For $t = 2, 3, \ldots, T$.

$$\delta(I_i, O_t) = \max_{\forall j} \left(\delta(I_j, O_{t-1}) \cdot a_{ji} \cdot b_i(O_t) \right)$$

$$\text{Prev\_state}(i, t) = \operatorname{argmax}_j (\quad " \quad)$$

3. Final state:

$$P(O, I|\lambda) = \max_j \left(\delta(I_j, O_T) \right)$$

$$\text{Final\_state} = \operatorname{argmax}_j \left(\delta(I_j, T) \right).$$

As, prob. expressed in terms of products, ~~it~~ true cost function can be conveniently expressed by logarithm to bring summation during maximization opn.

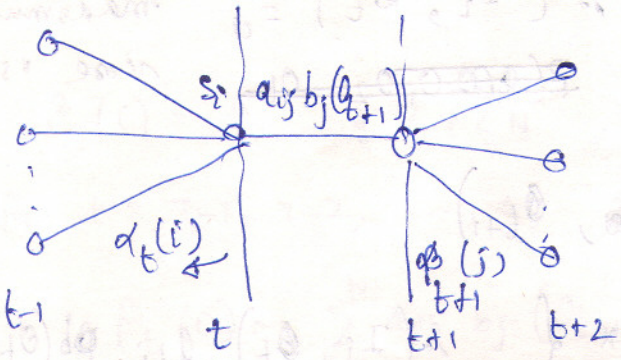
$\therefore \delta(I_i, 1) = \ln(\pi_i \, b_i(O_1))$.

$t = 2, 3, \ldots, T$

$$\delta(I_i, t) = \max_j \left\{ \delta(I_j, t-1) + \ln\{a_{ji} \, b_i(O_t)\} \right\}$$

• Solution to Prb. 3.

~~Prob. 3.~~ Baum-Welch method.



Let, $\xi_t(i,j) = P\left(q_t = i,\ q_{t+1} = j \mid O, \lambda\right)$

$$\therefore \xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}.$$

Note, $P\left(q_t = i,\ q_{t+1} = j,\ O \mid \lambda\right) = \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)$

Let, $\gamma_t(i) = \text{Prob.}\left(q_t = i \mid O, \lambda\right) = \dfrac{\alpha_t(i)\,\beta_t(i)}{P(O|\lambda)}.$

$$\therefore \gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

∴ Expected no. of transitions from i or expected no. of times that a state i is visited

$$= \sum_{t=1}^{T-1} \gamma_t(i) \quad \left[\text{Exclude time slot } T\right]$$

Expected no. of transitions from state i to j

$$= \sum_{t=1}^{T-1} \xi_t(i,j)$$

- Baum-Welch reestimation formulae:

$$\hat{\pi}_i = \gamma_1(i), \quad 1 \le i \le N$$

= expected no. of times state i occurs at $t = 1$.

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

= expected no. of transitions to state j from state i.

$$\hat{b}_j(O_k) = \frac{\sum_{\substack{t=1 \\ \text{s.t. } O_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Alg. 1. Initialise randomly to $\lambda = (A, B, \pi)$.

2. Use reestimation till it converge.

Baum & Welch proved that the convergence is guaranteed w.r.t.

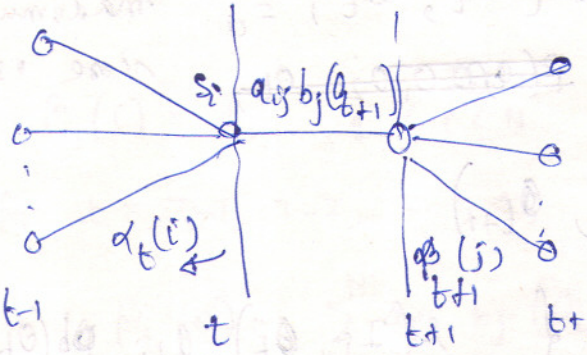
$$P(O \mid \lambda_{(i+1)}) \ge P(O \mid \lambda_i)$$

after $(i+1)^{th}$ iteration.

$i = 0, 1, \ldots$

• Solution to Prb. 3.

~~Prb. 3.~~ Baum-Welch method.



Let, $\xi_t(i,j) = P\left(q_t = i,\ q_{t+1} = j \mid O, \lambda\right)$

$$\therefore \xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}$$

Note, $P\left(q_t = i,\ q_{t+1} = j,\ O \mid \lambda\right) = \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)$

Let, $\gamma_t(i) = \text{Prob.}\left(q_t = i \mid O, \lambda\right) = \dfrac{\alpha_t(i)\, \beta_t(i)}{P(O|\lambda)}$

$$\therefore \gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

∴ Expected no. of transitions from i or expected no. of times that a state i is visited

$$= \sum_{t=1}^{T-1} \gamma_t(i)$$

[Exclude time slot t]

Expected no. of transitions from state i to j

$$= \sum_{t=1}^{T-1} \xi_t(i,j)$$

- Baum-Welch reestimation formulae:

$$\hat{\pi}_i = \gamma_1(i), \quad 1 \le i \le N$$

= expected no. of times state i occurs at $t=1$.

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

= expected no. of transitions to state j from state i.

$$\hat{b}_j(O_k) = \frac{\sum_{\substack{t=1 \\ \text{s.t. } O_{kt} = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Alg. 1. Initialise randomly to $\lambda = (A, B, \pi)$.

2. Use reestimation till at convergence.

Baum & Welch proved that the convergence is guaranteed w.r.t.

$$P(O \mid \lambda_{(v+1)}) \geq P(O \mid \lambda_v)$$

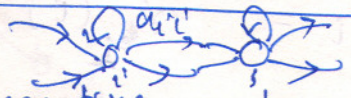
after $(v+1)^{th}$ iteration.

$v = 0, 1, \ldots$

Ref. L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proc. of the IEEE, vol. 77, no. 2, Feb., 1989, p. 257 – 284.

(*) Explicit State duration Density in HMMs:

With conventional HMM:



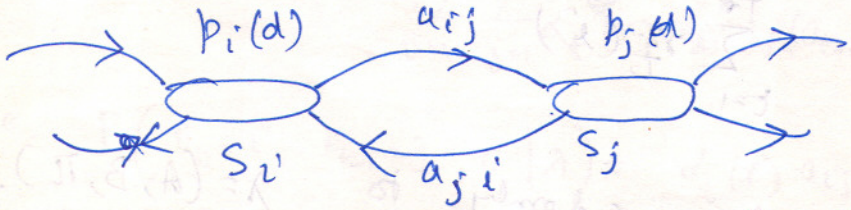
$p_i(d)$ = Prob. of d consecutive observations in state i

$$= (a_{ii})^{d-1} \cdot (1 - a_{ii})$$

⇒ Exponential state duration density model

However, in reality, most physical systems follow different state duration density model.

Here the HMM is modified as:



& $a_{ii} = 0$ for all i.

In fact: $a_{ij} = 0$; $\sum_{j=1}^{N} a_{ij} = 1$.

→ Transition is made only after the appropriate number of observations have occured in the state (as specified by the duration density).

Hence for calculating $P(O|\lambda)$:

$$\alpha_t(i) = P(O_1, O_2 \cdots, O_t, \text{ } S_i \text{ ends at } t \,|\, \lambda)$$

Let us assume that a total r states have been visited during the first t observations & let us denote the states as $q_1, q_2, \cdots, q_r$ with durations associated with each state as $d_1, d_2, \cdots, d_r$.

$$\therefore \quad q_t = S_i$$

$$\& \quad \sum_{s=1}^{r} d_s = t$$

$$\therefore \alpha_t(i) = \sum_{q} \sum_{d} \pi_{q_1} \cdot p_{q_1}(d_1) \cdot P(O_1 O_2 \cdots, O_{d_1} | q_1)$$
$$\cdot a_{q_1 q_2} \cdot p_{q_2}(d_2) \cdot P(O_{d_1+1} \cdots O_{d_1+d_2} | q_2)$$
$$\cdots a_{q_{r-1} q_r} p_{q_r}(d_r) \cdot P(O_{d_1+d_2+\cdots+d_{r-1}+1} \cdots O_t | q_r)$$

By induction:

$$\alpha_t(j) = \sum_{i=1}^{N} \sum_{d=1}^{D} \alpha_{t-d}(i)\, a_{ij}\, p_j(d) \prod_{s=t-d+1}^{t} b_j(O_s) \quad \cdots \text{(A)}$$

where D is the max. duration within any state.

To initialize the computation of $\alpha_t(j)$ we use,

$$\alpha_1(i) = \pi_i \, p_i(1) \cdot b_i(O_1)$$

$$\alpha_2(i) = \pi_i \, p_i(2) \cdot \prod_{s=1}^{2} b_i(O_s) + \sum_{\substack{j=1 \\ j \neq i}}^{N} \alpha_1(j)\, a_{ji}\, p_i(1)\, b_i(O_2)$$

$$\alpha_3(i) = \pi_i \, p_i(3) \prod_{s=1}^{3} b_i(O_s) + \sum_{d=1}^{2} \sum_{\substack{j=1 \\ j \neq i}}^{N} \alpha_{3-d}(j)\, a_{ji}\, p_i(d) \prod_{s=4-d}^{3} b_i(O_s)$$

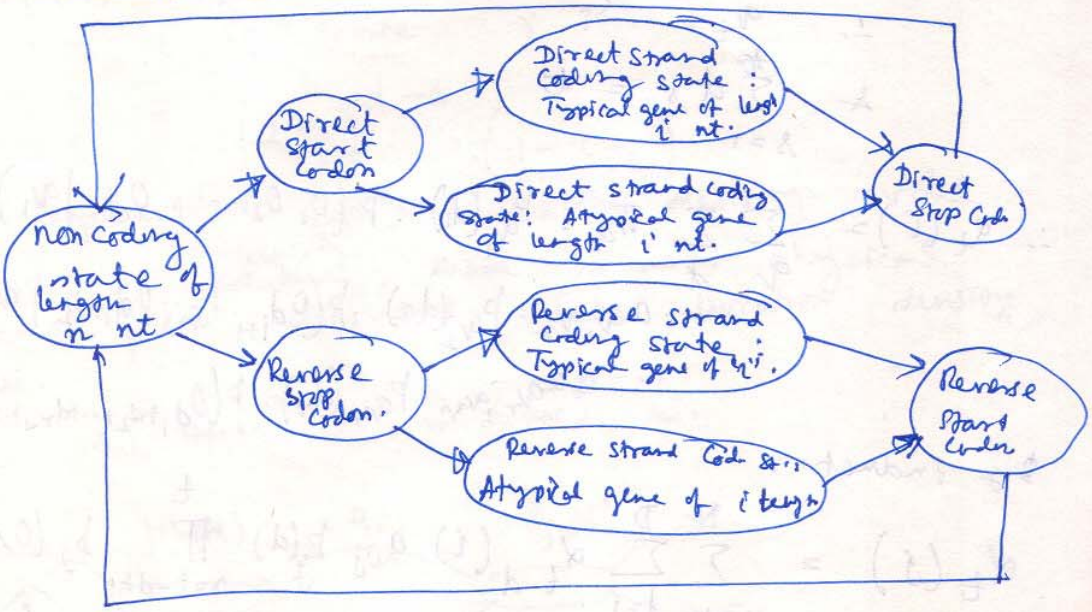
etc. until $\alpha_D(i)$.

Then eqn (A) should be used for all $t > D$.

$$\therefore \quad P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Ref. "GeneMark.hmm: new solutions for gene finding", A. V. Lukashin & M. Borodovsky, Nucleic Acids Research, 1998, vol. 26, no. 4, 1107–1115.

① For prokaryotic nucleotide sequences



"Typical" genes: Vertically transferred.

"Atypical" " : Horizontally " .

Accuracy Measure:

AP = Actual positive (Set of genes say)

AN = " -ve (" ")

[according to Annotation]

PP = Predicted Positive

PN = " -ve

[following HMM prediction]

∴ TP = True Positive = $\{PP \cap AP\}$

∴ TN = " −ve = $\{PN \cap AN\}$

FP = False +ve = $\{PP \cap AN\}$

FN = " −ve = $\{PN \cap AP\}$

∴ Sensitivity = $Sn = \frac{|TP|}{|AP|}$ $|X| \Rightarrow$ cardinality of set X.

& Specificity = $Sp = \frac{|TP|}{|PP|}$

CC = Corr. coeff. = $\frac{(|TP|)(|TN|) - (|FP|)(|FN|)}{\sqrt{(|PP|)(|PN|)(|AP|)(|AN|)}}$

AC = Approximate Corr.

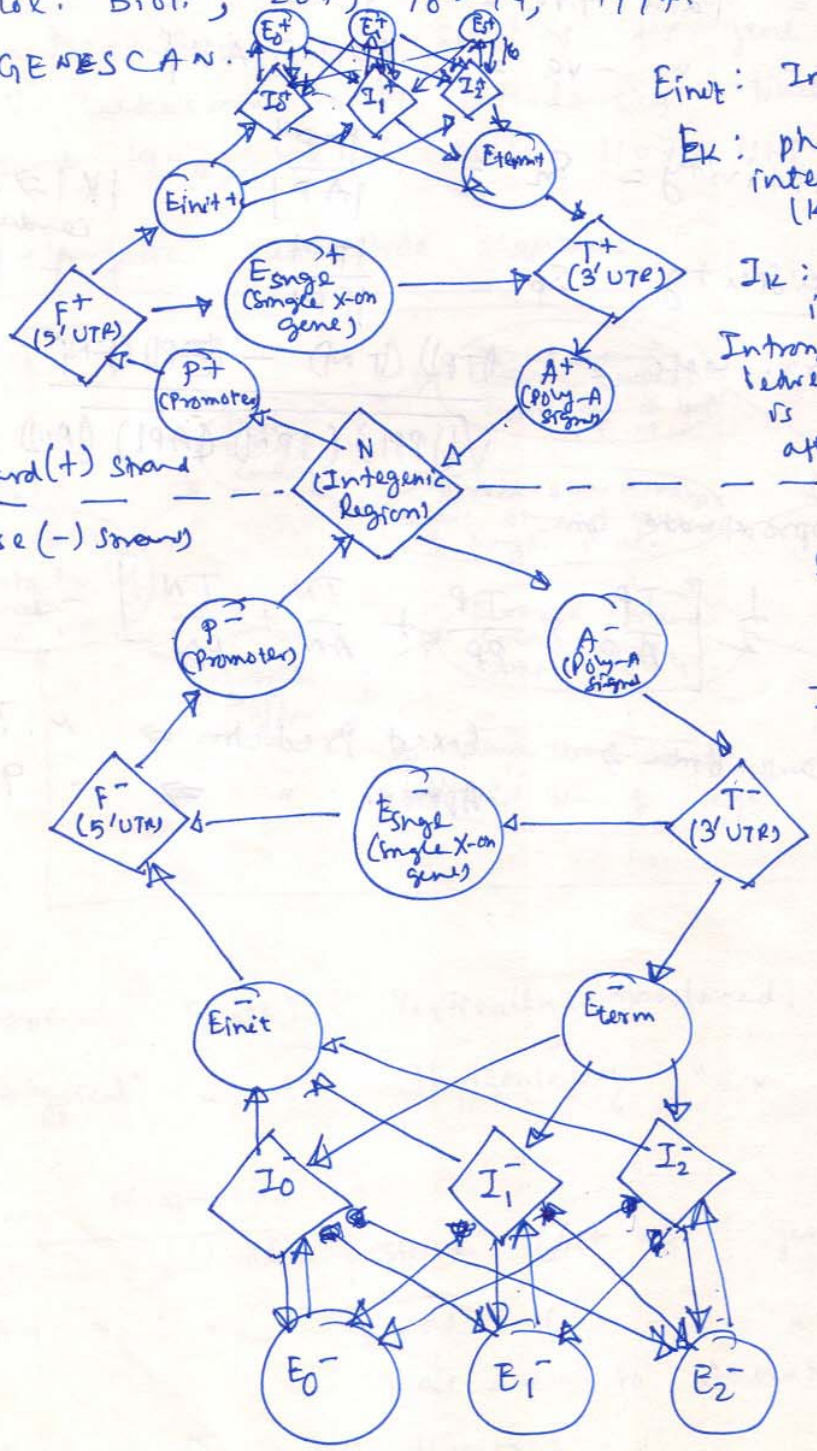
$= \frac{1}{2}\left[\frac{TP}{AP} + \frac{TP}{PP} + \frac{TN}{AN} + \frac{TN}{PN}\right] - 1$

∴ Gene Marker Hmm ⇒ Exact Prediction ⇒ ~ 75%

Approx. " ⇒ ~ 95%

Ref.: "Prediction of Complete Gene Structures in Human Genomic DNA", C. Burge & S. Karlin, J. Mol. Biol., 267, 78-94, 1997.

(1) → GENESCAN.



Einit: Initial Exon

Ek: phase k internal exon (k = 0, 1, 2).

Ik: phase k intron

Introns falls between codons iff at phase 0, after first base of a codon at phase 1 & after 2nd. base of a codon at phase 2.

Internal exons (Ek) are similarly denoted by the phase of its preceding Intron.

Performance: 75-80% exon correctly detected.

GENSCAN ~~Genie Scan~~ Performance:

CC: 0.93 S_n = 0.95 – 0.98 S_p = 0.90 – 0.94

Exons correct: 0.76 – 0.79 ; Exons overlapped 0.91 – 0.96

(*) Similarity-Based Approaches: (Should be discussed with the topic on ~~Similarity~~ Sequence Alignment)

Prob. Defn: Given a known target protein and a genomic sequence, find a set of substrings (candidate exons) of the genomic sequence whose concatenation (splicing) best fits the target.

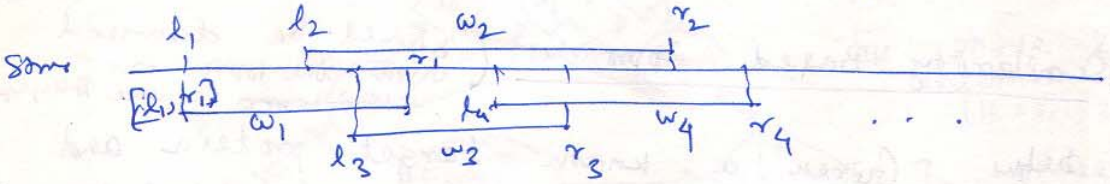
- Find all local similarities between the genomic sequence & the target protein sequence.
- Each substring from the genomic sequence that exhibits sufficient similarity to the target protein could be considered as a putative exon.
- The putative exons so chosen may lack the canonical exon-flanking dinucleotides AG and GT, but we can extend or shorten them slightly to make sure they are flanked by AG & GT:

- The resulting sometime may contain overlapping substring, out of which best possible non-overlapping substrings to be chosen with a maximal match to the target protein

- **Exon - Chaining Problem :**

Weighted Interval : $[l, r] \Rightarrow$ exon & weight w.

$\Rightarrow (l, r, w)$, $w > 0$.



Chain : any set of non-overlapping interval

weight of a chain : Sum of their weights (intervals)

Maximum Chain : having maximum weight among all possible chains.

Problem Defn :

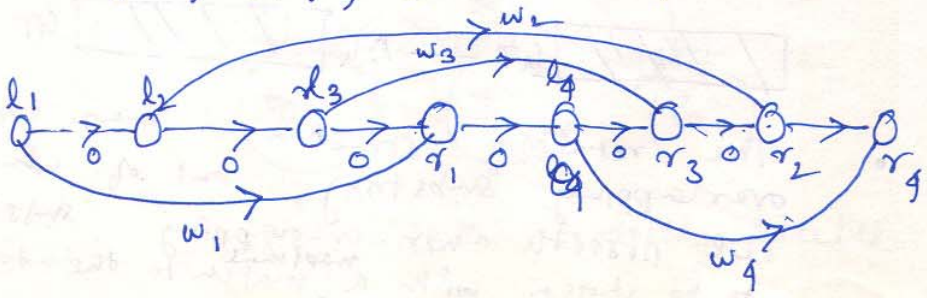
Given a set of putative exons, find a max. set of nonoverlapping putative exons.

Soln : $G(V,E)$: $V = 2n$ vertices., n of which represent left positions & n of which represent right positions (let them be distinct).

Let $V = (v_1, v_2, \ldots, v_{2n})$ be the ordered array of vertices.

E : $\langle l_i, r_i \rangle$. with w_i

& (v_i, v_{i+1}), with 0 weight.

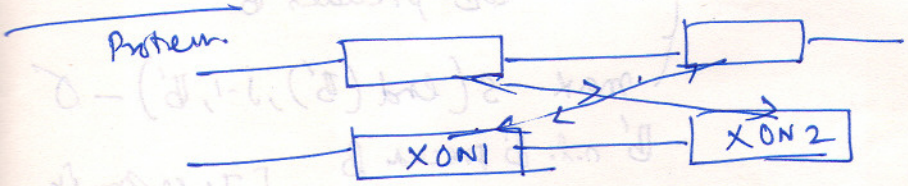


The objective is to find a path from v_1 to v_{2n} with maximal weight s.

Use of dynamic prog.

At vertex v_i, ~~weight~~ $s_i = \max\{s_j + w, s_{i-1})$
s.t. v_j is connected to v_i.
$\leftrightarrow \exists$ an interval $[v_j, v_i]$ with w.

Shortcomings:



An infeasible chain that might have a maximal score.

- Spliced Alignment:

Problem Statement:
Given Genomic seq. G, target seq. T and a set of candidate exon (blocks) $\mathbb{B}$, compute a chain of candidate exons Γ s.t. the global alignment score $s(\Gamma^*, T)$ is maximum among all possible chains of $\mathbb{B}$.

Soln.: Compute,
1. Score $(i,j, \mathbb{b})$, $\mathbb{b} \in \mathbb{B}$, & $G(i) \in \mathbb{b}$.
~~s that~~ $\rightarrow \mathbb{b}$ & maximal score terminating at $G(i)$ ~~for~~ with target seq. $T(0)\ T(1) \cdots T(j)$

using DP.

2. ~~Finally~~, Let end$(\mathbb{b})$ denote the ~~pos~~ end position in G. & m be the length of target seq. T.

∴ maximal spliced score = $\max_{\mathbb{b}} s(end(\mathbb{b}), m, \mathbb{b})$

Recurrence relation.

Let i be the starting vertex of $\mathbb{b}$.

$$\therefore s(0, j, \mathbb{b}) = \max \begin{cases} s(i, j-1, \mathbb{b}) - \sigma & [\text{deletion in the target seq.}] \\ \max_{\mathbb{b}' \text{ s.t. } \mathbb{b}'\mathbb{b} \text{ precedes } \mathbb{b}} s(end(\mathbb{b}'), j-1, \mathbb{b}') + \delta(g_i, t_j) \\ \max_{\mathbb{b}' \text{ s.t. } \mathbb{b}' \text{ precedes } \mathbb{b}} s(end(\mathbb{b}'), j-1, \mathbb{b}') - \sigma \end{cases}$$

[Insertion in genome seq.]

Let i' be an intermediate or end position of $\mathbb{b}$.

$$s(0, j, \mathbb{b}) = \max \begin{cases} s(i, j-1, \mathbb{b}) - \sigma & / \text{ deletion} \\ \max_{\mathbb{b}' \text{ precedes } \mathbb{b}} (s(end(\mathbb{b}'), j-1, \mathbb{b}') + \delta(g_i, \\ s(i', j, \mathbb{b}) - \sigma & / \text{ insertion.} \\ s(i-1, j-1, \mathbb{b}) + \delta(g_i, t_j) \end{cases}$$

N.B. Small Exon blocks should be filtered out, so that misleading effect causing incorrect predictions is avoided.

e.g. "fold" ⇒ could be matched easily with this 20 alphabet length for any random sequence of our alphabets.