Hidden Markov Modelling in Prokaryotic Gene Finding and Sequence Alignment

A thesis submitted in partial fulfillment of the requirements for the award of the degree of
MASTER OF SCIENCE

by

Sweta Mahajan
15MS036

under

Dr.Anirvan Chakraborty

to the
DEPARTMENT OF MATHEMATICS

Indian Institute of Science Education and Research
Kolkata
June, 2020

DECLARATION OF THE STUDENT

I hereby declare that this thesis is my own work and to the best of my knowledge, it contains no materials previously published or written by any other person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at IISER Kolkata or any other educational institution, except where due acknowledgement is made in the thesis.

[illegible]

June 2020 STUDENT SIGNATURE

IISER KOLKATA SWETA MAHAJAN

ACKNOWLEDGEMENT

First of all I pay my gratitude to all the people who taught me how to live life in their small or big ways.

I consider myself extremely lucky to have crossed path with Dr. Anirvan Chakraborty, a man of such intellect, who guided me in completing my thesis and has been a great source of knowledge in my IISER life.

I pay my utmost reverence to my parents who have been sacrificing on their life choices to make me a good human being, inculcate in me good values, to provide me with all kinds of assistance I need for my academics or otherwise and have been very supportive of my choices till date. My brother and sister have supported me morally and academically as and when needed. I will forever be indebted to them for their support.

There have been people whose companionship has helped keep me sane during the hectic and chaotic life at IISER-K. Writing names here would be unjust for I might forget a few but I will forever chesish the moments spent with them.

To my most beloved, my parents, sister and brother

ABSTRACT

This report discusses the foundation on which the entire Hidden Markov Modelling(HMM) is set up and deals with the three problems that this modelling in particular tries to answer and then proceeds to use it in domains like prokaryotic gene finding and pairwise or multiple alignment of biological sequences. The use of HMM in computational biology is immense. HMM is an extension of the Markov models which introduces another layer of flexibility into the system.

In this article we have tried to find out all the proteins of the prokaryote by employing a fairly simple model, after which we cross verify which of the transmembrane proteins mentioned in the uniprot database are actually found by the minimalistic model we have implemented. Then we use these proteins to find out portions of hydrophobic and hydrophilic part and make a note of the resemblance to the labeled sequences in the uniprot database. Next we describe pairwise alignment and multiple sequence alignment and have mould them into the HMM set up to exploit the modelling platform to extract information about the similarity among sequences.

Keywords: Hidden markov modelling, EM algorithm, Baum-Welch algorithm, Viterbi algorithm, AIC, BIC, Transmembrane protein, Open Reading Frames, Acaryochloris Marina, pairwise alignment, substitution matrix, silent states, Needleman Wunsch algorithm, Smith Waterman algorithm, Multiple sequence alignment, Position specific score matrix, profile hmm.

Contents

Chapter 1

Preliminaries: Mixtures and markov chains

1.1 Introduction

Hidden Markov Models are models in which the distribution that generates an observation depends on the state of an underlying and unobserved markov process. This setup provides general purpose models for univariate and multivariate time-series. The marginal distribution of an HMM is conditionally independent i.e., given the underlying states, the marginal distributions are independent .

EXAMPLE: Series of annual counts of major earthquakes

Events like these are usually modelled with discrete distributions with unbounded support such as Poisson but data analysis indicates presence of considerable amount of over dispersion relative to Poisson distribution and serial dependence in the data.
So, a model of independent Poisson random variables would be inappropriate. HMMs, on the other hand which allow the probability distribution of each observation to depend on the unobserved state of a markov chain can accommodate both for over dispersion and serial dependence.

1.2 Independent Mixture Models

1.2.1 Definition and Properties

One method of dealing with over dispersed data with a bimodal or multimodal distribution is to use a mixture model. Mixture models are designed to accommodate unobserved heterogeneity in the population.

NOTATION:
X= observation variable, x=realization of observation variable.
C= state of the Markov chain
$p_i(x) = P(X = x|C = i)$
δ_i=P(C=i)

Suppose each count in the earthquake series is generated by X, where X follows

$$\begin{gathered}
\text{Poisson}(\lambda_1) \text{ with probability } = \delta_1 \\
\text{Poisson}(\lambda_2) \text{ with probability } = \delta_2 \\
\vdots \\
\text{Poisson}(\lambda_m) \text{ with probability } = \delta_m
\end{gathered}$$

The probability/density function of the mixture at any time step t is as follows:

$$\begin{aligned}
P(x) &= \sum_{i=1}^{m} P(X = x, C = i) \\
&= \sum_{i=1}^{m} P(C = i)P(X = x|C = i) \\
&= \sum_{i=1}^{m} \delta_i p_i(x)
\end{aligned}$$

The continuous case is same as above where p represents the density function.
Similarly the expectation of the mixture is as follows;

$$\begin{aligned}
E(X) &= \sum_{i=1}^{m} P(C = i)E(X|C = i) \\
&= \sum_{i=1}^{m} \delta_i \; \mu_i
\end{aligned}$$

The continuous case is same as above where p represents the density function.

Proposition:

$$
\begin{aligned}
P(X) &= \sum_{i=1}^{m} P(C=i)P(X=x|C=i) \\
\Rightarrow E(X) &= \sum_{i=1}^{m} P(C=i)E(X|C=i)
\end{aligned}
$$

Proof:(We consider the discrete observation sequence and the discrete state sequence here. In continuous case the corresponding summation is replaced by integral sign.)

$$
\begin{aligned}
E(X) &= \sum_{j=1}^{n} x_j P(X=x_j) \\
&= \sum_{j=1}^{n} x_j \sum_{i=1}^{m} P(C=i)P(X=x|C=i) \\
&= \sum_{i=1}^{m} P(C=i) \sum_{j=1}^{n} x_j P(X=x|C=i) \\
&= \sum_{i=1}^{m} P(C=i)E(X|C=i)
\end{aligned}
$$

More generally, for a mixture, the k^{th} moment about the origin is as follows:(by the similar line of argument as above)

$$E(X^k) = \sum_{i=1}^{m} \delta_i E(Y_i^k)$$

1.2.2 Parameter Estimation

We have just found out the probability mass function/density function of the mixture, i.e.,

$$P(x) = \sum_{i=1}^{m} \delta_i p_i(x)$$

.

Let $\boldsymbol{\theta}_i$ be the parameters of the i^{th} component of the mixture. So, instead, we write

$$P(x, \boldsymbol{\theta}) = \sum_{i=1}^{m} \delta_i p_i(x, \boldsymbol{\theta}_i)$$

Let $x_1, x_2, \cdots, x_n$ be the sample. Now, we want to find out the likelihood function. We have

$$L(x_1, \cdots, x_n | \boldsymbol{\theta_1, \cdots, \theta_m}) = \prod_{j=1}^{n} \sum_{i=1}^{m} \delta_i p_i(x_j, \boldsymbol{\theta_i})$$

So, the Likelihood function is given by,

$$L(\boldsymbol{\theta_1, \cdots, \theta_m} | x_1, \cdots, x_n) = \prod_{j=1}^{n} \sum_{i=1}^{m} \delta_i p_i(x_j, \boldsymbol{\theta_i})$$

In general, analytical maximization of this complicated function is very difficult. So, we resort to numerical maximization most of the time barring special cases.

1.2.3 Unbounded likelihood

It can happen that, in the vicinity of certain parameter combinations, the likelihood explodes. For example, in case of normal mixture, if we let one component mean equal to one of the observations and allow the corresponding variance to go to zero. If we replace the density by the probability of the interval corresponding to that observation, this issue can be sorted. In the context of independent mixtures, we replace

$$\prod_{j=1}^{n} \sum_{i=1}^{m} \delta_i p_i(x_j, \boldsymbol{\theta_i})$$

by the discrete likelihood,

$$\prod_{j=1}^{n} \sum_{i=1}^{m} \delta_i \int_{a_j}^{b_j} p_i(x, \boldsymbol{\theta_i})\, dx$$

Another way to deal with this is to impose a lower bound on the variances and search for the best local maximum subject to the constraint. One might be fortunate enough to avoid the likelihood spikes when searching for local maximum, where starting with good starting values can help. This scenario does not arise in case of discrete valued observation, since the probability is bounded between 0 and 1.

1.3 Markov Chain

A sequence of discrete random variables $\{C_t : t \in \mathbb{N}\}$ is said to be a Markov chain if for all $t \in \mathbb{N}$, it satisfies the Markov property,

$$P(C_{t+1} | C_t, \cdots, C_1) = P(C_{t+1} | C_t)$$

We will assume at all times that the Markov chain is Homogeneous. And at times to be Stationary.

NOTATION AND DEFINITION:(No. of states=m)

$$\begin{aligned}
\gamma_{ij} &= P(C_{t+1} = j | C_t = i), \qquad 1 \le i \le m, 1 \le j \le m \\
\boldsymbol{\Gamma} &= [\gamma_{ij}] \\
\gamma_{ij} &\ge 0, \qquad\qquad\qquad\qquad 1 \le i \le m, 1 \le j \le m \\
\sum_{j=1}^{m} \gamma_{ij} &= 1, \qquad\qquad\qquad\qquad 1 \le i \le m
\end{aligned}$$

If γ_{ij} is independent of time, then the Markov chain is HOMOGENEOUS.
$\boldsymbol{\delta}$ is said to be a stationary distribution if $\boldsymbol{\delta\Gamma} = \boldsymbol{\delta}$ i.e., $\boldsymbol{\delta}(\mathbf{I_m} - \boldsymbol{\Gamma} - \mathbf{U}) = \mathbf{1}$, where $\mathbf{1}$ is a row vector of ones, $\mathbf{I_m}$ is the $m \times m$ identity matrix and $\mathbf{U}$ is the $m \times m$ matrix of ones.

1.3.1 Autocorrelation Function

Let the states of the Markov chain be $1, 2, 3, \cdots, m$. We assume these states are quantitative not merely categorical.
Define, $\mathbf{v} = (1, 2, \cdots, m)$ and $\mathbf{V} = diag(1, 2, \cdots, m)$. We have, for all $k \in \mathbb{N}$

$$\begin{aligned}
E(C_t C_{t+k}) &= \sum_{i,j=1}^{m} i\ j\ P(C_t = i, C_{t+k} = j) \\
&= \sum_{i,j=1}^{m} i\ j\ P(C_t = i) P(C_{t+k} = j | C_t = i) \\
&= \sum_{i,j=1}^{m} P(C_t = i) i\ \gamma_{ij}\ j \\
&= \boldsymbol{\delta\Gamma^{t-1}V\Gamma^{k}v'} \qquad \text{(Homogeneous markov chain)} \\
&= \boldsymbol{\delta V\Gamma^{k}v'} \qquad \text{(stationary markov chain)}
\end{aligned}$$

$$\begin{aligned}
E(C_t) &= \sum_{i=1}^{m} i P(C_t = i) \\
&= \boldsymbol{\delta\Gamma^{t-1}v'} \qquad \text{(Homogeneous markov chain)} \\
&= \boldsymbol{\delta v'} \qquad \text{(stationary markov chain)}
\end{aligned}$$

$$\begin{aligned}
\text{Cov}(C_t, C_{t+k}) &= E(C_t C_{t+k}) - E(C_t)E(C_{t+k}) \\
&= \boldsymbol{\delta\Gamma^{t-1}V\Gamma^{k}v'} - (\boldsymbol{\delta\Gamma^{t-1}v'})(\boldsymbol{\delta\Gamma^{t-k-1}v'}) \qquad \text{(Homogeneous markov chain)} \\
&= \boldsymbol{\delta V\Gamma^{k}v'} - (\boldsymbol{\delta v'})^2 \qquad \text{(stationary markov chain)}
\end{aligned}$$

Now proceeding further with the stationarity assumption, if $\mathbf{\Gamma}$ is diagonalisable and it's eigenvalues are (other than 1) $\omega_2, \omega_3, \cdots, \omega_m$, then $\mathbf{\Gamma}$ can be written as

$$\mathbf{\Gamma = U \Omega U^{-1}}$$

where, Ω is diag$(1, \omega_2, \omega_3, \cdots, \omega_m)$ and the columns of $\boldsymbol{U}$ are the corresponding to the right eigenvectors of $\mathbf{\Gamma}$. Then we have, for $k \in \mathbb{N} \cup \{0\}$

$$\begin{aligned} Cov(C_t, C_{t+k}) &= \boldsymbol{\delta} \mathbf{V U \Omega}^{\boldsymbol{k}} \mathbf{U^{-1} v'} - (\boldsymbol{\delta} \mathbf{v'})^{\mathbf{2}} \\ &= \mathbf{a \Omega^k b'} - a_1 b_1 \\ &= \sum_{i=2}^{m} a_i b_i \omega_i^k \end{aligned}$$

where $\boldsymbol{a} = \boldsymbol{\delta}\mathbf{VU}$ and $\mathrm{b}' = \mathbf{U^{-1} v'}$. Hence, $Var(C_t) = \sum\limits_{i=2}^{m} a_i b_i$ and, for $k \in \mathbb{N} \cup \{0\}$,

$$\rho(k) = Corr(c_t, c_{t+k}) = \sum_{i=2}^{m} a_i b_i \omega_i^k / \sum_{i=2}^{m} a_i b_i$$

1.3.2 Estimating transition probabilities

For an m-state Markov chain, we have to estimate the $m^2 - m$ parameters(transition probabilities.) from a realization $c_1, c_2, \cdots, c_T$.
Let us take m=5 .
Let us say, we have the observation sequence, S=$\{s_3, s_3, s_3, s_1, s_1, s_3, s_2 | \lambda\}$.

$$\begin{aligned} P(S|\lambda) &= P(s_3 | s_3, s_3, s_1, s_1, s_3, s_2, \lambda) P(s_3, s_3, s_1, s_1, s_3, s_2 | \lambda) \\ &= \gamma_{33} P(s_3 | s_3, s_1, s_1, s_3, s_2, \lambda) P(s_3, s_1, s_1, s_3, s_2 | \lambda)) \end{aligned}$$

In a similar fashion , at the end, we get

$$P(S|\lambda) = \delta_3 \gamma_{33} \gamma_{33} \gamma_{31} \gamma_{11} \gamma_{13} \gamma_{32}$$

More generally, we can write the likelihood as,

$$P(S|\lambda) = \prod_{i=1}^{m} \prod_{j=1}^{m} \gamma_{ij}^{f_{ij}}$$

where, f_{ij}=the number of transitions from state 'i' to state 'j'. The log likelihood is,

$$l = \sum_{i=1}^{m} \left(\sum_{j=1}^{m} f_{ij} \log\ \gamma_{ij} \right) = \sum_{i=1}^{m} l_i \ \text{(Say)}$$

We can maximize l by maximizing each l_i separately.

$$\begin{aligned} l_i &= \sum_{j=1}^{m} f_{ij}\, log(\gamma_{ij}) && (1.1) \\ &= \sum_{i \neq j} f_{ij}\, log(\gamma_{ij}) + f_{ii}\, log(1 - \sum_{i \neq k} \gamma_{ik}) && (1.2) \end{aligned}$$

Now, differentiating l_i w.r.t γ_{ij}, $i \neq j$, we have

$$0 = \frac{f_{ii}}{\gamma_{ij}} - \frac{f_{ii}}{1 - \sum_{i \neq k} \gamma_{ik}} = \frac{f_{ij}}{\gamma_{ij}} - \frac{f_{ii}}{\gamma_{ii}} \tag{1.3}$$

If the denominators in the above expression are not zero, we have

$$f_{ij}\gamma_{ii} = f_{ii}\gamma_{ij} \tag{1.4}$$

Summing over j, we get,

$$\gamma_{ii} \sum_{j=1}^{m} f_{ij} = f_{ii} \tag{1.5}$$

$$\gamma_{ii} = \frac{f_{ii}}{\sum_{j=1}^{m} f_{ij}} \tag{1.6}$$

And putting the value of γ_{ii} in 1.4, we get

$$\gamma_{ij} = \frac{f_{ij}}{\sum_{j=1}^{m} f_{ij}}$$

USING LAGRANGE MULTIPLIER

We want to maximise the function

$$l_i = \sum_{j=1}^{m} f_{ij} log(\gamma_{ij})$$

subject to the constraint $\sum_{j=1}^{m} \gamma_{ij} = 1$

So, now differentiate the function $\sum_{j=1}^{m} f_{ij} log(\gamma_{ij}) - \lambda \sum_{j=1}^{m} \gamma_{ij}$ w.r.t $\gamma_{ij}, i \neq j$, for some λ, and we get

$$\frac{f_{ij}}{\gamma_{ij}} = \lambda \tag{1.7}$$

$$f_{ij} = \lambda\gamma_{ij} \tag{1.8}$$

Summing over j, we get

$$\lambda = \sum_{j=1}^{m} f_{ij}$$

Putting the value of λ in 1.8, we get

$$\gamma_{ij} = \frac{f_{ij}}{\sum_{j=1}^{m} f_{ij}}$$

Now, again differentiating the function w.r.t γ_{ii}, we get

$$\frac{f_{ii}}{\gamma_{ii}} = \lambda \tag{1.9}$$

$$\gamma_{ii} = \frac{f_{ii}}{\sum_{j=1}^{m} f_{ij}} \tag{1.10}$$

Chapter 2

Estimation by direct maximization of the likelihood and EM algorithm

2.1 Introduction

We saw in the previous chapter how we can evaluate the likelihood function inductively by the forward algorithm which merely involves steps of order Tm^2. So, we can estimate the parameters by numerical maximization of the likelihood w.r.t the parameters. In this chapter, we will try to look at the problems associated with this process.
So, the main problems are,

- Numerical Undeflow
- Constraints on the parameters
- Multiple local maxima in the likelihood function

2.2 Problem 1: Scaling the likelihood function(reference $Section$ 3.3)

In the case of discrete state-dependent distribution, the elements of α_t becomes progressively very smaller as t increases, eventually it becomes intractable by the computer and is rounded to zero. Since the likelihood is a product of matrices, not product of scalars, we can not just take log of the likelihood and get done with it.

We prefer, to compute the log of the likelihood, L_T by scaling the forward probabilities α_t at each step. So, we scale the α_ts at each time step so that its elements add up to 1.

Define,

$$\boldsymbol{\phi}_t = \frac{\boldsymbol{\alpha}_t}{w_t} \qquad t = 0, 1, 2, \cdots, T$$

where $w_t = \sum_{i=1}^{m} \alpha_t(i) = \boldsymbol{\alpha}_t \mathbf{1}'$ and $\boldsymbol{\alpha}_0 = \boldsymbol{\delta}$.

Now, we have

$$L_T = \boldsymbol{\alpha}_T \mathbf{1}' = w_T \boldsymbol{\phi}_T \mathbf{1}'$$

Since, $\boldsymbol{\phi}_T$ is a probability, $\boldsymbol{\phi}_T \mathbf{1}' = 1$ and we have $L_T = w_T$.

Now, notice that,

$L_T = \frac{w_1}{w_0}\frac{w_2}{w_1} \cdots \frac{w_T}{w_{T-1}}$ because, $w_0 = \sum_{i=1}^{m} \delta_i = 1$ and $\boldsymbol{\phi}_1 = \boldsymbol{\delta}\mathbf{P}(x_1)$. So,

$$log(L_T) = \sum_{t=1}^{T} log(w_t/w_{t-1}) = \sum_{t=2}^{T} log(\boldsymbol{\phi}_{t-1}\boldsymbol{\Gamma}\mathbf{P}(x_t)\mathbf{1}')$$

where, the second equality comes from the fact that for $t = 2, 3, \cdots, T$

$$\begin{aligned}
\boldsymbol{\alpha}_t &= \boldsymbol{\alpha}_{t-1}\boldsymbol{\Gamma}\mathbf{P}(x_t) \\
w_t\boldsymbol{\phi}_t &= w_{t-1}\boldsymbol{\phi}_{t-1}\boldsymbol{\Gamma}\mathbf{P}(x_t) \\
w_t\boldsymbol{\phi}_t\mathbf{1}' &= w_{t-1}\boldsymbol{\phi}_{t-1}\boldsymbol{\Gamma}\mathbf{P}(x_t)\mathbf{1}' \\
w_t/w_{t-1} &= \boldsymbol{\phi}_{t-1}\boldsymbol{\Gamma}\mathbf{P}(x_t)\mathbf{1}'
\end{aligned}$$

The above process can be written compactly in an algorithm form as follows:

```
$w_1 = \boldsymbol{\delta}\mathbf{P}(x_1)\mathbf{1}'; \boldsymbol{\phi}_1 = \boldsymbol{\delta}\mathbf{P}(x_1)/w_1; l = log(w_1)$

for $t = 2, 3, \cdots, T$
        $\mathbf{v} \longleftarrow \boldsymbol{\phi}_{t-1}\boldsymbol{\Gamma}\mathbf{P}(x_t)$
        $u \longleftarrow \mathbf{v}\mathbf{1}'$
        $l \longleftarrow l + log(u)$
        $\boldsymbol{\phi} \longleftarrow \mathbf{v}/u$
    return $l$
```

2.3 Problem 2: Maximization of the likelihood subject to constraints

Sometimes we have to maximize the likelihood functions subject to constraints. However, depending on the nature of the data and the implementation, the constrained optimization can be slow and hence we have to transform the parameters to unconstrained ones. To understand the method better, let us consider the case of Poisson model.

The relevant constraints are

- the means λ_i of the state-dependent distributions must, for $i = 1, 2, \cdots, m$, be non-negative
- the rows of the transition probability matix Γ must add to 1, and all the parameters γ_{ij} must be non-negative.

For the first problem, we define, $\eta_i = log\lambda_i$, for $i = 1, \cdots, m$. Then $\eta_i \in \mathbb{R}$. After we have maximized the likelihood w.r.t the unconstrained parameter, we invert the function to get the original estimation i.e., $\hat{\lambda}_i = exp(\hat{\eta}_i)$

The solution of the second problem is not that easy and straight forward. We see that Γ has m^2 entries but only $m(m-1)$ of them are free parameters since we have m row sum constraints. We will show here one possible transformation(lgit transformation.) For the sake of readability, let us display the $n = 3$ case. We define the matrix,

$$\begin{bmatrix} - & \tau_{12} & \tau_{13} \\ \tau_{21} & - & \tau_{23} \\ \tau_{31} & \tau_{32} & - \end{bmatrix}$$

Let $g : \mathbb{R} \longrightarrow \mathbb{R}^+$ be a strictly increasing(for the sake of inversion of the trandformation function), for example, take

$$g(x) = exp(x)$$

Define,

$$\nu_{ij} = \begin{cases} g(\tau_{ij}) & \text{for} i \neq j \\ 1 & \text{for} i = j \end{cases}$$

We then set, $\gamma_{ij} = \nu_{ij} / \sum_{k=1}^{m} \nu_{ik} (\text{for} i, j = 1, 2, \cdots, m)$ and $\Gamma = [\gamma_{ij}]$.

- $\nu_{ij} > 0 \forall i, j$, we have $\gamma_{ij} > 0$
- $\sum_{j=1}^{m} \gamma_{ij} = \nu_{ij} / \sum_{k=1}^{m} \nu_{ik}$

So, Γ is a transition probability matrix.

Now, let us write τ_{ij} in terms of γ_{ij}.

$$\gamma_{ij} = \frac{exp(\tau_{ij})}{1+\sum_{i\neq k} exp(\tau_{ik})} \quad i \neq j \tag{2.1}$$

$$\implies \sum_{i\neq j} \gamma_{ij} = \frac{\sum_{i\neq j} exp(\tau_{ij})}{1+\sum_{i\neq k} exp(\tau_{ik})} \tag{2.2}$$

$$\implies 1-\sum_{i\neq j} \gamma_{ij} = 1-\left[\frac{\sum_{i\neq j} exp(\tau_{ij})}{1+\sum_{i\neq k} exp(\tau_{ik})}\right] \tag{2.3}$$

$$\implies 1-\sum_{i\neq j} \gamma_{ij} = \frac{1}{1+\sum_{i\neq k} exp(\tau_{ik})} \tag{2.4}$$

$$\implies 1+\sum_{i\neq k} exp(\tau_{ik}) = \frac{1}{1-\sum_{i\neq j}\gamma_{ij}} \tag{2.5}$$

$$\implies \gamma_{ij}(1+\sum_{i\neq k} exp(\tau_{ik})) = \frac{\gamma_{ij}}{1-\sum_{i\neq j}\gamma_{ij}} \quad i \neq j \tag{2.6}$$

$$\implies exp(\tau_{ij}) = \frac{\gamma_{ij}}{1-\sum_{i\neq k}\gamma_{ik}} \quad i \neq j \tag{2.7}$$

$$\implies \tau_{ij} = log\left(\frac{\gamma_{ij}}{1-\sum_{i\neq k}\gamma_{ik}}\right) \quad i \neq j \tag{2.8}$$

$$\implies \tau_{ij} = log(\frac{\gamma_{ij}}{\gamma_{ii}}) \tag{2.9}$$

2.4 Problem 3:Multiple local maxima in the likelihood

The likelihood of HMM is a very complicated function of its parameters and has several local maxima. We have to find out the local maximum but their is no general methodology to find it out or to say whether the local maximum is global. Depending on the starting values, we might reach the local maximum, but not the global. A sensible strategy is therefore to use a range of starting values for the maximization, and to see if the same maximum is reached in every case.

2.5 Other problems

2.5.1 Starting values for the iterations

It is easy to find plausible starting values for the parameters of an HMM, for example, if one seeks to fit a Poisson-HMM with two states, and the sample mean is 20, we could use the component means as 20-k or 20+k. We can follow a more systematic approach based on quantile, say, for a 3-component HMM, we could use the lower quartile, median and the upper quartile as the component means.

Choosing good initial values can prevent numerical instability. As we saw in case of independent mixtures, the case of unbounded likelihood could be prevented by using the corresponding interval probability instead of the likelihood.

2.6 Standard errors and confidence intervals

In case of MLEs, we have asymptotic results and to exploit these, requires knowladge of the variance-covariance matrix of the esimators of the parameters. To estimate thestandard errors, we can use

- The Hessian of the log-likelihood at the maximum, but this approach runs into difficulties when some of the parameters are on the boundary of the parameter space.
- Parametric bootstrap

2.6.1 Standard errors via Hessian

Under certain regularity conditions, the MLEs of HMM parameters are consistent, asymptotically normal. So, if we can estimate the errors associated with the estimators, then using asymptotic normality, we can compute the confidence intervals.

Let $\boldsymbol{\theta_0}$ be the true parameter and $\hat{\theta}$ is the MLE. Let us expand the log likelihood around the true value,

$$\frac{\partial \mathcal{L}}{\partial \theta}\Big|_{\hat{\boldsymbol{\theta}}} = 0 \tag{2.10}$$

$$\implies \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{\boldsymbol{\theta_0}} + \frac{\partial^2 \mathcal{L}}{\partial \theta \partial \theta'}\Big|_{\boldsymbol{\theta_0}} (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta_0}) = 0 \tag{2.11}$$

$$\implies (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta_0}) = -\frac{\partial \mathcal{L}}{\partial \theta}\Big|_{\boldsymbol{\theta_0}} \left[\frac{\partial^2 \mathcal{L}}{\partial \theta \partial \theta'}\Big|_{\boldsymbol{\theta_0}}\right]^{-1} \tag{2.12}$$

$$\implies var(\hat{\boldsymbol{\theta}}) = E\left[\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta_0}\right)\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta_0}\right)\right] \tag{2.13}$$

$$\implies var(\hat{\boldsymbol{\theta}}) = E\left[\left[\frac{\partial^2 \mathcal{L}}{\partial \theta \partial \theta'}\right]^{-1} \frac{\partial \mathcal{L}}{\partial \theta}\frac{\partial \mathcal{L}'}{\partial \theta}\left[\frac{\partial^2 \mathcal{L}}{\partial \theta \partial \theta'}\right]^{-1}\right] \tag{2.14}$$

$$\implies var(\hat{\boldsymbol{\theta}}) = -\left[E\left[\frac{\partial^2 \mathcal{L}}{\partial \theta \partial \theta'}\right]^{-1}\right] \tag{2.15}$$

Standard errors of $\hat{\theta}$ from the square roots of the diagonal elements of the matrix. We know, the fisher information is given by:

$$I(\boldsymbol{\theta}) = -E\left[H(\boldsymbol{\theta})\right]$$

where H is the Hessian matrix.
Cramer Rao theorem: Given the regularity conditions($\boldsymbol{\theta}$-Unbiased estimator)

$$var(\boldsymbol{\theta}) \geq \left(-E\left[H(\boldsymbol{\theta})\right]^{-1}\right)$$

So, MLE is Unbiased Minimum Variance Estimator(UMVE).

The Hessian w.r.t the working parameters is,

$$H = -\left(\frac{\partial^2 l}{\partial \phi_i \partial \phi_j}\right),$$

but what we really need is the Hessian w.r.t the natural parameters,

$$G = -\left(\frac{\partial^2 l}{\partial \theta_i \partial \theta_j}\right),$$

One problem with this process is, we have found out the Hessian matrix w.r.t the working parameters but what we need is w.r.t the naturala parameters. However, we have the following relationship between the two matrices at the minimum:

$$H = MGM' \quad \text{and} G^{-1} = M'H^{-1}M,$$

where M is defined by $m_{ij} = \partial\theta_j / \partial\phi_i$

First find the MLE by solving constrained optimization problem, then rerun the optimization without constraints, starting at values close to the MLE. If this MLE is same as the MLE previously found, the corresponding Hessian then directly gives the standard errors w.r.t the natural parameters.

We know the Hessian gives information about the curvature of the function. The more curved the likelihood function, the more certainity we have regarding estimating the right parameter. And also, we can see the variance is inverse of the Hessian. So, larger the hessian, lesser is the variance.

2.6.2 Parametric bootstrap

1. Fit the model, estimate the parameters Θ by $\widehat{\Theta}$.

2. Generate a sample, called bootstrap sample from the just fitted model in (1), keep the length of this sample same as the original sample.

3. Now again, fit this model, estimate the parameters Θ by $\widehat{\Theta}^*$ for the bootstrap sample.

4. Repeat the previous two steps for 'k' times, where 'k' is a large number.

5. The variance-covariance matrix of $\widehat{\Theta}$ is then estimated by the sample variance-covariance matrix of the bootstrap estimates $\widehat{\Theta}^*(i), i = 1, \cdots, k$.

$$\widehat{Var - Cov}(\widehat{\Theta}) = \frac{1}{k-1}\sum_{i=1}^{k}\left(\widehat{\Theta}^*(i) - \Theta_0\right)'\left(\widehat{\Theta}^*(i) - \Theta_0\right)$$

where, $\Theta_0 = \frac{1}{k}\sum_{i=1}^{k}\widehat{\Theta}^*(i)$

Chapter 3

Hidden Markov Models

3.1 Introduction

Usually what we have done in, say regression, we have done the prediction of one data point independent of the previous points and so on. But there are lots of applications where the data comes from a sequence, and the prediction you make for one data point is correlated with the prediction of the next data point. For example, in speech recognition, prediction of one word depends on the previous word(s). So, to model these kind of problems, we introduce Hidden Markov Models which also increase the accuracy of prediction.

Problems

1. Efficiently evaluating the probability of observation sequence given the model.
2. Determination of sequence of states which best explains the given observation sequence.
3. Adjustment of model parameters so as to best account for the observed sequence.

Elements of HMM(Notation):

There are 'm' number of states and 'n' number of states.

- $\gamma_{ij} = P(c_{t+1} = j | c_t = i)$
- $p_j(k) = P(x_t = k | c_t = j)$
- $\delta_i = P(c_1 = i)$
- $\delta_i(t) = P(c_t = i)$
- $c_1^t = c^{(t)} = (c_1, c_2, \cdots, c_t)$

3.2 Properties of Markov chain and HMM

Key assumptions:

$$\begin{aligned} P(c_t|c^{(t-1)}) &= P(c_t|c_{t-1}) \\ P(x_t|x^{(t-1)}, c^{(t)}) &= P(x_t|c_{t-1}) \end{aligned}$$

The first assumption is that of the markov chain. The second assumption tells that c_t is the only variable of the markov chain that affects the distribution of x_t. In other words, the distribution of x_t only depends on the state c_t.

For example, the distribution of x_t may be poisson, whose mean is determined by c_t or x_t may follow a normal distribution whose mean and variance are determined by c_t. Here the observation x_t is only observed and the Markov chain c_t is not. So, all the statistical inference, even on the Markov chain itself has to be done in terms of x_t only.

Proposition 3.2.0.1 $P(x^{(T)}, c_t = i) = \alpha_i(t)\beta_i(t)$

Proof:

$$\begin{aligned} P(x^{(T)}, c_t = i) &= P(x^{(T)}|c_t)P(c_t) \\ &= P(x^{(t)}|c_t)P(x_{t+1}^T|c_t)P(c_t) \\ &= P(x^{(t)}, c_t)P(x_{t+1}^T|c_t) \\ &= \alpha_t(i)\beta_t(j) \end{aligned}$$

Proposition 3.2.0.2 $P(c_t = j|x^{(T)}) = \alpha_t(j)\beta_t(j)/L_T \qquad t = 1, \cdots, T$

Proof:

$$P(c_t = j|x^{(T)}) = P(c_t = j, x^{(T)})/P(x^{(T)})$$

Using Proposition 3.2.0.1 , we have

$$= \alpha_i(t)\beta_i(t)/L_T$$

Proposition 3.2.0.3 $P(c_{t-1} = j, c_t = k|x^{(T)}) = \alpha_{t-1}(j)\gamma_{jk}p_k(x_t)\beta_t(k)/L_T$

Proof:

$$\begin{aligned} P(c_{t-1} = j, c_t = k|x^{(T)}) &= P(c_{t-1} = j, c_t = k, x^{(T)})/P(x^{(T)}) \\ &= P(x^{(t-1)}, x_t^T|c_{t-1} = j, c_t = k)P(c_t = k|c_{t-1} = j)P(c_{t-1} = j)/L_T \\ &= P(x^{(t-1)}|c_{t-1} = j)P(x_t^T|c_t = k)\gamma_{jk}P(x_t^T|c_t = k)/L_T \\ &= \alpha_{t-1}(j)\gamma_{jk}p_k(x_t)\beta_t(k)/L_T \end{aligned}$$

3.2.1 Marginal Distributions

We shall often need the marginal distribution of X_t or joint marginal distribution, such as that of (X_t, X_{t+k}). Here it is done for discrete state dependent distribution, the continuous case can be derived similarly.

Univariate Distributions

$$
\begin{aligned}
P(x_t) &= \sum_{i=1}^{m} P(c_t = i) P(x_t | c_t = i) \\
&= \sum_{i=1}^{m} \delta_i(t) p_i(x_t)
\end{aligned}
$$

This expression can be conveniently written in matrix form as,

$$
\begin{aligned}
P(x_t) = [\delta_1(t), \cdots, \delta_m(t)] & \begin{bmatrix} p_1(x_t) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_m(x_t) \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\
&= \boldsymbol{\delta}(t) \mathbf{P}(x_t) \mathbf{1}' \\
&= \boldsymbol{\delta} \boldsymbol{\Gamma}^{t-1} \mathbf{P}(x_t) \mathbf{1}' \quad \text{Homogeneous markov chain} \\
&= \boldsymbol{\delta} \mathbf{P}(x_t) \mathbf{1}' \qquad \text{Stationary Markov chain}
\end{aligned}
$$

Bivariate Distributions

To calculate many distributions in HMM, we always introduce the states in the Markov chain and sum over the states or to put it another way, in a directed graphical model, the joint distribution of a set of random variables V_i is given by,

$$
P(V_1, V_2, \cdots, V_n) = \prod_{i=1}^{m} P(V_i | Pa(V_i))
$$

where $Pa(V_i)$ denotes the set of all parents of V_i in the set $V_1, V_2, \cdots, V_n$. For example,

$$
P(x_t, x_{t+k}, c_t, c_{t+K}) = P(c_t) P(x_t | c_t) P(c_{t+k} | c_t) P(x_{t+k} | c_{t+k})
$$

and hence

$$
\begin{aligned}
P(x_t, x_{t+k}) &= \sum_{i=1}^{m} \sum_{j=1}^{m} P(x_t, x_{t+k}, c_t = i, c_{t+k} = j) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{m} P(c_t) P(x_t | c_t) P(c_{t+k} | c_t) P(x_{t+k} | c_{t+k}) \\
&= \sum_{i=1}^{m} \sum_{j=1}^{m} \delta_i(t) p_i(x_t) \gamma_{ij} p_j(x_{t+k}) \\
&= \boldsymbol{\delta}(t) \mathbf{P}(x_t) \boldsymbol{\Gamma}^k \mathbf{P}(x_{t+k}) \mathbf{1}'
\end{aligned}
$$

Similarly, we have expressions for higher order marginal distributions,

$$P(x_t, x_{t+k}, x_{t+k+l}) = \boldsymbol{\delta(t)}\mathbf{P}(x_t)\boldsymbol{\Gamma}^k\mathbf{P}(x_{t+k})\boldsymbol{\Gamma}^l\mathbf{P}(x_{t+k+l})\mathbf{1}'$$

3.3 Solution: Problem 1

$$\begin{aligned}
P(X_1^T|\lambda) &= \sum_{c_1,\cdots,c_T} P(X_1^T, C_1^T|\lambda) \\
&= \sum_{c_1,\cdots,c_T} \delta_{c_1}p_{c_1}(x_1)\gamma_{c_1c_2}p_{c_2}(x_2)\gamma_{c_2c_3}\cdots\gamma_{c_{T-1}c_T}p_{c_T}(x_T)
\end{aligned}$$

Proof:

$$\begin{aligned}
P(X_1^T|\lambda) &= \sum_{c_1} P(X_1^T, c_1|\lambda) \\
&= \sum_{c_1} P(c_1|\lambda)P(X_1^T|c_1,\lambda) \\
&= \sum_{c_1} \delta_1 P(x_1|c_1,\lambda)P(x_2^T|c_1,\lambda) \\
&= \sum_{c_1,c_2} \delta_1 p_{c_1}(x_1)P(X_2^T, c_2|c_1,\lambda) \\
&= \sum_{c_1,c_2} \delta_1 p_{c_1}(x_1)P(c_2|c_1\lambda)P(x_2^T|c_1,c_2,\lambda) \\
&= \sum_{c_1,c_2} \delta_1 p_{c_1}(x_1)\gamma_{c_1c_2}P(x_2^T|c_2,\lambda) \\
&= \sum_{c_1,c_2} \delta_1 p_{c_1}(x_1)\gamma_{c_1c_2}P(x_2|c_2,\lambda)P(x_3^T|c_2,\lambda) \\
&= \sum_{c_1,c_2} \delta_1 p_{c_1}(x_1)\gamma_{c_1c_2}p_{c_2}(x_2)P(x_3^T|c_2,\lambda)
\end{aligned}$$

And in a similar way, you add the state c_3 and then c_4 till c_T, to reach to the end form

$$P(X_1^T|\lambda) = \sum_{c_1,\cdots,c_T} \delta_{c_1}p_{c_1}(x_1)\gamma_{c_1c_2}p_{c_2}(x_2)\gamma_{c_2c_3}\cdots\gamma_{c_{T-1}c_T}p_{c_T}(x_T) \tag{3.1}$$

$$= \boldsymbol{\delta}\mathbf{P}(x_1)\boldsymbol{\Gamma}\mathbf{P}(x_2)\boldsymbol{\Gamma}\mathbf{P}(x_3)\cdots\boldsymbol{\Gamma}\mathbf{P}(x_T)\mathbf{1}' \tag{3.2}$$

$$= \boldsymbol{\delta}\boldsymbol{\Gamma}\mathbf{P}(x_1)\boldsymbol{\Gamma}\mathbf{P}(x_2)\boldsymbol{\Gamma}\mathbf{P}(x_3)\cdots\boldsymbol{\Gamma}\mathbf{P}(x_T)\mathbf{1}' \quad \text{(Stationary distribution)} \tag{3.3}$$

This algorithm is called FORWARD ALGORITHM to find out the likelihood.

To get an idea of how well this method will perform, let us calculate the order of complexity:

- #of multiplication in each summand=2T-1
- #of state sequence=m^T
- #of total multiplications=$m^T(2T-1)$
- #of additions=m^T-1
- #of total computations=$m^T(2T-1)+m^T-1$

If we increase the number of observations, it increases exponentially and if we increase the number of states, it increases with a very higher rate as the exponent is usually very large. Hence, the process makes it computationally unfeasible.

3.3.1 Forward Algorithm:

So, to help combat the previous problem, we introduce here, the forward algorithm. Let us define,

- Definition: $\alpha_t(i) = P(x_1^t, c_t = s_i|\lambda) \qquad 1 \leq j \leq m, 1 \leq t \leq T-1$
- Initialization: $\alpha_1(i) = \delta_i p_i(x_1) \qquad 1 \leq i \leq m$
- Induction step:
  $\alpha_{t+1}(j) = \Big[\sum_{i=1}^m \alpha_t(i)\gamma_{ij}\Big]p_j(x_{t+1}) \qquad 1 \leq j \leq m, 1 \leq t \leq T-1$
- Final step: $P(X_1^T|\lambda) = \sum_{i=1}^m \alpha_T(i) \qquad 1 \leq i \leq m$

Proof of Induction step:

$$\alpha_{t+1}(j) = P(X_1^{t+1}, c_{t+1} = j|\lambda) \tag{3.4}$$

$$= \sum_{i=1}^m P(X_1^t, x_{t+1}, c_t = i, c_{t+1} = j|\lambda) \tag{3.5}$$

$$= \sum_{i=1}^m P(X_1^t|x_{t+1}, c_t = i, c_{t+1} = j, \lambda)P(x_{t+1}, c_t = i, c_{t+1} = j|\lambda) \tag{3.6}$$

$$= \sum_{i=1}^m P(X_1^t|c_t = i, \lambda)P(x_{t+1}|c_t = i, c_{t+1} = j|\lambda)P(c_t = i, c_{t+1} = j|\lambda) \tag{3.7}$$

$$= \sum_{i=1}^m P(X_1^t|c_t = i, \lambda)P(x_{t+1}|c_{t+1} = j, \lambda)p(c_{t+1} = j|c_t = i, \lambda)P(c_t = i|\lambda) \tag{3.8}$$

$$= \sum_{i=1}^m \alpha_t(i)p_j(x_{t+1})\gamma_{ij} \tag{3.9}$$

$$= \sum_{i=1}^m \alpha_t(i)\gamma_{ij}p_j(x_{t+1}) \tag{3.10}$$

$$\tag{3.11}$$

Now let us again calculate the complexity of this algorithm.

- #of multiplication in initialization=m
- #of multiplication in induction step=$m * m * (T-1) + m * (T-1)$
- #of addition in induction step=$(m-1) * m * (T-1)$
- #of addition in final step=m-1

So, as we can see no of computations $O(m * T^2)$ which is computationally much more faster.
Now lets us convert the algorithm in terms of the matrices which makes it easier to implement through coding.

3.3.2 Forward algorithm via Matrices:

As we have seen in the previous section, the likelihood is given by:

$$L_T = \boldsymbol{\delta}\mathbf{P}(x_1)\boldsymbol{\Gamma}\mathbf{P}(x_2)\boldsymbol{\Gamma}\mathbf{P}(x_3)\cdots\boldsymbol{\Gamma}\mathbf{P}(x_T)\mathbf{1}'$$

If $\boldsymbol{\delta}$ is the stationary distribution of the Markov chain, then we have

$$L_T = \boldsymbol{\delta}\boldsymbol{\Gamma}\mathbf{P}(x_1)\boldsymbol{\Gamma}\mathbf{P}(x_2)\boldsymbol{\Gamma}\mathbf{P}(x_3)\cdots\boldsymbol{\Gamma}\mathbf{P}(x_T)\mathbf{1}'$$

So, now we redefine the previous algorithm in terms of matrices.

- Definition:$\boldsymbol{\alpha}_t = \boldsymbol{\delta}\mathbf{P}(x_1)\boldsymbol{\Gamma}\mathbf{P}(x_2)\boldsymbol{\Gamma}\mathbf{P}(x_3)\cdots\boldsymbol{\Gamma}\mathbf{P}(x_t)$
- Initialization:$\boldsymbol{\alpha}_1(i) = \boldsymbol{\delta}\mathbf{P}(x_t)$
- Induction step:$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t\boldsymbol{\Gamma}\mathbf{P}(x_{t+1}) \qquad t = 1,\cdots,T-1$
- Final step:$P(X_1^T|\lambda) = \sum\limits_{i=1}^{n} \boldsymbol{\alpha}_T(i)$

where, $\mathbf{P}(x_t) = diag(p_1(x_t), p_2(x_t), \cdots, p_n(x_t))$

3.3.3 Backward probability:

In a similar fashion, we define the backward probability.

- Definition: $\beta_t(i) = P(x_{(t+1)}^t | c_t = s_i, \lambda)$
- Initialization: $\beta_T(i) = 1 \quad 1 \le i \le m$
- Induction step:
  $\beta_t(j) = \Big[\sum_{i=1}^m \beta_{t+1}(i)\gamma_{ij}\Big] p_i(x_{t+1}) \qquad 1 \le j \le m, 1 \le t \le T-1$

Proof of induction step:

$$\begin{aligned}
\beta_t(i) &= P(X_{t+1}^T | c_t = i, \lambda) \\
&= P(X_{t+1}^T, c_t = i|\lambda)/P(c_t = i|\lambda) \\
&= \sum_{j=1}^{m} P(x_{t+1}, X_{t+2}^T, c_{t+1} = j, c_t = i|\lambda)/P(c_t = i|\lambda) \\
&= \sum_{j=1}^{m} P(x_{t+1}, X_{t+2}^T | c_t = i, c_{t+1} = j, \lambda)P(c_{t+1} = j|c_t = i, \lambda)P(c_t = i|\lambda)/P(c_t = i|\lambda) \\
&= \sum_{j=1}^{m} P(x_{t+1}, x_{t+2}^T | c_{t+1} = j, \lambda)\gamma_{ij} \\
&= \sum_{j=1}^{m} P(x_{t+2}^T | c_{t+1} = j, \lambda)P(x_{t+1}|c_{t+1} = j, \lambda)\gamma_{ij} \\
&= \sum_{j=1}^{m} \beta_{t+1}(j)p_j(x_{t+1})\gamma_{ij}
\end{aligned}$$

3.4 Solution: Problem 2

The solution to find the state sequence which best explains the observation sequence brings to the surface the criterion of optimality. To begin with, we define optimality as to choose the states c_t which are individually most likely.

$$\begin{aligned}
P(c_t = i|X_1^T, \lambda) &= P(c_t = i, X_1^T|\lambda)/P(X_1^T|\lambda) \\
&= \alpha_i(t)\beta_i(t)/L_T \qquad \text{(From proposition 3.2.0.1)}
\end{aligned}$$

Notice that this is a probability measure and we choose

$$c_t = \underset{1 \le i \le m}{\operatorname{argmax}} P(c_t = i|X_1^T, \lambda) \qquad t = 1, \cdots, T$$

This process maximizes the expected number of correct states. The problem with this approach is that it does not take into account the whole state sequence but each state individually. But, there might be consecutive states whose corresponding transition probability might be zero and it does not make sense.

So, the solution to this problem is to modify the definition of the optimality criterion. Now instead of choosing individual states, we choose the state sequence which makes the observation sequence most likely i.e., maximizing $P(C_1^T|X_1^T, \lambda)$ which is equivalent to maximizing $P(C_1^T, X_1^T|\lambda)$. An algorithm exists for finding out this optimal state sequence, based on dynamic programming, and is called the ”viterbi algorithm”.

Viterbi Algorithm
To find the best sequence of states given the observation sequence, we define

$$\pi_t(i) = \max_{c_1, c_2, \cdots, c_T} P(c_1, c_2, \cdots, c_t, x_1, x_2, \cdots, x_t | \lambda)$$

i.e., $\pi_t(i)$ is the highest probability along a single state sequence, at time t, which accounts for the first t observations and ends in state s_i. By induction we have

$$\pi_{t+1}(j) = \Big[\max_{1 \le i \le m} \pi_t(i)\gamma_{ij}\Big] p_j(x_{t+1})$$

we also need to keep track of the state sequences at each step which makes the corresponding probability maximum and retrieve theses steps at the end. For this reason we need to define another matrix where we have to store the states. The complete procedure is as follows:

- Initialization:

$$\begin{aligned} \pi_1(i) &= \delta_i p_i(x_1) \qquad 1 \le i \le m \\ \psi_i &= 0 \end{aligned}$$

- Induction:

$$\begin{aligned} \pi_t(j) &= \Big[\max_{1 \le i \le m} \pi_{t-1}(i)\gamma_{ij}\Big] p_j(x_t) \qquad 1 \le i \le m, 2 \le t \le T \\ \psi_t(j) &= \Big[\operatorname*{argmax}_{1 \le i \le m} \pi_{t-1}(i)\gamma_{ij}\Big] \qquad\qquad 1 \le i \le m, 2 \le t \le T \end{aligned}$$

- Termination:

$$\begin{aligned} p &= \max_{1 \le i \le m} \Big[\pi_T(i)\Big] \\ c_T &= \operatorname*{argmax}_{1 \le i \le m} \Big[\pi_T(i)\Big] \end{aligned}$$

- state backtracking:

$$q_t = \psi_{t+1}(c_{t+1}) \qquad 1 \le t \le T-1$$

3.5 Solution: Problem 3

Coming up with the best model is by far the most difficult task. There is no analytic way to solve for the model which maximizes the given observation sequence. Given any finite observation sequence as training data, there is no best way to estimate the model parameters globally but we can always locally maximise the probability using the iterative Baum-Welch algorithm(EM algorithm for HMM).

3.5.1 EM algorithm

The EM algorithm is an iterative method for performing maximum likelihood estimation when some of the data are missing and exploit the fact that the complete data log likelihood may be straight forward to compute even if the likelihood of the observed data is not.

Complete Data Log Likelihood: log likelihood of parameters of interest , based on both the observed data and the missing data.

The general setup

Let $\mathcal{X}$-Complete data set which contains the missing data as well.
and $\mathcal{Y}$-Incomplete data set which contains only the observed data
$f : \mathcal{X} \to \mathcal{Y}$ is a many to one map.

Let $g_c(x, \Theta)$ denote the pdf of the random vector X corresponding to the complete data vector x.
Θ_0=initial values of the parameters.

1. E step: Compute the complete data log likelihood by substituting the functions of missing data by the conditional expectations of the same given the observations and the current value of Θ.

$$log(\widehat{g_c(x;\Theta)}) = E_{\Theta_0}(log(g_c(x, \Theta|y)))$$

2. M step: Maximize $log(\widehat{g_c(x;\Theta)})$, w.r.t Θ.

3. This is repeated until $log(\widehat{g_c(x;\Theta_{k+1})}) - log(\widehat{g_c(x;\Theta_k)})$ becomes arbitrarily small. DLR (Dempster, Laird and Rubin) shows that $log(\widehat{g_c(x;\Theta_{k+1})})$ is not decreased after an EM iteration, i.e.,

$$log(\widehat{g_c(x;\Theta_{k+1})}) \geq log(\widehat{g_c(x;\Theta_k)})$$

Baum-Welch ALgorithm(EM algorithm for HMM)

Since the sequence of states occupied by the Markov chain of an HMM is not observed, we treat them as missing data and try to implement the EM algorithm to find the maximum likelihood estimators of the parameters.

In the case of HMM, it is convenient to represent the states of the markov chain or the transitions by 0 or 1s as follows:

$$\begin{aligned} u_j(t) = 1 \quad &\text{iff} \quad c_t = j, \;\; t = 1, 2, \cdots, T \\ v_{jk}(t) = 1 \quad &\text{iff} \quad c_{t-1} = j \quad \text{and} \quad c_t = k, \;\; t = 1, 2, \cdots, T \end{aligned}$$

With this, the CDLL of the HMM is given by:

$$\begin{aligned}\log(P(X_1^T, C_1^T)) &= \log(\delta_{c_1} \prod_{t=2}^{T} \gamma_{c_{t-1},c_t} \prod_{t=1}^{T} p_{c_t}(x_t)) \\ &= \log(\delta_{c_1}) + \sum_{t=2}^{T} \log(\gamma_{c_{t-1},c_t}) + \sum_{t=1}^{T} \log(p_{c_t}(x_t)) \\ &= \sum_{j=1}^{m} u_j(1)\log(\delta_j) + \sum_{j=1}^{m}\sum_{k=1}^{m}\left(\sum_{t=2}^{T} v_{jk}(t)\log(\gamma_{jk})\right) + \sum_{j=1}^{m}\sum_{t=1}^{T} u_j(t)\log(p_j(x_t))\end{aligned}$$

1. E step: Replace the functions of missing data by their conditional expectation given the observation and the current parameter estimate.

   $$\widehat{u}_j(t) = E(c_t|X_1^T) = P(c_t|X_1^T) = \alpha(j)\beta(j)/L_T$$

   and

   $$\widehat{v}_{jk}(t) = E(c_{t-1} = j, c_t = k|X_1^T) = P(c_{t-1} = j, c_t = k|X_1^T) = \alpha_{t-1}(j)\gamma_{jk}p_k(x_t)\beta_t(k)/L_T$$

2. M step: Having replaced the $u_j(t)$ and $v_{jk}(t)$ by $\widehat{u}_j(t)$ and $\widehat{v}_{jk}(t)$, maximize the CDLL w.r.t the transition probabilities and intial distribution and the model parameters.

If we notice carefully, we will see the maximization step splits into 3 separate maximization problems. The first term w.r.t to the initial distribution, the second term w.r.t the transition probabilities and the third one w.r.t the model parameters. So, we should maximize:

1. $\sum\limits_{j=1}^{m} \widehat{u}_j(1)\log(\delta_j)$ w.r.t to $\boldsymbol{\delta}$.

   $$\begin{aligned}\nabla\left(\sum_{j=1}^{m} \widehat{u}_j(1)\log(\delta_j)\right) &= \lambda\nabla\left(\sum_{j=1}^{m}\delta_j\right) \\ \Rightarrow \widehat{u}_i(1)/\delta_i &= \lambda \quad i = 1, \cdots, m \quad \text{(Differentiating w.r.t } \delta_i\text{)} \qquad (3.12) \\ \Rightarrow \lambda_i &= \widehat{u}_i(1)/\lambda \\ \Rightarrow \sum_{i=1}^{m}\delta_i &= \sum_{i=1}^{m}\widehat{u}_i(1)/\lambda \\ \Rightarrow \lambda &= \sum_{i=1}^{m}\widehat{u}_i(1)/\lambda \\ \Rightarrow \delta_i &= \frac{\widehat{u}_i(1)}{\sum\limits_{i=1}^{m}\widehat{u}_i(1)} \quad \text{(From 3.12)}\end{aligned}$$

2. $\sum\limits_{j=1}^{m}\sum\limits_{j=1}^{m}\left(\sum\limits_{t=2}^{T}\widehat{v}_{jk}(t)\right)\log(\gamma_{jk})$ w.r.t Γ. Like in the first chapter section 1.3.2, proceed in the similar using the lagrange multiplier and we get,

   $$\gamma_{jk} = \frac{f_{jk}}{\sum\limits_{k=1}^{n} f_{jk}}, \quad \text{where} f_{jk} = \sum_{t=2}^{T}\widehat{v}_{jk}(t)$$

3. The maximization of the third term may be easy or difficult depending on the model assumed.

   It should be noted that the computation of forward and backward algorithm is prone to under- or overflow. So, proper scaling should be done to prevent it.

3.6 Model Validation

We can either use the training data or test data to perform model validation. Validation done on the training data usually talks about goodness of fit of the model i.e., whether the error between the prediction value and true value is random or not. Validation done on the test data usually analyzes the model's predictive performance when applied to new data. Usually we do both the analysis for a wholesome assessment of the proposed model in which we use the training set to find the parameters of the model and use the test set to analyse the errors due to prediction.

While we fit the model, we have a trade-off between the the number of parameters and the goodness of fit of the model. In very loose terms, increasing the number of parameters improves the fit of the model where as it increases the time and space complexity of the method. We use AIC or BIC to choose the model and then use pseudo residuals to verify the goodness of fit. The most common model selection methods are:

1. AIC(and related methods like C_p)
2. Cross-validation
3. BIC(MDL and Bayesian model selection)

We need to distinguish between 2 goals given a set of proposed models:

1. We assume any of the proposed models need not be the true model but we want to find the model which gives the best prediction
2. We assume one of the proposed models is the true model and try to find that ”true” model.

Generally, AIC and cross-validation are used for goal 1 while BIC is used for goal 2. In this section, we will focus on AIC and BIC.

3.7 AIC and BIC

3.7.1 AIC: Akaike Information Criterion

Suppose f is the true underlying model governing the process. We have $g_1, g_2 \cdots g_k$, the approximating models. Our aim is to choose the best model in some sense. In AIC, we define the discrepancy between the true and the approximating models using the Kullback Leibler divergence(KL-divergence) i.e., $K(f, \widehat{g}_1), K(f, \widehat{g}_2), \cdots, K(f, \widehat{g}_k)$

and we choose the model whose corresponding KL-divergence is the least. We have used $\hat{g}_i$ instead of g_i because we do not have the true parameters of the approximating model but only the estimates found out by the data.

Since these discrepancies depend on the true model 'f', it is not possible to compute them. Instead, either we make our decision depending on the estimators of the expected discrepancies $\widehat{E}_f(K(f,\widehat{g}_1)), \cdots, \widehat{E}_f(K(f,\widehat{g}_2))$ which are referred to as the model selection criteria or we can always measure how much more information is lost by $\widehat{g}_1$ than by $\widehat{g}_2$ using AIC while approximating. The later estimate is only valid asymptotically and when small amount of data is provided, some correction is necessary.

We define the KL-divergence between f and g as follows,

$$K(f,g) = \int f(x)\, \log(f(x)/g(x))\, dx$$

where the divergence basically represents the information loss when we try to approximate 'f' by 'g'.

$$\begin{aligned} K(f,\widehat{g}_j) &= \int f(x)\, \log(f(x)/\widehat{g}_j(x))\, dx \\ &= \int f(x)\, \log(f(x))\, dx - \int f(x)\, \log(\widehat{g}_j(x))\, dx \end{aligned}$$

The first term is independent of 'j'. So, minimizing the second term is equivalent to maximizing

$$K_j = \int f(x) \log(\widehat{g}_j(x,\widehat{\theta}_j))dx$$

So, a nice estimate for K_j is

$$\widehat{K}_j = \frac{1}{n}\sum_i^n \log(\widehat{g}_j(X_i;\widehat{\theta}_j)) = \frac{\ell_j(\widehat{\theta}_j)}{n}$$

where, $\ell_j(\hat{\theta}_j)$ is the log likelihood function for model j and $X_1, X_2, \cdots, X_n$ are data points drawn from the density f.

As we have used the same data twice, once to find out the mle and once to approximate the integral, hence the estimator turns out to be biased. Akaike showed that the bias is approximately d_j/n where d_j is the dimension of the parameter space of the model 'j'. Therefore we use,

$$\overline{K}_j = \frac{\ell_j(\widehat{\theta}_j)}{n} - \frac{d_j}{n} = \widehat{K}_j - \frac{d_j}{n}$$

Now, define

$$\text{AIC}(j) = 2n\overline{K}_j = 2\ell_j(\widehat{\theta}_j) - 2d_j$$

The preferred model is the one with maximum AIC value.
Note that, AIC does not say anything about the absolute goodness of fit. So, if all of the models fit poorly, it will not serve a warning for that. Hence, we should always verify the absolute quality of fitting.

When the sample size is small, there is a higher chance that AIC will choose the model with higher number of parameters(it will overfit), hence a correction term is added which is AICc. AICc is AIC with the correction term for small sample size which we are not going to deal here.

3.7.2 BIC: Bayesian Information Criterion

This approach is to select the model which is most likely to be true. We put priors $\pi_j(\theta_j)$ on the parameter θ_j. We also put priors p_j that the model 'j' is the true model. Then

$$P(g_j|X_1^T) \propto P(X_1^T|g_j)\, p_j$$

Moreover,

$$P(X_1^T|g_j) = \int P(X_1^T|g_j, \theta_j)\, \pi_j(\theta_j)\, d\theta_j = \int L(\theta_j)\, \pi_j(\theta_j)\, d\theta_j$$

We have to maximize $P(g_j|X_1^T)$. So, we choose 'j' to maximize

$$\log \int L(\theta_j)\, \pi_j(\theta_j)\, d\theta_j + \log(p_j)$$

Taylor series expansion of the first term shows that

$$\log \int P(X_1^T|g_j, \theta_j)\, \pi_j(\theta_j)\, d\theta_j + \log(p_j) \approx \ell_j(\hat{\theta}_j) - \frac{d_j}{2}\log(n) = \text{BIC}_j$$

It can be shown that the terms involving the prior are lower order than the term that appears in the formula for BIC$_j$, so they have been dropped. Note that, BIC assumes that one of the the proposed models is true and we assign prior probabilities to them as well and then find the one which is most likely to be true by examining the posterior corresponding probabilities.

3.8 Model Checking with Pseudo Residuals

Here comes the verification of the model fitting. We need to assess the general goodness of fit and identify the outliers relative to the model. In this section we define pseudo residuals which help us in model checking.

3.8.1 Continuous Case

We know that if X is a random variable with continuous distribution function F, then $F(X) \sim U(0, 1)$.

Uniform pseudo residual: The uniform pseudo residual of an observation x_t from a continuous random variable X_t is defined as

$$u_t = P(X_t \leq x_t) = F_{X_t}(x_t)$$

where the probability is found under the fitted model. If the model is correct, then, u_t is distributed $U(0, 1)$ and if the histogram or qq plot of the uniform pseudo residual are not as they should be, the model is not valid.

This approach has a problem when it comes to outlier identification. Here, if a value lies close to 0 or 1 on the uniform qq-plot, it is difficult to decide if it is likely or not since values like 0.99 and 0.96 lie very close. The problem can be solved by using normal pseudo residuals.

Normal Pseudo Residuals: The normal pseudo residual is defined as follows,

$$z_t = \Phi^{-1}(u_t)$$

If the fitted model is valid then the z_t follows standard normal. When the observation is the median, then value of the residual is zero. So, basically normal pseudo residuals measures deviation from median not mean. We can check the model either visually from the histogram or qq-plot of the residuals or by the tests of normality. The previous problem of outliers is solved by this approach. So, when the observation lies far from the median, the absolute value of the residual increases and outliers are readily observed on the normal scale as the domain is the whole of the real line $\mathbb{R}$.

3.8.2 Discrete Case

The theory of pseudo-residuals that is described in the previous section holds for continuous distributions only as Cumulative distribution Function F of a discrete random variable X does not follow $\sim U(0, 1)$.

In the case of discrete observations, we can modify it to allow for discreteness. Here, the pseudo residuals are defined as intervals. We define uniform pseudo-residual segments as,

$$[u_t^-; u_t^+] = [F_{X_t}(x_t^-); F_{X_t}(x_t)]$$

where x_t^- is the greatest realization possible that is strictly less than x_t. And we define the normal pseudo residual segments as

$$[z_t^-; z_t^+] = [\Phi^{-1}(u_t^-); \Phi^{-1}(u_t^+)]$$

This is only correct if the parameters of the fitted model are known, it still works well enough if the number of parameters is very small in comparison to the sample size. Since, we do not have qq plot for segments, we need to specify a representative element of the segments. So, we define "mid-pseudo residuals" for that purpose as

$$z_t^m = \Phi^{-1}\Big(\frac{u_t^- + u_t +}{2}\Big)$$

Now, having known about pseudo residuals in general. Let us see what purpose they serve in the context of HMMs:

- **Ordinary Pseudo Residuals:** Those residuals that are based on the conditional distribution given all other observations.
- **Forecast Pseudo Residuals:** Those residuals that are based on the conditional distribution given all the preceding observations.

Ordinary Pseudo Residuals:

This technique points out the observations which are sufficiently extreme to suggest that they differ in nature from the rest of the data. For continuous observations, we have

$$z_t = \Phi^{-1}(P(X_t \leq x_t|\mathbf{X}^{(-t)} = \mathbf{x}^{(-t)}))$$

For discrete observations we have,

$$z_t^- = \Phi^{-1}(P(X_t < x_t|\mathbf{X}^{(-t)} = \mathbf{x}^{(-t)}))$$
$$z_t^+ = \Phi^{-1}(P(X_t \leq x_t|\mathbf{X}^{(-t)} = \mathbf{x}^{(-t)}))$$

Forecast Pseudo Residuals:

These type of residuals measure the deviation of an observaton from the median of the corresponding one step ahead forecast. If the forecast pseudo residual is extreme, then we conclude that the observation is an outlier or the model no longer describes the sequence well enough.
For continuous observations, we have

$$z_t = \Phi^{-1}(P(X_t \leq x_t|\mathbf{X}^{(t-1)} = \mathbf{x}^{(t-1)}))$$

For discrete observations we have,

$$z_t^- = \Phi^{-1}(P(X_t < x_t|\mathbf{X}^{(t-1)} = \mathbf{x}^{(t-1)}))$$
$$z_t^+ = \Phi^{-1}(P(X_t \leq x_t|\mathbf{X}^{(t-1)} = \mathbf{x}^{(t-1)}))$$

Chapter 4

Types of HMM, covariates, random effects and pooling

One advantage of HMM is that the definition can easily be extended to cater to various needs. We can use discrete, continuous or the mixture of the two as state dependent distribution. We can also use different family of distributions for each state.

- We use HMMs for binary data to model daily rainfall occurrence(rain or no rain), daily trading share(traded or not traded).
- We use Binomial HMMs to model series of bounded counts.

  $$ {}_tp_i(x_t) = \binom{n_t}{x_t}\pi_i^{x_t}(1-\pi_i)^{n_t-x_t}, \qquad x_t = 0, 1, \cdots, n_t $$

  the prefix t suggests the dependence of number of successes on time. For forecasting purposes, to predict n_{T+h}, we can fit a separate model to it. Examples where the binomial HMM is used,

  - n_t = Number of mi smart band available on day 't', x_t = Number of mi bands purchased on day 't'.
  - n_t = Number of newspapers on day 't', x_t = Number purchased on day 't'.
- To model series of proportions, we use beta distribution.
- For continuous state dependent distribution, instead of probability mass function, we have probability density function.

4.1 Multinomial HMM

Instead of success and failure, if we have more than two classes of events, then we use multinomial HMM. Now we have 'q' time series data. Note that, in binomial-HMM, q=2.
$\mathbf{X}_t = (X_{t1}, X_{t2}, \cdots, X_{tq})\ \ \forall t \in \{1, 2, \cdots, T\}$

Similar to the univariate case, we assume $P(\mathbf{X}_1^T|\mathbf{C}_1^T) = \prod_{t=1}^{T} P(\mathbf{X}_t|C_1^T)$ i.e., conditional on the markov chain, the T random vectors $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_T$ are mutually independent. We have two constraints here,

- $\sum_{j=1}^{q} x_{tj} = n_t$
- $\sum_{j=1}^{q} \pi_{ij} = 1$

The likelihood function is,

$$L_T = \delta_1 \mathbf{P}(\mathbf{x_1})\mathbf{\Gamma}_2 \cdots \mathbf{\Gamma}_T \mathbf{P}(\mathbf{x_T})\mathbf{1}'$$

where, $P(\mathbf{x}_t) = {}_tP(\mathbf{x}_t) = \text{diag}({}_tp_1(\mathbf{x}_t), \cdots, {}_tp_m(\mathbf{x}_t))$

Using the assumption of conditional expectation, we have

$${}_tp_i(\mathbf{x_t}) = P(\mathbf{X_t} = \mathbf{x_t}|c_t = i) = \binom{n_t}{x_{t1}, x_{t2}, \cdots, x_{tq}} \pi_{i1}^{x_{t1}} \pi_{i2}^{x_{t2}} \cdots \pi_{iq}^{x_{tq}}$$

longitudinal conditional independence: Conditional on C_1^T, the random vectors $\mathbf{X_1}, \cdots \mathbf{X_T}$ are mutually independent.

Contemporaneous conditional independence:
$P(\mathbf{X_t} = x_t|c_t = i) = \prod_{t=1}^{T} P(X_t = x_t|c_t = i)$ i.e., conditional on the markov chain, the state dependent joint probability

4.2 Multivariate state dependent distribution

We have q time series $\{\mathbf{X}_t = (X_{t1}, X_{t2}, \cdots, X_{tq}) \;\; \forall t \in \{1, 2, \cdots, T\}\}$. We, of course assume longitudinal conditional independence.The likelihood function becomes

$$L_T = \delta_1 \mathbf{P}(\mathbf{x_1})\mathbf{\Gamma}_2 \cdots \mathbf{\Gamma}_T \mathbf{P}(\mathbf{x_T})\mathbf{1}'$$

If we assume contemporaneous conditional independence, then finding out the joint distribution becomes a lot easier.

4.2.1 Extensions to HMM

- We can let the observation depend on the previous observation in addition to the current state.(Markov switching models)
- We can make the markov chain of higher order, thereby letting the current observation depend on the last two states.
- We can allow for feedback by letting the previous observation affect the current state.

4.3 Longitudinal data

We have ’q’ times series data of the same type based on ’q’ subjects. We assume that the same type of model is used to describe the q series but the parameters might vary across subjects. The q component series in case of longitudinal data can be let govern by the same sequence of states or different sequence of states depending on the subjects.
Examples of longitudinal data:

- Disease status of q individual
- Rainfall at q places
- Number of products sold at q stores of a company

Note that, the series of longitudinal data govern by the same sequence of states can be handled using multivariate HMM. While estimating the parameters, we can always reduce the number of parameters to circumvent overparametrization by applying any of the three methods described below.

4.3.1 Covariates

We will discuss how covariates are incorporated while modelling time series. To forecast future values, we need the relevant future values of the covariate as well. So we fit another time series to forecast the values of the covariates.

- Trends and seasonal components modelled as parametric functions of time can be handled as covariates.
- We can assume the transition probabilities, rather are functions of time or other covariates. It is usually difficult to implement when the number of states is more than two since we have to make sure that for all value of the covariate, the sum of the probabilities is 1.
- We can assume that the parameters in the state dependent distributions are dependent on some covariates which affect the distribution. Suppose we are going to model the precipitation in a place, we might wanna let the parameters depend on some covariates derived form the atmospheric conditions of the place.

4.3.2 Random effects

Sometimes we assume that either the transition probabilities or the state dependent parameters come independently from a distribution itself. So, now instead of estimating a long list of parameters, we just estimate the parameters of the newly assumed distribution. Models with random effects, are called mixed-HMM. We can have continuous valued or discrete valued random effects.

Continuous valued random effects

When we use continuous valued random effects, often we use the normal distribution. To understand this better, let us take the example of partial pooling where we have pooled the transition probabilities. So we have to estimate 2q+2 parameters. While q might be a very large number, we introduce continuous random effects. So, we assume $\lambda_1^{(j)}$ follows Gamma(μ_1, σ_1) and $\lambda_2^{(j)}$ follows Gamma(μ_2, σ_2) for all the subjects. So, instead we have to estimate $\gamma_{12}, \gamma_{21}, \mu_1, \sigma_1, \mu_2, \sigma_2$ i.e., 6 parameters in total.

The likelihood here is given by

$$L_T = \prod_{j=1}^{q} \int_0^\infty \int_0^\infty \boldsymbol{\delta}\mathbf{P}(\mathbf{x_{1j}})\boldsymbol{\Gamma}\mathbf{P}(\mathbf{x_{2j}})\boldsymbol{\Gamma}\cdots\boldsymbol{\Gamma}\mathbf{P}(\mathbf{x_{Tj}})\mathbf{1}'\times f_1(\lambda_1;\mu_1,\sigma_1)f_1(\lambda_2;\mu_2,\sigma_2)\,d\lambda_1 d\lambda_2$$

The number of integrals is equal to the number of random effects we have introduced. Since we have to evaluate the integrals for each subject, barring special cases, this task is quite computationally heavy, where in we bring the concept of discrete random effects. Instead of specifying independent distributions, we can also introduce joint distribution of the parameters so as to allow for dependence among the parameters.

Discrete random effects

Again in case of partial pooling, we let the state dependent distribution to vary only and hence we have,

$$\lambda_i^{(k)} = \begin{cases} \lambda_{1,i} \text{with probability} \pi_{1,i}, \\ \lambda_{2,i} \text{with probability} \pi_{2,i}, \\ \vdots \\ \lambda_{q_i,i} \text{with probability} \pi_{q_i,i} \end{cases}$$

So, now instead of estimating the 2q+2 parameters, we have to estimate 2+2(q1+q2) parameters. THe likelihood in this case is,

$$L_T = \prod_{j=1}^{q} \sum_{j_1=1}^{q_1} \sum_{j_2=1}^{q_2} \boldsymbol{\delta}\mathbf{P}(\mathbf{x_{1j}})\boldsymbol{\Gamma}\mathbf{P}(\mathbf{x_{2j}})\boldsymbol{\Gamma}\cdots\boldsymbol{\Gamma}\mathbf{P}(\mathbf{x_{Tj}})\mathbf{1}'\pi_{j_1}\pi_{j_2}$$

4.3.3 Pooling

To make the ideas clear we will use the following scenario, Let $\{x_{tj} : t = 1, \cdots T; j = 1, \cdots, q\}$ and we wanna fit a stationary two state Poisson HMM. So, the parameters we will estimate are $\lambda_1^{(j)}, \lambda_2^{(j)}, \gamma_{12}^{(j)}, \gamma_{21}^{(j)}$. When no parameter is assumed to be same for the subjects is referred to as ”no pooling” and when all of the parameters are assumed to be the same for all the subjects is called ”complete pooling”. These are the extreme cases and partial pooling is when we let some of the parameters to be same across the subjects and let the others to vary.

Note that, complete pooling can be misleading when there is substantial difference present among the subjects. The parameters estimated using no pooling can guide us toward a more suitable alternative, for example if we have certain parameters which are nearly constant for the subjects, we can use partial pooling in this case.

- Complete pooling: Since we have $\lambda_1^{(j)}, \lambda_2^{(j)}, \gamma_{12}^{(j)}, \gamma_{21}^{(j)}$ to be the same for all the q subjects, we just have to estimate 4 parameters in total.
- No pooling: No parameter is assumed to be same for all the subjects and hence we have to estimate all the 4q parameters.
- Partial Pooling: If we assume the transition probabilities are same across the subjects, then we have to estimate 2q+2 parameters and if we assume the mean is same for the subjects, then we have to estimate 2+2q parameters.

The likelihood in case of longitudinal data is(assuming independence of the component series)

$$L_T = \prod_{j=1}^{q} \boldsymbol{\delta}^{(j)} \mathbf{P^{(j)}}(\mathbf{x_{1j}}) \mathbf{\Gamma^{(j)}} \cdots \mathbf{\Gamma^{(j)}} \mathbf{P}(\mathbf{x_{Tj}}) \mathbf{1}'$$

where $\mathbf{P^j}(\mathbf{x})$ is the transition probability matrix for the j^{th} component series and $\boldsymbol{\delta}^{(j)}$ is the stationary distribution of the j^{th} component series.

Chapter 5

Gene finding in prokaryotic cells

5.1 Introduction

Gene sequence is nothing but a string of nucleotides A, T, G, C in case of dna and A,U,G,C in case of rna which contains the information on how organisms function. To understand an organism at the molecular level, knowing the genome of the sequence is very crucial. The long strand of gene sequence is useless unless the functional subsequences are known. Traditionally genes were found and validated with wet lab experiments, which is the most reliable but it is time consuming, labor intensive and costly.
With the advent of powerful sequencing technologies, more and more organisms are being sequenced, hence pushing the need for an automated system to find genes.

5.2 Proteins

Prokaryotic cells must make appropriate phenotypic response to adapt to the constant changing surrounding they grow, in order to survive. Similarly, coordination is required among eukaryotic cells of tissues, organ and body of more complex organisms.

Cells produce the appropriate molecular machinery to perform these complex tasks. Proteins are the main components of this machinery and are involved in tasks as diverse as the formation of cellular structures, cellular communication or the regulation of gene expression. Proteins are molecular chains consisting of basic blocks or monomers called amino acids. There are 20 different amino acids which exhibit specific biochemical characteristics.

The necessary information to produce amino acids in the correct order to synthesize proteins is found in DNA fragments of known genes. However, the molecular machinery that synthesizes proteins, ribosomes, does not use the information in the DNA directly. On the contrary, information is copied from DNA during transcription into RNA preserving the same nucleotide in the same order with the exception of substituting thymine(T) for uracil(U) nucleotide. Finally, ribosomes synthesize RNA using information from the protein during the process called translation. During this process the ribosomes use the genetic code to determine the amino acid sequence to be assembled.

5.2.1 Transmembrane protein

A transmembrane protein (TP) is a type of integral membrane protein that spans the entirety of the cell membrane. Many transmembrane proteins function as gateways to permit the transport of specific substances across the membrane. Certain hydrophobic regions organize themselves inside the bilayer as transmembrane -helices while more hydrophilic regions are in contact with the aqueous intracellular and extracellular environments. Interaction energies are very high between hydrophobic regions of the protein and hydrophobic regions of the lipid bilayer, as well as between hydrophilic regions of the protein and the extracellular and intracellular environments. These interactions strongly stabilize transmembrane proteins within the bilayer, thus preventing their extracellular and cytoplasmic regions from flipping back and forth.

5.3 Open Reading Frames

The genetic code uses three nucleotides, a codon, to encode an amino acid. The code is redundant in the sense that several different codons encode the same amino acid. There are special codons that mark the beginning and end of the polypeptide chain. There is only one initiation codon, ATG, encoding the amino acid methionine and three stop codons TAA, TAG and TGA that do not encode any amino acid.

Definition 1 *Open Redaing Frames: A DNA fragment starting at the ATG start codon followed by a series of codons until reaching a stop codon TAA, TAG or TGA is called open reading frame (ORF).*

Given a sequence over the alphabets (a, c, g, t), a subsequence t of the dna is an open reading frame (ORF), if:

- length of t is a multiple of three, i.e., t consists of a series of codons.
- t begins with the ATG start codon.
- t ends by one of the stop codons TAA, TAG or TGA.

Every coding Deoxyribonucleic Acid (DNA) has six possible reading frames: three on the direct strand and three in the reverse complementary strand. The nucleotides on the direct strand are grouped into triplets starting from the first, second or third. Therefore, there are three possible reading frames in the direct strand. Similarly, there are three possible reading frames in the reverse complement.

Out of all the reading frames, typically, only one reading frame, the ORF, is used to translate the gene. Hence, the prokaryotic gene finder should objectively and primarily be able to identify which of the six possible reading frames contains the gene i.e. is an

ORF. In general, bacterial genes have long ORFs. This is a hint for gene finding. For example, if the first reading frame has the longest sequence without a stop codon, then its amino acid sequence most probably leads to the gene product.

5.4 Challenges

Uptill now, as we have discussed, just finding ORFs is not an assuring indication of functional genes.

- Not every ORF is a coding region. Telling the difference between genes and random ORFs is the most important goal of the gene finding process. Bacterial coding regions tend to be longer but even if we tune a certain length threshold and define that ORFs longer than that threshold are genes, differentiating between short genes and random ORFs remains a problem.
- Identifying the right ORFs is deteriorated when two ORFs overlap. Although this is considered to happen rarely in prokaryotes, it is difficult to automatically resolve the problem.
- Moreover, there are multiple start codons. In most cases, ATG is the start codon that suggests initiation of translation. Occasionally, GTG and TTG act as initiation sites. Multiple start codons can cause ambiguities, because their presence does not ensure translation initiation.
- Longer ORFs tend to be coding regions can be misleading for gene prediction in GC-rich organisms. Because the stop triplets (TAA, TGA, TAG) are AT rich, their frequency is lower in organisms with high GC content. Hence, the probability that long ORFs occur by chance increases proportionally to the GC content.

Therefore, recognising genes based on ORFs relies on additional biological factors such as signal sensors (start and stop codons, promoters, etc.), but they also use content sensors, such as patterns of codon usage or other statistically inferred features. To exploit the correlations between successive nucleotides and the structure of codons, k-order inhomogenous Markov chains are used as they have shown to be appropriate in inferring the statistical description of the gene structure where value of k is 2,5 or 8 as the codon size is three.

5.5 Gene Finding in Acaryochloris marina

For my project I have tried to implement Hidden markov models to predict the hydrophobic and hydrophilic parts of the transmembrane proteins of the prokaryote Acaryochloris marina.

Searching for genes in prokaryotes is considerably easier than in eukaryotes due to several reasons. There is a high density of genes in prokaryotic genomes, commonly more than 90% of the genome is coding. Prokaryotic genes consist of an open reading frame lacking the structure of eukaryotic genes that have introns and exons. Therefore, the problem of finding genes in prokaryotes is limited to two steps:

1. Identify all open reading frames.
2. According to a level of significance decide about which of them are genes.

5.5.1 Hypothesis testing

DNA sequences that encode proteins are not a random combination of codons but coding regions are often interspersed with non-coding regions. Hence there is a chance, an irrelevant ORF being tagged as coding region while a true coding region is being ignored. So, when all ORF have been determined in a genome, we must decide which of them have a greater confidence that represent true genes.

Length of an ORF is the classical criteria for this decision because the probability of an ORF being random is more if the length is less.

We propose a null and an alternate hypothesis. Given the data, we fabricate a statistic which is then used to contrast the null hypothesis against the alternate. So, we use hypothesis testing to find out the threshold length below which we discard the ORF.

For the present case, we have: **Null hypothesis, H_0:** The tested ORF is random.
Alternate hypothesis, H_A: The tested ORF is not random and hence there is evidence to accept it as a true gene.

There are two different types of error that can occur in a hypothesis testing:

- **type I error** or **false positives** is to reject the null hypothesis when it is indeed true.
- **type II error** or **false negatives** is not to reject the null hypothesis when it is actually false.

Usually the probability of committing a type I error(which is considered to be more serious than type II error) is denoted by alpha and is called the significance level which is usually set at 0.05 or 0.01 to control this error type. Here we use the length of the non-stop codons as the plausible statistic.

Finally, to decide whether or not to reject the null hypothesis is usually calculated by finding the p-value (probability value) associated with the used statistic. The p-value is the probability of obtaining a value of the statistic due to mere randomness i.e., just assuming that the null hypothesis is true. Commonly, a decision to reject the null hypothesis is made when the p-value is less than the significance level.

In the case at hand, we can obtain the p-value associated with the statistical length of ORF, calculating the probability of finding an ORF of length l codons due to mere randomness. Given an ORF, we known that the first codon is ATG (start codon) and the last is a stop codon, TAA, TAG or TGA. Therefore, we must determine the p-value corresponding to length $l - 2$.

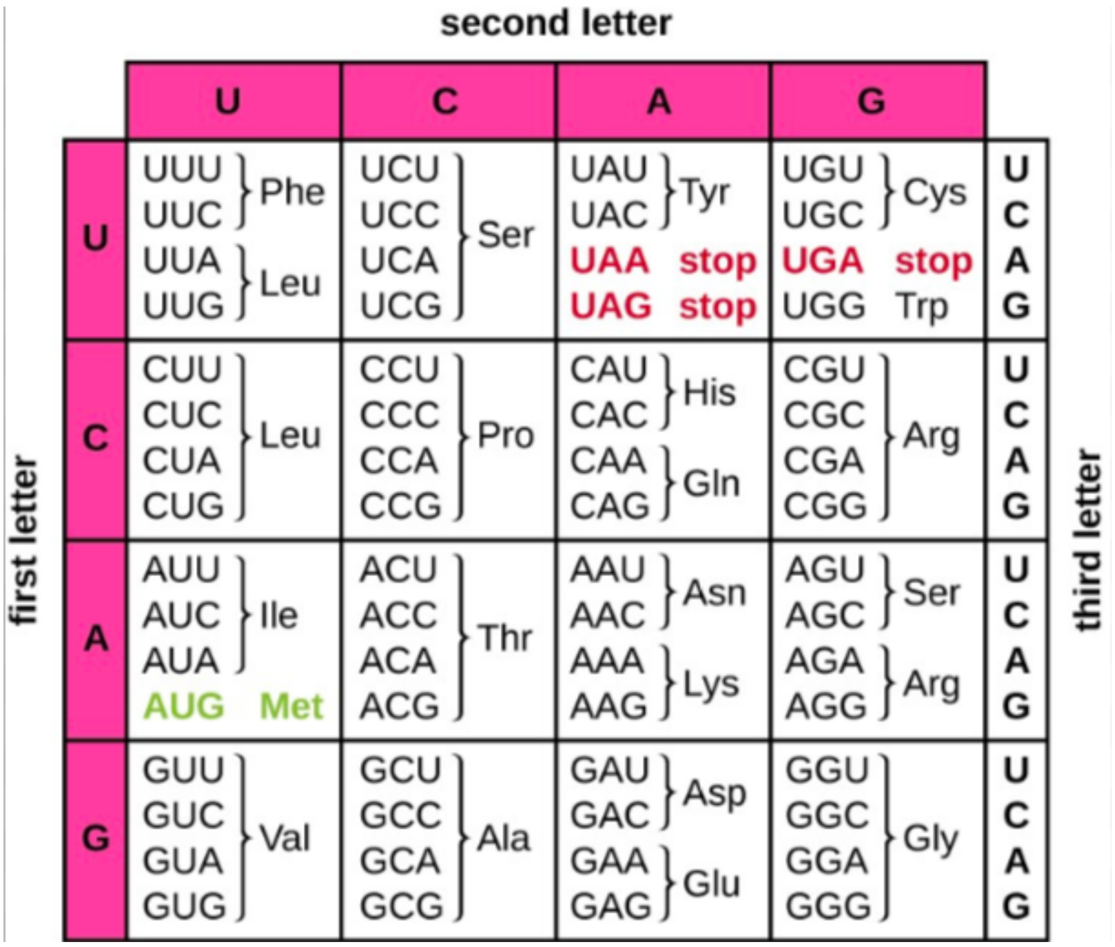
Assumption: The codons appear completely random.
Total number of codons possible $= 64$

$$P(\texttt{stop codon}) = \frac{3}{64}$$
$$P(\texttt{non-stop codon}) = 1 - \frac{3}{64}$$
$$P(l-2\ \texttt{non-stop codons}) = (1 - \frac{3}{64})^{(l-2)}$$

The initial estimate usually contains a lot of false positives but after applying the hypothesis testing, we could get rid some of those.

There are several overlapping ORFs i.e., ORF ending in the same stop codon but starting at different start codons. All these ORF differ only in some codons. So it is not expected that all these genes constitute independent ORF. Therefore, the most significant ORF will be taken from all these overlapping ORFs. Since the statistic used in hypothesis testing is the length of the corresponding sequence, the most significant ORF will be the longest.

Codon to protein conversion table:



first letter	second letter: U	C	A	G	third letter
U	UUU Phe	UCU Ser	UAU Tyr	UGU Cys	U
	UUC	UCC	UAC	UGC	C
	UUA Leu	UCA	UAA stop	UGA stop	A
	UUG	UCG	UAG stop	UGG Trp	G
C	CUU Leu	CCU Pro	CAU His	CGU Arg	U
	CUC	CCC	CAC	CGC	C
	CUA	CCA	CAA Gln	CGA	A
	CUG	CCG	CAG	CGG	G
A	AUU Ile	ACU Thr	AAU Asn	AGU Ser	U
	AUC	ACC	AAC	AGC	C
	AUA	ACA	AAA Lys	AGA Arg	A
	AUG Met	ACG	AAG	AGG	G
G	GUU Val	GCU Ala	GAU Asp	GGU Gly	U
	GUC	GCC	GAC	GGC	C
	GUA	GCA	GAA Glu	GGA	A
	GUG	GCG	GAG	GGG	G

5.6 Algorithm

I used python to code.

Algorithm of finding relevant proteins:

1. Take the dna sequence and store all the positions of the start codon in one array and all the positions of the stop codon in another array. This is implemented in the user defined function **get_special_codon_pos**.

2. Run through the stop codons and for each stop codon, find all the start codons that appear before it. Then Store the index of both the start and stop codons in a 2-dimensional array. This array gives index of all the possible ORFs present in the dna. This is implemented in **get_special_codon_mat**.

3. So, we got the ORFs of the first reading frame of the dna. Next we apply step 1 and 2 to the second and third reading frames. The function **get_strand_orfs** does this work.

4. We find out the threshold length of the ORFs using the p-value and keep the ORFs whose length is greater than the threshold length. Then out of all the overlapping ORFs, we keep the longest, which is implemented in **get_significant_orfs**

5. We repeat step 1, 2, 3 and 4 for the reverse complement of the dna.

So, once we find out all the proteins present in the bacteria. Next we try to see which ones are the transmembrane protein. For this we try to implement HMM(discrete), theory of which we have covered in the previous chapters.

Algorithm for implementing HMM:

I used the dna data of Acryochoris Marina from the NCBI website and the 'hydrophobic and hydrophilic labeled' protein sequences of the same prokaryote from the uniprot website.

The observation states are the 20 amino acids (A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V). The hidden states are the 'hydrophobic' and 'hydrophilic' parts abbreviated as 'b' and 'l' respectively.

1. I took some protein sequences from the above website whose hydrophobic and hydrophilic sections are labelled. I concatenated three sequence to form a single training sequence.

2. The function **get_supervised_estimate** takes the training set as the input. Using the first part of the baum welch algorithm, it calculates the initial probability, transition matrix and the emission matrix.

3. using these estimates, we try to estimate the hydrophobic and hydrophiic part of the test protein sequence, which is implemented in the **log_viterbi** function. Instead of using the viterbi algorithm straight, we convert the probabilities into log probability to deal with the underflow cause by multiplying long strings of probabilities.

Chapter 6

Sequence Alignment & HMM

6.1 Introduction

We can gather information about one biological sequence from another given they are similar to some extent. In this we are going to consider protein sequences. To decide upon their similarity, the notion of alignment is quite important. Sequences throughout history have been accumulating deletions, insertions and substitutions. So, before making any conclusive assertion on the similarity of two sequences, we look for possible alignments. For that we need an effective scoring mechanism which can gauge the similarity between two sequences based on the optimal alignment i.e., biologically most likely alignment. So, we want a scoring system which takes into consideration the evolutionary history of the biological molecules, their three dimensional structures and other characteristics which define which substitutions, deletions or insertions are more likely to happen.

To assert if two sequences are related, we have to assess the alignments for which the main issues that arise sequentially are:

1. Type of alignments to be used
2. Scoring system to rank the alignments
3. algorithm to find the best scoring alignment(s)
4. statistical method to evaluate the significance of the alignments

Below we will see two situations which will be sufficient for us to see why do we need a good scoring model.

```
HBA_HUMAN   GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
            ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU  NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

Figure 6.1: This particular alignment is meaningful as the two sequences are evolutionarily related, have the same three dimensional structure and same function in oxygen binding.

```
HBA_HUMAN   GSAQVKGHGKKVADALTNAVAHVDDMPNALSALSD----LHAHKL
            GS+ + G +   +D L  ++ H+ D+  A +AL D    ++AH+
F11G11.2    GSGYLVGDSLTFVDLL--VAQHTADLLAANAALLDEFPQFKAHQE
```

Figure 6.2: However this is an unwanted alignment in that the sequences have different three dimensional structures and functions.

In the central line, '+' represents a conservative/similar amino acid which contributes positive score whereas '-' represents non-conservative change which contributes negative score.

From the above example we can see that the major task in pairwise alignment is to come up with a scoring system which is successful in distinguishing cases like these.

6.2 Guide to a good score model

When we look at sequences for alignment; substitutions, insertions and deletions are the three mechanisms by which they could be different from each other. Natural selection screens the mutations and hence we see some pattern of change more than others. [Note: Insertions and deletions are termed as gaps.]

We assign score to an aligned pair of residues and to gaps and at the end we add all the score terms to find the total. In the probabilistic paradigm, this corresponds to the logarithm of the relative likelihood of the sequences being related, compared to being unrelated.
Informally, we will see later that, the scoring system takes into account that identities and conservative substitutions are more likely to be in alignment than not and hence these constitute positive score terms whereas non-conservative changes are less likely to be aligned and hence contribute negative score terms.

The additive scoring scheme implies the mutations at various sites are assumed to be independent of one another which works quite well for dnas and proteins.

6.2.1 Substitution matrix

$x = x_1, x_2, \cdots, x_n$
$y = y_1, y_2, \cdots, y_n$

where x and y are two sequences and x_i and y_j are the constituent blocks that can be bases in case of dna or amino acids in case of proteins. Since this is an ungapped alignment, the length of both x and y are same i.e., n.

$$\text{relative likelihood} = \frac{\text{Probability that the sequences are related}}{\text{Probability that the sequences are random}}$$

Under the random model, we have

$$P(x, y|R) = \prod_i q_{x_i} \prod_j q_{y_j}$$

where q_a is the probability of occurrence of a.
Under the Match model,

$$P(x, y|M) = \prod_i p_{x_i y_i}$$

where q_{ab} is the probability that residues a and b are derived from the same ancestor at some point.

$$\text{odds ratio} = \frac{P(x, y|M)}{P(x, y|R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} q_{y_i}}$$

In order to arrive at an additive scoring system, we take the log and finally we have

$$\text{log odds ratio} = S = \sum_i s(x_i, y_j)$$

where $s(x_i, y_j) = log\left(\frac{p_{x_i y_j}}{q_{x_i} q_{y_i}}\right)$
BLOSUM50 and PAM are two substitution matrices which are derived in this way and BLOSUM50 is shown in Figure 6.3.

6.2.2 Gap penalties

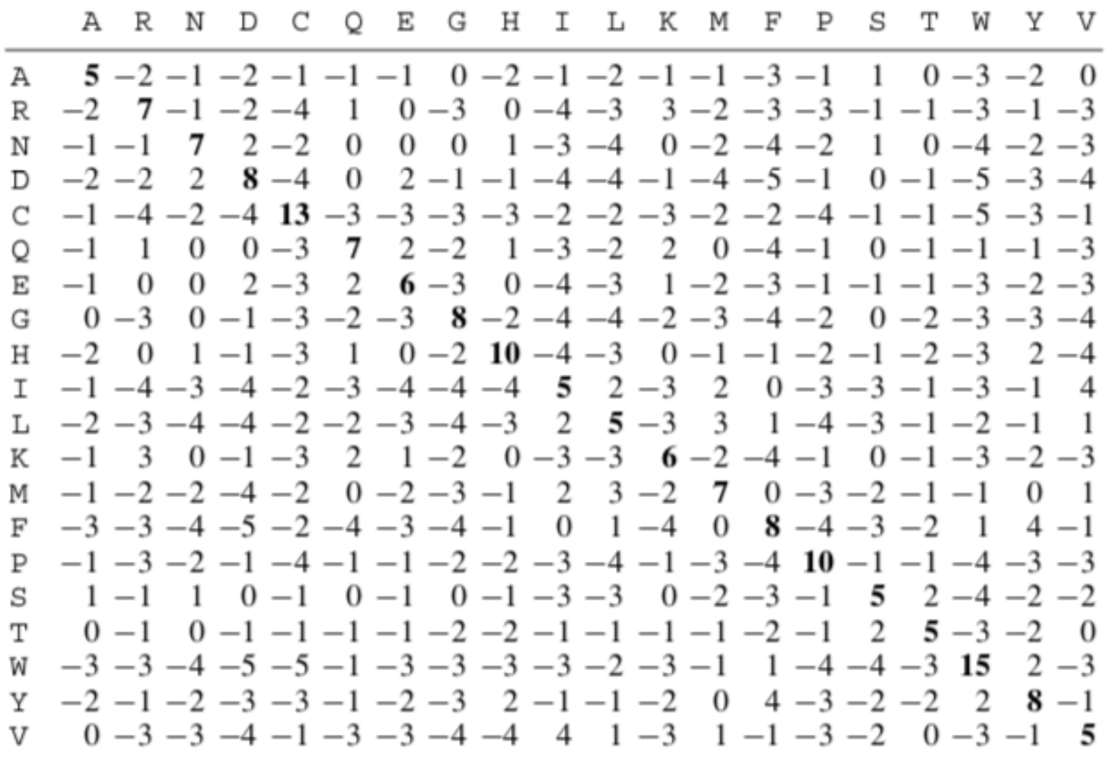
We are expected to penalise the gaps. So, the cost corresponding to 'g' number of gaps can be linear,

$$\gamma(g) = -gd$$

or can be affine

$$\gamma(g) = -d - (g - 1)e$$

d=gap open penalty
e=gap extension penalty.



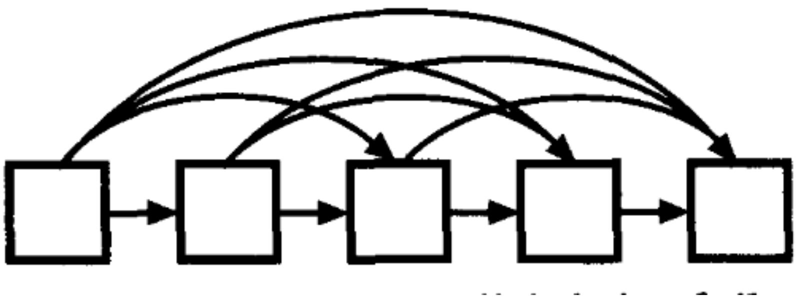
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	**5**	−2	−1	−2	−1	−1	−1	0	−2	−1	−2	−1	−1	−3	−1	1	0	−3	−2	0
R	−2	**7**	−1	−2	−4	1	0	−3	0	−4	−3	3	−2	−3	−3	−1	−1	−3	−1	−3
N	−1	−1	**7**	2	−2	0	0	0	1	−3	−4	0	−2	−4	−2	1	0	−4	−2	−3
D	−2	−2	2	**8**	−4	0	2	−1	−1	−4	−4	−1	−4	−5	−1	0	−1	−5	−3	−4
C	−1	−4	−2	−4	**13**	−3	−3	−3	−3	−2	−2	−3	−2	−2	−4	−1	−1	−5	−3	−1
Q	−1	1	0	0	−3	**7**	2	−2	1	−3	−2	2	0	−4	−1	0	−1	−1	−1	−3
E	−1	0	0	2	−3	2	**6**	−3	0	−4	−3	1	−2	−3	−1	−1	−1	−3	−2	−3
G	0	−3	0	−1	−3	−2	−3	**8**	−2	−4	−4	−2	−3	−4	−2	0	−2	−3	−3	−4
H	−2	0	1	−1	−3	1	0	−2	**10**	−4	−3	0	−1	−1	−2	−1	−2	−3	2	−4
I	−1	−4	−3	−4	−2	−3	−4	−4	−4	**5**	2	−3	2	0	−3	−3	−1	−3	−1	4
L	−2	−3	−4	−4	−2	−2	−3	−4	−3	2	**5**	−3	3	1	−4	−3	−1	−2	−1	1
K	−1	3	0	−1	−3	2	1	−2	0	−3	−3	**6**	−2	−4	−1	0	−1	−3	−2	−3
M	−1	−2	−2	−4	−2	0	−2	−3	−1	2	3	−2	**7**	0	−3	−2	−1	−1	0	1
F	−3	−3	−4	−5	−2	−4	−3	−4	−1	0	1	−4	0	**8**	−4	−3	−2	1	4	−1
P	−1	−3	−2	−1	−4	−1	−1	−2	−2	−3	−4	−1	−3	−4	**10**	−1	−1	−4	−3	−3
S	1	−1	1	0	−1	0	−1	0	−1	−3	−3	0	−2	−3	−1	**5**	2	−4	−2	−2
T	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	2	**5**	−3	−2	0
W	−3	−3	−4	−5	−5	−1	−3	−3	−3	−3	−2	−3	−1	1	−4	−4	−3	**15**	2	−3
Y	−2	−1	−2	−3	−3	−1	−2	−3	2	−1	−1	−2	0	4	−3	−2	−2	2	**8**	−1
V	0	−3	−3	−4	−1	−3	−3	−4	−4	4	1	−3	1	−1	−3	−2	0	−3	−1	**5**

Figure 6.3: **BLOSUM50 substitution matrix. The entries are scaled and rounded for computational ease.**

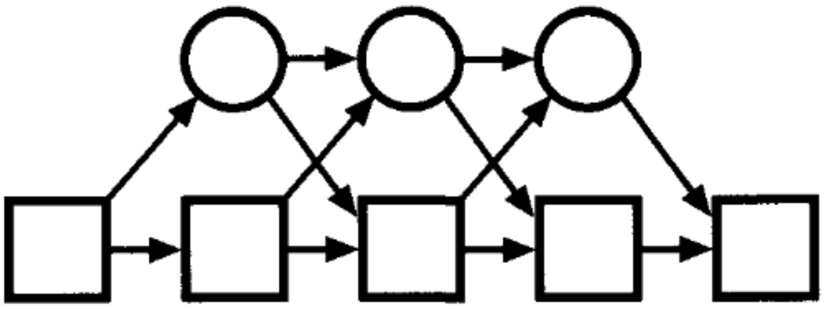
6.3 Silent States

Silent states or null states in an HMM are the ones which do not emit any symbols. Its usefulness is exaggerated when we need to connect each state in a chain to all the states that come after it. For dna sequence analysis in MSA, the length of such a chain could easily be 200 or more in which case we need around 20000 transition probability parameters. To estimate these many variables is an insuperable task using real life datasets. By using silent states, such a huge number could easily be reduced to around 600.

The situation is illustrated as follows:We need to make the staes completely forward connect as shown below



On the other hand we can connect the above chain in parallel to a chain of silent states. The introduction of silent states is not all gold and glitter, but there is a price we have



to pay. For instance, we can not have a model where the probability of transition from state 1 to state 4 is low but from state 1 to state 5 is high or a model where 1 to 5 is low but 1 to 4 is high.

6.4 Global Pairwise Sequence Alignment: Needleman Wunsch algorithm

This specific algorithm deals with getting the optimal global alignment of the pair of sequences, allowing gaps, using the previous solutions to optimal alignment of smaller subsequences. We are going to use linear score system to penalise gaps. We construct a matrix F in which $F(i, j)$ is the score of the best alignment of the sequence x and y up to the i^{th} and j^{th} position respectively.
To calculate $F(i, j)$, we can have three cases as follows:

- x_i is aligned with y_j in which case the score $F(i, j)$ becomes $F(i-1, j-1) + s(x_i, y_j)$
- x_i is aligned with a gap in which case the score $F(i, j)$ becomes $F(i-1, j) - d$
- y_j is aligned with a gap in which case the score $F(i, j)$ becomes $F(i, j-1) - d$

To help visualise the three cases, the following figure might help.

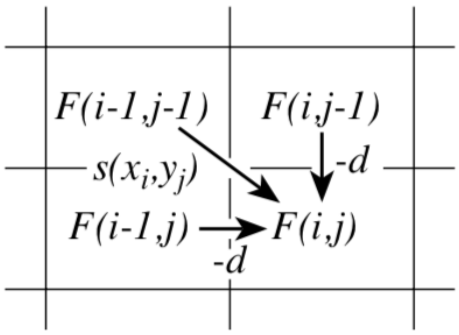
I G A x_i	A I G A x_i	G A x_i $-$ $-$
L G V y_j	G V y_j $-$ $-$	S L G V y_j

Figure 6.4: **x_i aligned to y_j, x_i aligned to gap, y_j aligned to gap**

The way we fill $F(i, j)$ is as follows:

$$F(0,0) = 0$$

$$F(i,j) = max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) - d \\ F(i, j-1) - d \end{cases}$$



The above procedure can be summarised in the picture below:
As we proceed to fill the values $F(i,j)$, we also store the cell from which the score $F(i,j)$ was derived which will be used to traceback and construct the alignment. We must notice that we don't have the values $F(i-1,j-1)$, $F(i,j-1)$ for the first row and the values $F(i-1,j-1)$, $F(i-1,j)$ for the first column. So, we define $F(i,0) = -id$ which means alignment of the sequence x till i^{th} position to all the gaps. Similarly define, $F(0,j) = -jd$.

The value in the final cell $F(m,n)$ by definition is the best score for an alignment of the two sequences x and y. We traverse back from $F(m,n)$ and write the alignment as follows:

- If we traverse from $F(i,j)$ to $F(i-1,j-1)$, then we align x_i with y_j
- If we traverse from $F(i,j)$ to $F(i,j-1)$, then we align y_j with a gap
- If we traverse from $F(i,j)$ to $F(i-1,j)$, then we align x_i with a gap

and we do this till we reach $F(0,0)$. At any point if we find two derivations to be equal, we can make a choice to keep both or arbitrarily choose one according to the need.

The alignment works because the scoring system is addition of independent scores. So, the best score up to some point is the best score up to the previous point, plus the incremental change.

Complexity of the algorithm We have to store the values in a $(m+1)*(n+1)$ matrix and roughly we have to do 3 calculation to fill each cell. In real life applications, m and n are comparable. So, we can conclude the time complexity of the above algorithm is $\mathcal{O}(n^2)$.

6.5 Local Pairwise Sequence Alignment: Smith-Waterman Algorithm

The problem with global alignment is that, while looking at a pair of highly diverged sequence, the global alignment fails to say they are similar even though they have shared evolutionary origin along their entire length. Local alignment comes to rescue as it can capture sensitive information which are present locally(as only part of such sequences are under strong natural selection to be preserved for a long time) otherwise

very difficult to be found by global alignment and as more proteins were sequenced, it was found that there was less overall similarities between two sequences of the same family but they share functional domains which are small in length. The sequence fragments might be conservative for the proteins belonging to the same family but it is impossible to find them by global alignment techniques.
So, most often than not we use local alignment algorithms.

We tweak the global alignment algorithm a bit to obtain the local alignment as follows:

$$F(0,0) = 0$$

$$F(i,j) = max \begin{cases} 0 \\ F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases}$$

The option 0 basically corresponds to starting a new alignment, the reason being, if the best score starts to become negative at some point, it is better to start a new alignment. Since we have penalties(negative scores) for gaps and the modified algorithm has an option zero, a consequence of it is that the first row and the first column in this matrix will now be filed with 0's. Note that the optimal local alignment might not be a subset of the optimal global alignment.

6.6 Pairwise Alignment using HMM

We are going to use the affine gap penalty system now. Let us specify the notations for the same.
M- Match state(the residues need not be identical)
X- Insert at sequence X
Y- Insert at sequence Y

$M(i,j)$- Score of best alignment between $x_{1,\cdots,i}$ & $y_{1,\cdots,j}$ given x_i is aligned with y_j
$X(i,j)$- Score of best alignment between $x_{1,\cdots,i}$ & $y_{1,\cdots,j}$ given x_i is aligned with a gap
$Y(i,j)$- Score of best alignment between $x_{1,\cdots,i}$ & $y_{1,\cdots,j}$ given y_j is aligned with a gap

So, the global alignment algorithm is modified as follows:

$$M(i,j) = max \begin{cases} M(i-1,j-1) + s(x_i, y_j) \\ X(i-1,j-1) + s(x_i, y_j) \\ Y(i-1,j-1) + s(x_i, y_j) \end{cases}$$

$$X(i,j) = max \begin{cases} M(i-1,j) - d \\ X(i-1,j) - e \end{cases}$$

$$Y(i,j) = max \begin{cases} M(i,j-1) - d \\ X(i,j-1) - e \end{cases}$$

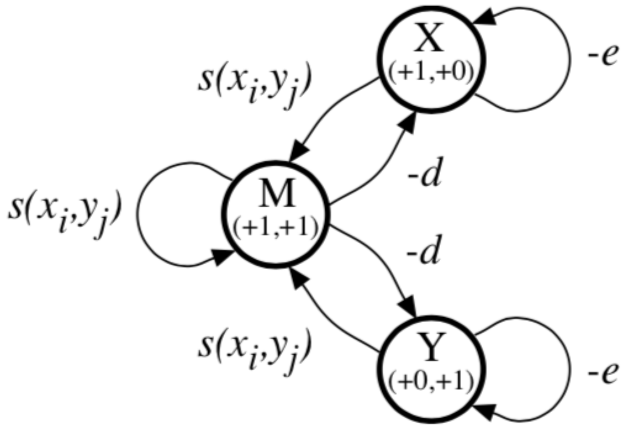
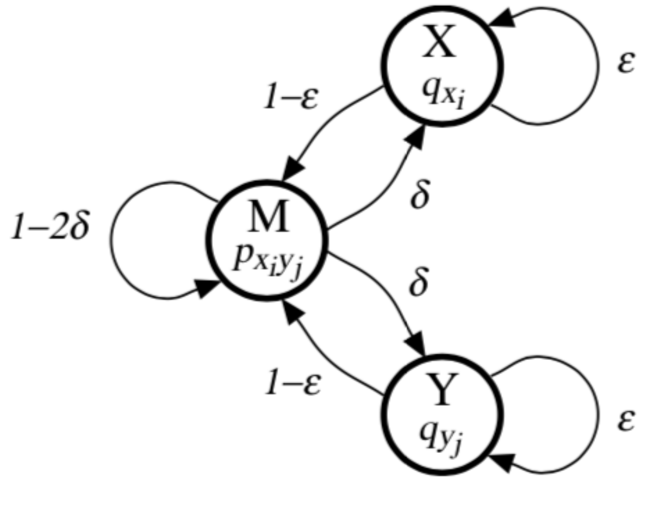


Figure 6.5: This picture summarises the above global alignment algorithm properly

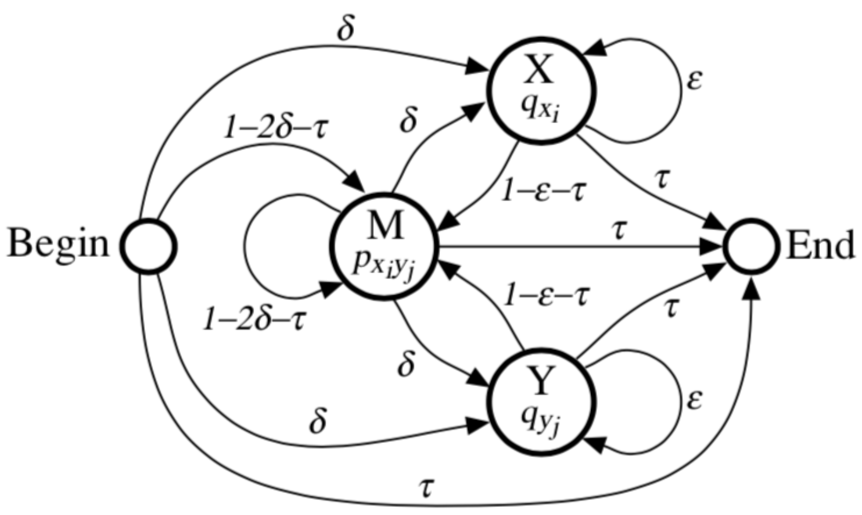
Now to convert the above algorithm into an HMM, we have to introduce two changes; bearing in mind that the HMM constructed emits pairwise alignment instead of a single observation:

- We define M, X, Y as the three underlying states and specify the associated transition probabilities.
- We also specify the emission probabilities $p_{x_iy_j}$ in the M state, q_{x_i} in the X state and q_{y_j} in the Y state.

The above HMM is picturised below:



One advantage of using the HMM to model the similarity between sequences is that we can use the probabilistic set up to find out the reliability of the alignment obtained by the dynamic programming and can find suboptimal alignments. By weighing all the possible alignments, we can score the similarity independent of any specific alignment.



Viterbi Algorithm

Initialisation:

$v^M(0,0) = 1$. All other $v^\bullet(i,o) = v^\bullet(0,j) = 0$.

Recursion:

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

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

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

Termination:

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

To find the optimal alignment, we trace back as before.

Let us now discuss how the similarity between two sequences can be measured irrespective of any specific alignment. We have to notice that when similarity is weak, it is hard to choose the correct alignment and proceed for significance. Now, we have a bypass to this problem using the HMM set up. We calculate the probability of two sequences being related as per the HMM by any alignment as shown below.

$$P(x,y) = \sum_{\text{all alignment } \pi} P(x,y,\pi)$$

Using the Forward algorithm makes the task really easy, algorithm for which is written below.

Initialisation:

$f^M(0,0) = 1, f^X(0,0) = f^Y(0,0) = 0.$
All $f^\bullet(i,-1), f^\bullet(-1,j)$ are set to 0.

Recursion:

$$\begin{aligned} f^M(i,j) &= p_{x_i y_j}\Big[(1-2\delta-\tau)f^M(i-1,j-1)+ \\ &\qquad (1-\epsilon-\tau)[f^X(i-1,j-1)+f^Y(i-1,j-1)]\Big] \\ f^X(i,j) &= q_{x_i}\Big[\delta f^M(i-1,j)+\epsilon f^X(i-1,j)\Big] \\ f^Y(i,j) &= q_{y_j}\Big[\delta f^M(i,j-1)+\epsilon f^Y(i,j-1)\Big] \end{aligned}$$

Termination:

$f^E = \tau\left[f^M(n,m)+f^X(n,m)+f^Y(n,m)\right]$

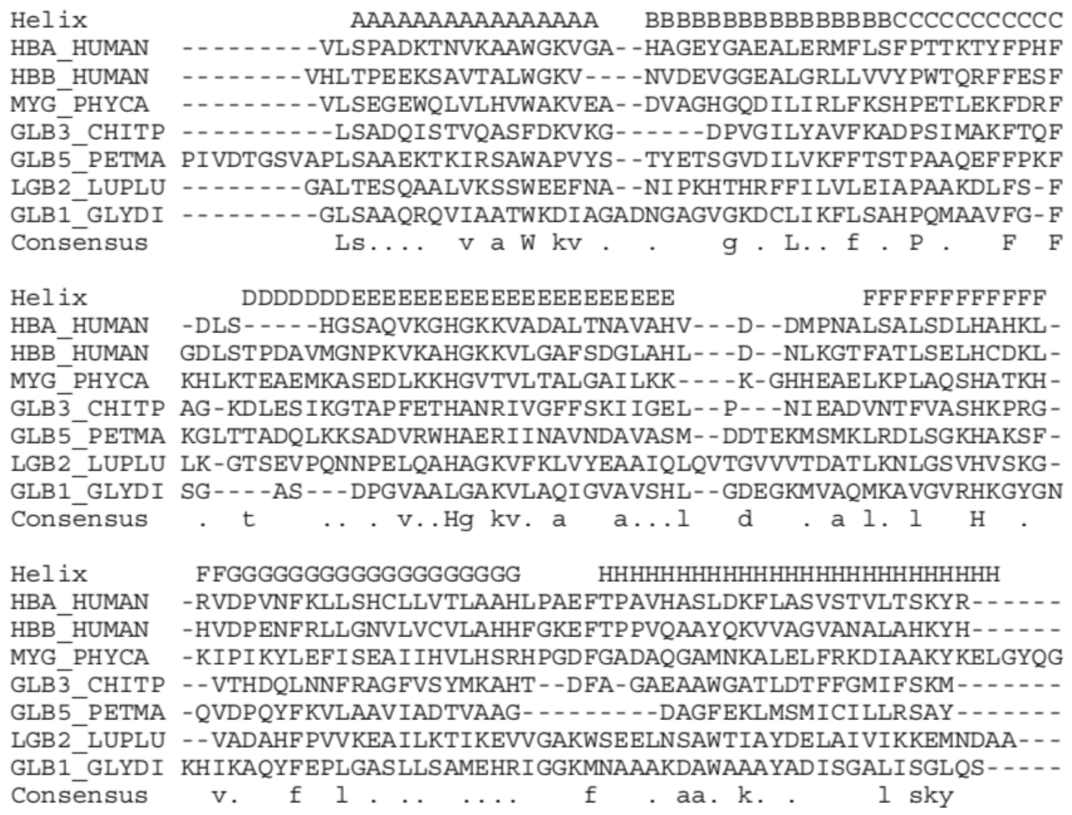
6.7 Profile HMM

So far we have discussed about pairwise alignment. But most often functional biological sequences come in a family having similar kind of function in different organisms. Sequences of a particular family are usually diverged from each other in their primary sequence due to duplication in the genome during cell division or by speciation which give rise to sequences with similar functions in related organisms. So, by identifying the family where the query sequence belongs gives us a hint about its function.

This is when we resort to Multiple Sequence Alignment(MSA) and profile HMM to probabilistically model the family of sequences and then check for similarity with query sequences. For this purpose we will focus on features which are conserved in the whole family. In this section we will not focus on getting the multiple sequence alignment from a set of sequences which could be built from available structural information or rather we will assume we are provided with one. So we will build an HMM out of the MSA which is called profile HMM and find out the probability that the query sequence belongs to the family.

6.8 Position Specific Score matrix

We can see in the figure below that the gaps tend to align with each other leaving ungapped regions in between, which is quite a general characteristic of an MSA. But these are unable to model variable length motifs which might have insertions or deletions in them.



```
Helix                 AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBBCCCCCCCCCCC
HBA_HUMAN  ---------VLSPADKTNVKAAWGKVGA--HAGEYGAEALERMFLSFPTTKTYFPHF
HBB_HUMAN  --------VHLTPEEKSAVTALWGKV----NVDEVGGEALGRLLVVYPWTQRFFESF
MYG_PHYCA  ---------VLSEGEWQLVLHVWAKVEA--DVAGHGQDILIRLFKSHPETLEKFDRF
GLB3_CHITP ----------LSADQISTVQASFDKVKG------DPVGILYAVFKADPSIMAKFTQF
GLB5_PETMA PIVDTGSVAPLSAAEKTKIRSAWAPVYS--TYETSGVDILVKFFTSTPAAQEFFPKF
LGB2_LUPLU --------GALTESQAALVKSSWEEFNA--NIPKHTHRFFILVLEIAPAAKDLFS-F
GLB1_GLYDI ---------GLSAAQRQVIAATWKDIAGADNGAGVGKDCLIKFLSAHPQMAAVFG-F
Consensus            Ls....  v a W kv .  .    g . L.. f . P .   F  F

Helix          DDDDDDDEEEEEEEEEEEEEEEEEEEEE            FFFFFFFFFFFF
HBA_HUMAN  -DLS-----HGSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL-
HBB_HUMAN  GDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHL---D--NLKGTFATLSELHCDKL-
MYG_PHYCA  KHLKTEAEMKASEDLKKHGVTVLTALGAILKK----K-GHHEAELKPLAQSHATKH-
GLB3_CHITP AG-KDLESIKGTAPFETHANRIVGFFSKIIGEL--P---NIEADVNTFVASHKPRG-
GLB5_PETMA KGLTTADQLKKSADVRWHAERIINAVNDAVASM--DDTEKMSMKLRDLSGKHAKSF-
LGB2_LUPLU LK-GTSEVPQNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG-
GLB1_GLYDI SG----AS---DPGVAALGAKVLAQIGVAVSHL--GDEGKMVAQMKAVGVRHKGYGN
Consensus   .  t    .. . v..Hg kv. a   a...l   d   . a l. l   H  .

Helix       FFGGGGGGGGGGGGGGGGGGG     HHHHHHHHHHHHHHHHHHHHHHHHHH
HBA_HUMAN  -RVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR------
HBB_HUMAN  -HVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH------
MYG_PHYCA  -KIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG
GLB3_CHITP --VTHDQLNNFRAGFVSYMKAHT--DFA-GAEAAWGATLDTFFGMIFSKM-------
GLB5_PETMA -QVDPQYFKVLAAVIADTVAAG---------DAGFEKLMSMICILLRSAY-------
LGB2_LUPLU --VADAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDAA---
GLB1_GLYDI KHIKAQYFEPLGASLLSAMEHRIGGKMNAAAKDAWAAAYADISGALISGLQS-----
Consensus    v.   f  l . ..  ....    f   . aa. k. .     l sky
```

Figure 6.6: MSA built by taking into account the structural information and conserved residues about these sequences. The last line in each MSA is called the consensus line, where upper case letters indicate an alignment that is identical in at least six of the sequences, lower case indicates alignment identical in four or five of the sequences, dot when alignment is identical in three of the sequences.

The Position Specific Score Matrix(PSSM) gives the distribution of the residues at each position of a conserved motif(ungapped region). So, we start by modelling the ungapped regions. The probability of a query sequence x according to this model is

$$P(x|M) = \prod_{i=1}^{L} e_i(x_i)$$

but we are more interested about the log odds ratio, which is

$$S = \sum_{i=1}^{L} log \frac{e_i(x_i)}{q_{x_i}}$$

Note that it is similar to the log odds ratio defined in the start of this chapter. The term $log\frac{e_i(x_i)}{q_{x_i}}$ is interpreted as the element $s(a, b)$ where instead of the residue $'b'$, we have the position index $'i'$. This is the reason why this approach is known as Position

Specific Score Matrix(PSSM). A PSSM can also be used for match in a longer sequence x of length N by finding the score for each starting point k from 1 to $N - L - 1$, L being the length of the PSSM.

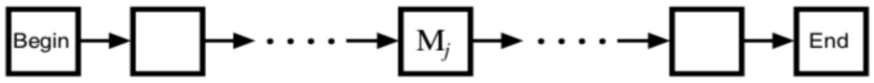
6.9 Profile HMM

Although PSSM has been quite successful in modelling ungapped regions, we still need to account for insertions and deletions. We develop a probabilistic model called ”profile HMM” for the same. We take an example to help illustrate how to build profile HMMs. Suppose we have a motif **”WEIRD”** and an MSA as follows.

WEIRD
WEIRE
WEIQH

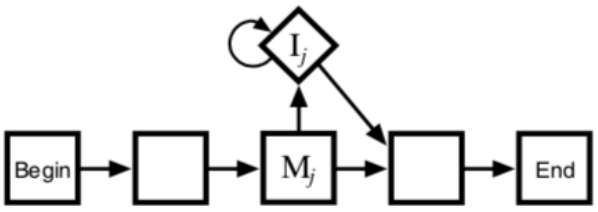
Backbone= columns that represent conserved motif of the family

The first step in building the HMM is to define a chain of repetitive match states corresponding to the backbone of the MSA, but with different emission probabilities. We can see PSSM can be modelled by such a structure as shown below.

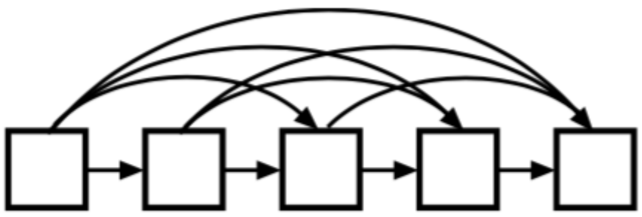


Here the alignment is trivial because there is no choice of transitions, rather we just follow the chain in a forward manner.

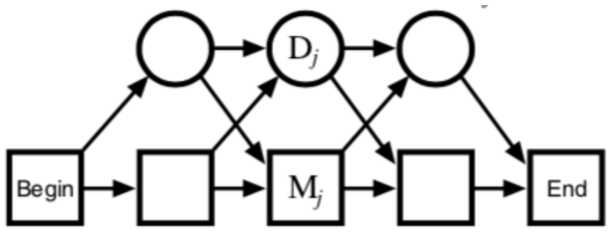
Next we deal with insertions. When the sequence contains a region that is not present in the model, that is an insertion in the sequence. Now to identify query sequences like **WECIRD**, we need to add an insert state between the corresponding match states but in general insertion can be anywhere, hence we need to add insert state between ant two consecutive match states. The emission probabilities for the insert states are the background probabilities.
Note that we could have a query sequence something like **WECLIRD**, hence we need to accommodate multi-residue insertions at each insertion step. Here is a single insert state:



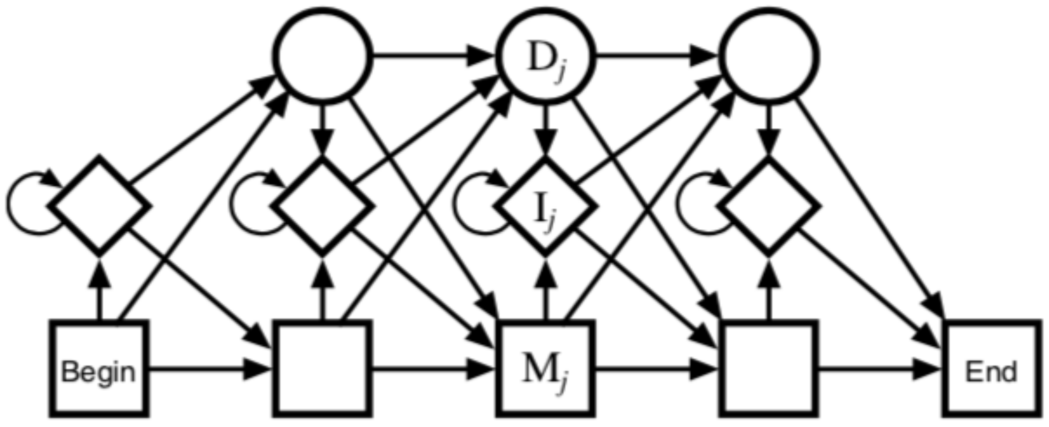
When there is a region in the model that is not present in the sequence, there is a deletion in the sequence. To model a query sequence like **WEID**, we need to add deletion states. But keep in mind that there could be multiple deletions as in **WED**. So, the deletions could be handled by adding jumps from any match state to any later non-neighbouring match state as shown below:



But we have seen how such a huge number of variables could be bypassed by using silent states. So, instead we use silent state D_j as described in Section 6.3. The deletion states emit '-' with probability 1.



The full resulting profile HMM is shown below.



Let us review the whole process in steps:

- The MSA λ of a set of sample sequences from the protein family is provided.
- We choose the most conserved columns $1, 2, \cdots, L$ of the MSA λ as our backbone and define match states $\mathrm{M}_1, \mathrm{M}_2, \cdots, \mathrm{M}_L$.
- Estimate probabilities a_{kl} and $e_{\mathrm{k}}(a)$

$$a_{\mathrm{kl}} = \frac{\mathrm{A}_{\mathrm{kl}} + 1}{\sum_{\mathrm{q}}(\mathrm{A}_{\mathrm{kq}} + 1)} \quad \text{and} \quad e_{\mathrm{k}}(a) = \frac{\mathrm{E}_{\mathrm{k}}(a) + 1}{\sum_{\mathrm{b}}[\mathrm{E}_{\mathrm{k}}(\mathrm{b}) + 1]}$$

where,
- A_{kl}= the count of transitions from k $\Rightarrow$ l in λ
- $E_k(a)$= the count of emissions of 'a' from state k in λ

The additional '1' added in the numerator and denominator is due to Laplace rule of pseudocounts.

Example of Profile HMM We are given with a multiple sequence alignment as follows:

```
VG--H
V---N
VE--D
IAADN
```

The length of the profile HMM is defined to be the number of match states except the begin and end states.

The length of the profile HMM is taken to be the average of the length of the sequences in the MSA. In this example the lengths are 3,2,3,5(before inserting gaps) whose average is 3.25. So, we take three match states M_1, M_2, M_3, insertion states I_0, I_1, I_2, I_3 and deletion states D_1, D_2, D_3.

In order to estimate the transition and emission probailities we have to assign labels to the MSA. Columns in the alignment with more than 50% of the rows as gaps are insertion states and the rest are match states.

V	G	-	-	H
V	-	-	-	N
V	E	-	-	D
I	A	A	D	N
M_1	M_2	I_2	I_2	M_3

This gives us the following labeled sequences:

V G H
M_1 M_2 M_3

V – N
M_1 D_2 M_3

V E D
M_1 M_2 M_3

I A A D N
M_1 M_2 I_2 I_2 M_3

From these labeled sequences, we can now find out the parameters, for example, for state M_1

$$\mathrm{e}_{\mathrm{M}_1}(\mathrm{V}) = \frac{3+1}{(3+1)+(1+1)+(18)}$$

since three of the four labeled sequences in state M_1 emit the protein "V", we have 3 in the numerator and to that we have added a pseudocount 1.
$(3+1)$ is contributed to the denominator for the protein V, $(1+1)$ for the protein I and $(0+1)$ for all the 18 other proteins total summing to 24, where 1 is the pseudocount.

Similarly the transition probability

$$a_{\mathrm{M}_2\mathrm{I}_2} = \frac{1+1}{(1+1)+(2+1)+(0+1)}$$

There is only one transition from M_2 to I_2 which is there in the fourth sequence, hence the term $(1+1)$ in the numerator. The three sums in the denominator correspond to all the three transitions possible from the state M_2 and there are two transitions from M_2 to M_3, one transition from M_2 to I_2 and zero from M_2 to D_3

Use Viterbi algorithm to find the optimal alignment and the probability of that alignment will give the probability that the sequence belongs to that protein family and use the forward algorithm to find the probability of similarity without regard to any specific alignment.

Bibliography

[1] Walter Zucchini and Iain L. MacDonald. *Hidden Markov Models for Time Series. An Introduction Using R*. Florida: CRC Press, 2009.

[2] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis. Probabilistic models of proteins and nucleic acids*. Cambridge: Cambridge University Press, 1998

[3] Lawrence R Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Proceedings of the IEEE, Vol. 77, NO. 2, February 1989.

[4] Mihaela Angelova, Slobodan Kalajdziski and Ljupco Kocarev. *Computational Methods for Gene Finding in Prokaryotes*.ICT Innovations 2010, Web Proceedings, ISSN 1857-7288.

[5] https://www.stat.cmu.edu/ larry/=stat705/Lecture16.pdf

[6] https://www.cs.helsinki.fi/u/lmsalmel/cmsb09/lectures/CompMSysBio2009-Lecture2.pdf

[7] http://www.cs.cmu.edu/ durand/03-711/2014/Lectures/

[8] https://www.cs.princeton.edu/ mona/Lecture/

[9] https://www.cs.us.es/ fran/students/julian/genindex.html