International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878, Impact Factor (2015): 3.791

Parameters Estimation of HMM Model for the Classification of Biological Sequences

Shitalkumar R. Sukhdeve[1], Manish P. Kurhekar[2]

[1]VNIT, South Ambazari Road, Nagpur, Maharashtra 440010, India
[2]VNIT, South Ambazari Road, Nagpur, Maharashtra 440010, India

Abstract: ***Classification of the biological data is a fundamental topic of research in computer science, especially in bioinformatics. Given an unknown sequence, finding the common patterns in it which appears in the known sequences and placing it into the most probable family is the classification problem. The bacteria's of bacillus and clostridia are so conserved that they show nearly same characteristics which make them difficult to classify. In this paper we have tried to classify such biological sequences with the help of Hidden markov model (HMM) in combination with the Baum Welch model. In this experiment, we have tried to estimate the parameters of HMM so that the classification process should converge in the minimum number of iterations. We have achieved approximately 89% and 90.73% accuracy of classification in case of bacillus and clostridia respectively.***

Keywords: Hidden Markov Model, HMM, DNA Sequences, biological, Baum Welch algorithm, parameter estimation.

1. Introduction

Most of the biological data is available in the form of biological sequences which are nothing but the sequence of characters a, g, c, t and u stands for adenine, guanine, cytosine, thymine and uracil respectively. This five letters represents the bases. In DNA (deoxyribonucleic acid) and RNA (ribonucleic acid) sequences out of five any four bases are generally present. For example, "ctggcggcgtgcctaata" is a segment of DNA sequence of Alicyclobacillus_sp._FR-6; _AJ133635 bacillus. This large amount of biological data is studied in the field of bioinformatics. One of the most important practices is to classify such biological sequences into different classes. The classification of biological sequences requires more efficient methodologies so that biological sequences grouped together in the most probable classes.

One of the method of patterns or motifs discovery in protein and nucleotide sequences has been use to establish the evolutionary relationship among sequences. Organization of the protein cluster into Hierarchical trees is also used to establish functional and evolutionary relationships among proteins [6]. The problems with such methodology are that sequences can have limited homology but proteins can also have structural and mechanistic similarities. Even common ancestry is not visible through alignment [7].Promoter predication is also used to classify the sequences. But the development of efficient models for promoter prediction includes the relatively low number of characterized promoters and poor understanding of the signals for start and stop of transcription and translation, especially in eukaryotes [8]. The division of patterns can be deterministic and probabilistic on the general level. A deterministic pattern either matches given string or not. While for probabilistic patterns, probabilistic models are generated to assign some probabilistic value to each sequence. The higher relevance between the probabilities ensures the better is match between sequence and pattern. The problem with the deterministic approach is that it cannot capture the invisible information in a pattern. The types of patterns we have discussed so far are explicit in nature that it shows the user the important characteristics of the occurrences of a pattern. The BLAST tool that is popular for its simplest nearest neighbour approach [10]. It exploits pair wise local alignments to measure sequence similarity. The BLAST technique compares the queried unknown protein with the labelled and classified sequences in the database and produces normalized alignment scores for each and every comparison by calculating the expected value i.e. E-value. The best pair wise alignment scores produced which is the minimum E-value gives the bases of classification. Support vector machines (SVMs) [11] have been also applied to protein homology detection problems. Such an approach, which has been introduced in [12], feeds probabilistic score values from all motifs available (nearly 10000) in the BLOCKS database [12] into an SVM classifier. Obviously, this scheme uses only local features but the dimensionality of the input space is extremely high. Another method that combines the HMM with SVMs has been proposed in [13] for finding remote protein homologies. In particular, an HMM is first trained to model a protein family, and then the observed probabilities (in the log space) of each sequence with respect to each parameter of the HMM are calculated. The obtained gradient-log-probability vectors are applied to an SVM to identify the decision boundary between the family and the rest of the protein universe. The Techniques Neural Network based Classifier is better for non linear and non noisy data only. Rough Set based Classifier needs extra space and time and it also do not produce analytical output [14].Sometimes, representation of the patterns as some discriminating rule is more advantageous. Especially, discrimination rule, which decides whether a given sequence is an occurrence of the modelled pattern or not. Such a discrimination rule can be based on some stochastic model, such as hidden Markov model (HMM).Statistical methods are well suited for classification of problems and one of those is the Hidden Markov Model (HMM). This is because of its richness in mathematical structure. The basic theory was published in a series of classic papers by Baum and his colleagues [1]-[2] in the late 1960s and early 1970s and was implemented for speech processing applications by Baker at CMU, and by Jelinek and his colleagues at IBM in the 1970s.[9] The structure of Hidden Markov Model is well suited for the analysis of biological sequences.

In our approach, we have simulated Hidden Markov Model for training and Baum Welch for optimizing the parameters.

We have used two datasets of gene 16S rRNA. The datasets are of two different families of bacteria bacillus and clostridia. We would train HMM separately by both the datasets. After generating and training the model we can group them into clusters and compare a unified sequence with these models on the basis of likelihood probability to determine the one which has the closest match to sequence. The primary advantage of HMMs over other sequence matching algorithms like distance based sequence matching algorithms is that it can automatically estimate, or be trained for, a cluster of unaligned sequences. With the proposed approached, we have achieved approximately 89% and 90.73% accuracy of classification in case of bacillus and clostridia respectively. The paper is organized as follows: Section II describes the HMM model used for the classification of the biological sequences. In Section III, we have presented the experimental results. Section IV is the conclusion.

2. HMM and Sequence Classification

A Hidden Markov Model is a generalization of a Markov chain, in which each ("internal") state is not directly observable (hence the term hidden) but produces ("emits") an observable random output ("external") state, also called "emission", according to a given stationary probability law. In this case, the time evolution of the internal states can be induced only through the sequence of the observed output states. If the number of internal states is N, the transition probability law is described by a matrix with N times N values; if the number of emissions is M, the emission probability law is described by a matrix with N times M values. A model is considered defined once given these two matrices and the initial distribution of the internal states. We now formally define the elements of an HMM, and explain how the model generates observation sequences.

An HMM is characterized by the following:

1) N, the number of states in the model. Although the states are hidden, for many practical applications there is often some physical significance attached to the states or sets of states of the model. Generally the states are interconnected in such a way that any state can be reached from any other state (e.g., an ergodic model). We denote the individual states as $S = \{S_1, S_2, ..., S_N\}$, and the state at time t as q_t.

2) M, the number of distinct observation symbols per state, i.e., the discrete alphabet size. The observation symbols correspond to the physical output of the system being modelled. We denote the individual symbols as $V = \{v_1, v_2, ..., v_M\}$.

3) The state transition probability distribution $A = \{a_{ij}\}$

Where,

$$a_{ij} = P[q_{t+1} = S_j \mid q = S_i], \quad 1 \le i, j \le N \tag{1}$$

For the special case where any state can reach any other state in a single step, we have $a_{ij} > 0$ for all i, j. For other types of HMMs, we would have $a_{ij} = 0$ for one or more (i, j) pairs.

4) The observation symbol probability distribution in state j, $B = \{b_j(K)\}$ where,

$$b_j(K) = P[v_k att \mid q_t = S_j] \quad 1 \le i \le N \tag{2}$$

5) The initial state distribution $\pi = \{\pi_i\}$ where,

$$\pi_i = P[q_1 = S_i], \quad 1 \le i \le N \tag{3}$$

Given appropriate values of N, M, A, B and π, the HMM can be used as a generator to give an observation sequence

$$O = O_1 O_2 .. O_T . \tag{4}$$

(Where each observation O is one of the symbols from V, and T is the number of observations in the sequence) as follows:

1) Choose an initial state $q_1 = S$, according to the initial state distribution π.
2) Set $t = 1$.
3) Choose $O_t = v_k$ according to the symbol probability distribution in state S_i i.e. $b_i(k)$.
4) Transit to a new state $q_{t+1} = S_j$ according to the state transition probability distribution for state S_j, i.e. a_{ij}.
5) Set $t = t+1$; return to step 3) if $t < T$; otherwise terminate the procedure.

The above procedure can be used as both a generator of observations, and as a model for how a given observation sequence was generated by an appropriate HMM.

It can be seen from the above discussion that a complete specification of an HMM requires specification of two model parameters (N and M), specification of observation symbols, and the specification of the three probability measures A, B and π. For convenience, we use the compact notation $\lambda = (A, B, \pi)$ to indicate the complete parameter set of the model.

There are two approaches to the learning task based on the form of the database available for learning process, supervised and unsupervised training. If the training examples contain both the inputs and outputs of a process, we can perform supervised learning. It is done by equating inputs to observations, and outputs to states, but we have only the inputs datasets (gene 16S RNA) in the training data then we must use unsupervised training. Unsupervised training guesses a model that may have produced those observation sequences.

2.1 Forward Algorithm

The Forward Algorithm is used for the Training Phase of the HMM process. Each state's values are dependent only on the pervious state. The aim of this problem is to find the

probability $P(O|\lambda)$ of the observation sequences, $O=\{O_1,O_2,...,O_T\}$, given the model $\lambda=(A,B,\pi)$. Consider a forward variable $\alpha_t(i)$, defined as:

$$\alpha_t(i)=P(o_1,o_2,..o_t,q_t=i|\lambda) \quad (5)$$

Where t represents time and i is the state. This gives that $\alpha_t(i)$ will be the probability of the partial observation sequence $o_1,o_2,..o_t$ (until time t) when being in state i at time t. The forward variable for all N states at time t multiplied with their corresponding state transition probability, a_{ij}, and by the emission probability $b_j(o_{t+1})$.

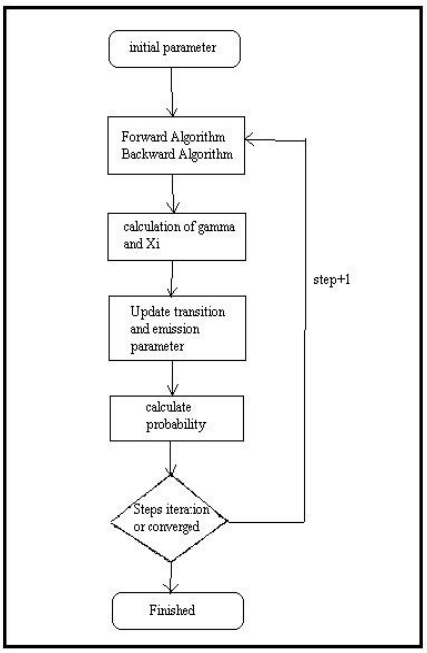

Figure 1: Flow chart for learning of model

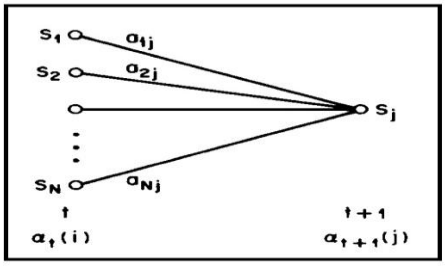

Figure 2: Illustration of sequence of operation required for the computation of the forward variable $\alpha_{t+1}(j)$ where S_j indicates state j [3]

The initialization step 1) arbitrarily defines $\beta_T(i)$ to be 1 for all i. Step 2), which is illustrated in Fig.3, shows that in order to have been in state S_i at time t, and to account for the observation sequence from time $t+1$ on, you have to consider all possible states S_j at time $t+1$ accounting for the transition from S_i to S_j, (the a_{ij} term), as well as the observation o_{t+1} in state j (the $b_t(o_{t+1})$ term), and then account for the remaining partial observation sequence from state.

The computation of $\beta_t(i)$, $1\le t\le T$, $1\le i\le N$ requires on the order of N^2T calculations, and can be computed in a lattice structure.

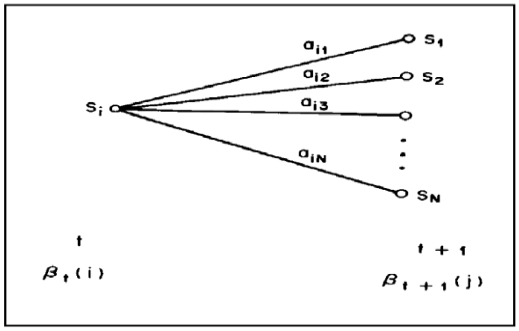

Figure3: Illustration of the sequence of operations required for the computation of the backward variable $\beta_t(i)$ [8].

2.2 Baum-Welch Algorithm

In this section we discuss one iterative procedure, based primarily on the classic work of Baum and his colleagues, for choosing model parameters. In order to describe the procedure for reestimation (iterative update and improvement) of HMM parameters, we first define $\xi_t(i,j)$, the probability of being in state S_i at time t, and state S_j at time $t+1$, given the model and the observation sequence, i.e.

$$\xi_t(i,j)=P(q_t=S_i,q_{t+1}=S_j|O,\lambda) \quad (6)$$

The above equation can be shown as sequence of events in the figure 4.6.We can write $\xi_t(i,j)$ using forward and backward variables as follows:

$$\xi_t(i,j)=\frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} \quad (7)$$

$$=\frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \quad (8)$$

Here, numerator term is $P(q_t=S_i,q_{t+1}=S_j,O|\lambda)$ and division by $P(O|\lambda)$ gives the desired probability measure.

Further we have to define $\gamma_t(i)$, which is defined as the probability of being in state S_i at time t, given the observation sequence O, and the model λ.

$$\gamma_t(i)=P(q_t=S_i|O,\lambda) \quad (9)$$

Equation (1) can be expressed in terms of forward-backward variables as follows:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(j)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(j)} \quad (10)$$

Since $\alpha_t(i)$ accounts for the partial observation sequence $o_1, o_2, ..o_t$ and state S_i at t, while $\beta_t(i)$ accounts for the remainder of the observation sequence $o_{t+1}, o_{t+2}, ...o_T$ given state S_i at t. The normalization factor $P(O \mid A) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(j)$ makes $\gamma_t(i)$ a probability measure so that

$$\sum_{i=1}^{N} \gamma_t(i) = 1 \quad (11)$$

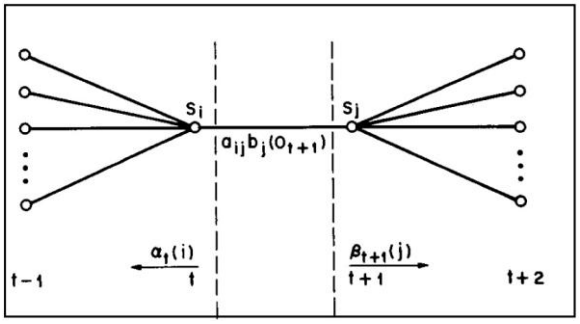

Figure 5: Illustration of the sequence of operations required for the computation of the joint event that the system is in state S_i at time t and state S_j at time $t+1$ [4]

We can relate $\gamma_t(i)$ to $\xi_t(i, j)$, by summing over j, giving

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j) \quad (12)$$

If we sum $\gamma_t(i)$ over the time index t, we get a quantity which an be interpreted as the expected (over time) number of times that state S_i is visited, or equivalently, the expected number of transitions made from state S_i (if we exclude the time slot $t = T$ from the summation). Similarly, summation of $\xi_t(i, j)$, over t (from $t = 1$ to t = T-1) can be interpreted as the expected number of transitions from state S_i to state S_j. [4] That is

$$\sum_{i=1}^{T-1} \gamma_t(i) = \text{expected number of transition from } S_i \quad (13)$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{expected number of transitions from } S_i \text{ to } S_j \quad (14)$$

We can use above two formulas to give a method for reestimation of parameters of an HMM. The reestimation formulas can be given as follows:

$$\bar{\pi}_i = \text{expected frequency in state } S_i \text{ at time}(t=1) = \gamma_1(i) \quad (15)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transition from state } S_i \text{ to state } S_j}{\text{expected number of transition from state } S_i} \quad (16)$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (17)$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state j}} \quad (18)$$

$$= \frac{\sum_{t=1 \; s.t. O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad (19)$$

Let's define the current model a as $\lambda = (A, B, \pi)$ and we define the reestimated model as $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ then it has been proven by Baum and his colleagues [5], [6] that either

1)The initial model λ defines a critical point of the likelihood function, in which case

$\bar{\lambda} = \lambda$; (20)

Or

2)Model $\bar{\lambda}$ is more likely than model λ in the sense that $P(O \mid \bar{\lambda}) > P(O \mid \lambda)$ i.e., we have found a new model $\bar{\lambda}$ from which the observation sequence is more likely to have been produced.

As per the above procedure, if we iteratively use $\bar{\lambda}$ in place λ of and repeat the reestimation calculation, we then can improve the probability of O being observed from the model until some limiting point is reached. The final result of this reestimation procedure is called a maximum likelihood estimate of the HMM. It should be pointed out that the forward—backward algorithm leads to local maxima only, and that in most problems of interest, the optimization surface is very complex and has many local maxima.

The reestimation formulas can be derived directly by maximizing (using standard constrained optimization techniques) Baum's auxiliary function

$$Q(\lambda,\bar{\lambda}) = \sum_{Q} P(Q \mid O,\lambda)\log[P(O,Q \mid \bar{\lambda})] \quad (21)$$

Over $\bar{\lambda}$. It has been proven by Baum and his colleagues [9] [10] that maximization of $Q(\lambda,\bar{\lambda})$ leads to increased likelihood, i.e.

$$\max_{\bar{\lambda}}[Q(\lambda,\bar{\lambda})] \Rightarrow P(O \mid \bar{\lambda}) \geq P(O \mid \lambda). \quad (22)$$

Eventually the likelihood function converges to a critical point. An important aspect of the reestimation procedure is the stochastic constraints of the HMM parameters, namely

$$\sum_{i=1}^{N} \bar{\pi} = 1 \qquad 1 \leq i \leq N \quad (23)$$

$$\sum_{j=1}^{N} \bar{a}_{ij} = 1 \qquad 1 \leq i \leq N \quad , \quad (24)$$

$$\sum_{k=1}^{M} \bar{b}_j(k) = 1 \qquad 1 \leq i \leq N \, , \quad (25)$$

are automatically satisfied at each iteration.

Parameter reuse in Baum Welch algorithm:

HMM model is often learned from training data using the Baum-Welch algorithm. It's an iterative process for estimation HMM parameters and guaranteed to be converged, but have some drawback such as local maxima and convergence requires more number of iterations.

As mentioned in first issue the local maxima of Baum Welch can almost be solved by carefully starting the initial parameter selection and issue of number of iterations can be reduced in the following manner: Usually the O(t) length of observation sequences is in our work bacillus and clostridia have large number of strings. The Baum-Welch algorithm is trained incrementally on all observation sequences considering one by one observation sequence, at each value of t it consider input (O_t) and computes probability of this observation sequence. Here we consider a common parameter S_0 (x_0) on all observation sequence O= (2, 3, 4… T-1), which stores the maximum probability for this observation symbol calculation for next observation sequence (O_{t+1}). So the ratio of observation probability for next iteration is as follows:

$$\frac{b_j(o)}{b_o(o)} = \frac{b_j(o_j)}{b_j(x_j)}, \quad 1 \leq j \leq N \quad (26)$$

The left hand side of equation is likelihood function of observation probability scaled by factor $b_o(x)$. The right hand side of equation gives the testing criteria for the sufficiency of x_j for state S_j vs. state S_0. The dimensions at each state j are sufficient to discriminate it from the common state S_0. For time t=2, 3, …, T, all the parameter estimation procedure is same except for the state likelihood function $b_j(x)$.

The reestimation procedure of the HMM parameters using parameter reuse is provided here.

HMM reestimation formulas:

1. Re-estimation method of γ id same as estimated by conventional BW.
2. Re-estimation of ξ

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}\dfrac{b_j(o_j)[t+1]}{b_o(x_o)[t+1]}\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{m=1}^{N}\alpha_t(i)a_{im}\dfrac{b_m(o_m)[t+1]}{b_o(x_o)[t+1]}\beta_{t+1}(m)} \quad (27)$$

We describe the reuse of parameters for training of HMM that uses maximum probability of observation sequence in the forward and backward procedures. The HMM trained with this way can reduce the number of iteration for convergence.

Scaling of parameter to avoid floating point underflow:

From the definition of $\alpha_t(i)$, we can see that $\alpha_t(i)$ consists of the sum of a large number of terms, each of the form

$$\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(O_s) \quad (28)$$

Since each a and b term is less than 1, it can be seen that as t start to get big, each term of $\alpha_t(i)$ starts to head exponentially to zero. For sufficiently large t (e.g. 100 or more) the dynamic range of the $\alpha_t(i)$ computation will exceed the precision range of essentially any machine (even in double precision). Hence the only reasonable way of performing the computation is by incorporating a scaling procedure. Hence, we need to scale parameters to avoid floating point underflow.

We solve this problem differently for each algorithm e.g. forward, backward and Baum-Welch.

Scaling of forward and backward variables:

The basic scaling procedure which is used is to multiply $\alpha_t(i)$ by a scaling coefficient that is independent of i (i.e., it depends only on t), with the goal of keeping the scaled $\alpha_t(i)$ within the dynamic range of the computer for $1 \leq j \leq N$.

A similar scaling is done to the $\beta_t(j)$ coefficients (since these also tend to zero exponentially fast) and then, at the end of the computation, the scaling coefficients are canceled out exactly.

The main difference between the scaled and non-scaled forward algorithm lies in steps two and four. By induction, the forward variable can be found in terms of the none scaled as:

$$\hat{\alpha}_{t-1}(j) = \prod_{r=1}^{t-1} c_T \;\; \alpha_{t-1}(j) \, , \; 1 \leq j \leq N \quad (29)$$

The ordinary induction step can be found as:

$$\alpha_t(i) = b_i(o_t)\sum_{j=1}^{N}\alpha_{t-1}(j)a_{ji}\ ,\ 1 \le j \le N \tag{30}$$

Now we can write equation:

$$\hat{\alpha}_t(i) = \frac{b_i(o_t)\sum_{j=1}^{N}\hat{\alpha}_{t-1}(j)a_{ji}}{\sum_{k=1}^{N}b_i(o_t)\sum_{j=1}^{N}\hat{\alpha}_{t-1}(j)a_{jk}} \tag{31}$$

$$= \frac{b_i(o_t)\sum_{j=1}^{N}\prod_{r=1}^{t-1}c_r\ \alpha_{t-1}(j)a_{ji}}{\sum_{k=1}^{N}b_i(o_t)\sum_{j=1}^{N}\prod_{r=1}^{t-1}c_r\ \alpha_{t-1}(j)a_{ji}} \tag{32}$$

$$= \frac{\prod_{r=1}^{t-1}c_r\ \ b_i(o_t)\sum_{j=1}^{N}\alpha_{t-1}(j)a_{ji}}{\prod_{r=1}^{t-1}c_r\ \sum_{k=1}^{N}\ b_i(o_t)\sum_{j=1}^{N}\alpha_{t-1}(j)a_{ji}} \tag{33}$$

$$= \frac{\alpha_t(i)}{\sum_{k=1}^{N}\alpha_t(i)}\ ,\ 1 \le j \le N \tag{34}$$

As (4.41) shows, each $\alpha_t(i)$ is scaled by the sum over all states of $\alpha_t(i)$ when the scaled forward algorithm is applied.

The termination (step 4) of the scaled forward algorithm, evaluation of $P(O|\lambda)$, must be done in different way, because the sum of $\hat{\alpha}_T(i)$ cannot be used, it is scaled already. However the following properties can be used:

$$\prod_{r=1}^{T}c_r\sum_{k=1}^{N}\alpha_T(i) = 1 \tag{35}$$

$$\prod_{r=1}^{T}c_r P(O \mid \lambda) = 1 \tag{36}$$

$$P(O \mid \lambda) = \frac{1}{\prod_{r=1}^{T}c_r} \tag{37}$$

The scaled backward algorithm can be found more easily, since it will use the same scale factor as the forward algorithm. The notations used are similar to the forward variable notations, $\beta_t(i)$ denote the non-scaled backward variable, $\hat{\beta}_t(i)$ denote the scaled and iterated variant of $\beta_t(i)$.

Scaling of training Variables during Baum-Welch algorithm

With scaled variables we will find $\xi_t(i,j)$ as:

$$\xi_t(i,j) = \frac{\hat{\alpha}_t(i)a_{ij}b_j(o_{t+1})\hat{\beta}_{t+1}(j)}{\sum_{i=1}^{N}\sum_{m=1}^{N}\hat{\alpha}_t(i)a_{ij}b_j(o_{t+1})\hat{\beta}_{t+1}(j)} \tag{38}$$

$$= \frac{\prod_{r=1}^{t}c_r\ \ \alpha_t(i)a_{ij}b_j(o_{t+1})\ \prod_{r=t+1}^{T}c_r\ \ \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\ \prod_{r=1}^{t}c_r\ \ \alpha_t(i)a_{ij}b_j(o_{t+1})\ \prod_{r=t+1}^{T}c_r\ \ \beta_{t+1}(j)} \tag{39}$$

$$= \frac{\prod_{r=1}^{t}c_r\ \ \prod_{r=t+1}^{T}c_r\ \ \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\ \prod_{r=1}^{t}c_r\ \ \prod_{r=t+1}^{T}c_r\ \ \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \tag{40}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \tag{41}$$

$\gamma_t(i)$ is same if scaled or not scaled such as:

$$\gamma_t(i) = \frac{\hat{\alpha}_t(i)\hat{\beta}_t(j)}{\sum_{i=1}^{N}\hat{\alpha}_t(i)\hat{\beta}_t(j)} \tag{42}$$

$$= \frac{\prod_{r=1}^{t}c_r\ \ \hat{\alpha}_t(i)\ \prod_{r=r+1}^{t}c_r\ \ \hat{\beta}_t(j)}{\sum_{i=1}^{N}\ \prod_{r=1}^{t}c_r\ \ \hat{\alpha}_t(i)\ \prod_{r=r+1}^{t}c_r\ \ \hat{\beta}_t(j)} \tag{43}$$

$$= \frac{\alpha_t(i)\beta_t(j)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(j)} \tag{44}$$

As the above equations shows, $\gamma_t(i)$ and $\xi_t(i,j)$ are the same if scaled or not scaled.

Since π, A and B uses $\gamma_t(i)$ and $\xi_t(i,j)$ for calculation, will these probabilities also be independent of which forward and backward variables are used (scaled and none scaled).

All these algorithms mentioned in above section solve the problem of clustering biological sequences into different groups according to their functionality or sequence structure. The simulation of HMM based on these algorithm and implementation issues are presented in the following section of report.

3. Experimental Setup and Results

In our approach, the algorithm we have followed for the generation of model for the given sequences can be shown in flowchart given below.

Fig. 1 shows that the first step in this process is initialization of starting parameters for HMM (calculating the initial state probabilities, the transition probabilities, and the emission probabilities). Once these values have been initialized, we use the Forward, the Backward and the Baum-Welch Algorithms. The next step is learning of model. This part requires the Baum-Welch Algorithm where iterative approach is applied on the given observation sequence to maximize likelihood for given observation sequence until the model converges and a feasible model is obtained.

We repeat this process until a satisfactory model is generated. Once a feasible model is generated, then we proceed to Evaluation phase of HMM. Evaluation phase of HMM uses the Forward and the backward algorithm to test and validate the learned model.

We have applied the proposed approach and used a simulated model of the same. We have experimented with the two types of data sets of 16S rRNA sequences of highly conserved bacteria's bacillus and clostridia.

Here our aim is to estimate the parameters of HMM for the models generated for bacillus and clostridia and to classify them in their affinity groups. The experiments and their results are presented as follows.

Experiment 1:

We have used 16S bacillus data set for the simulation of HMM. The fig. 3.1 shows the snapshot text file of dataset of bacillus sequences. For each species, first line contains its name and next line includes DNA sequence of that species. For example, first line of the figure contains name of bacillus

">Alicyclobacillus_sp._FR-6;_AJ133635".Here._FR-6; _AJ133635 is name of species and Alicyclobacillus is the type of bacteria. Next line "ctggcggcgtgcctaatacatgcaagtcgagcggtttt…"is a DNA sequence of that bacterium. In the same way, this file contains name of 3821 different species and DNA sequence of each species 1000

DNA sequences had been used for the training and 2844 sequences had been used for testing purpose in this model. For this model we were able to achieve 89% of correctness. Figure illustrate the tradeoff of probability of observation of given sequences P(O|λ) versus number of iteration for training of HMM. P(O|λ) is given in logarithmic scale. It can be seen from the Fig. that in the very first iteration the probability of outputting the sequence is in the range of "E-118" but after second iteration onwards, it improved up to the "E-12". Here, it may require on an average five iterations to converge.

We have trained the model for bacillus in previous section using BW algorithm with some randomly chosen initial parameters (A and π) and B according to the method we have described for HMM model, which are as follows

$$A = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{bmatrix} \quad (45) \qquad \pi = \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} \quad (46)$$

$$B = \{b_{ij}\} = \begin{bmatrix} 0.2 & 0.25 \\ 0.1 & 0.15 \\ 0.6 & 0.55 \\ 0.1 & 0.05 \end{bmatrix} \quad (47)$$

Estimated parameters are as follows

$$A = \{a_{ij}\} = \begin{matrix} 0.49992600581216 & 0.50007399418784 \\ 0.38388139236515 93 & 0.41011863648407 \end{matrix} \quad (48)$$

```
>Alicyclobacillus_sp._R-10926;_AJ438297
gacgaacgctggcggcgtgcctaatacatgcaagtcgagcggacctcttykga
ggtcagcggcggacgggtgaggaacacgtgggtaatctgcctttcagaccgga
ataacgcccggaaacgggcgctaatgccggatacgcccgcgaggaggcatctt
cttgcggggaaaggcccrattgggccgctgagagaggagcccgcggcgcatta
gctngttggcggggtaacggcccaccaaggcgacgatgcgtagccgacctgag
agggtgaccggccacactgggactgagacacggcccagactcctacgggaggc
agcagtagggaatcttccgcaatgggcgcaagcctgacggagcaacgccgcgt
gagcgaagaaggccttcgggttgtaaagctctgttgctcggggagagcggcat
ggggagtggaaagccccatgcgagacggtaccgagtga
>Alicyclobacillus_sp._DSM_11984;_AB059668
agagtttgatcctggctcaggacgaacgctggcggcgtgcctaatacatgcaa
gtcgagcgggtctcttcggaggccagcggcggacgggtgaggaacacgtgggt
aatctgcctttcaggccggaataacgcccggaaacgggcgctaatgccggata
cgcccgcgaggaggcatcttcttgcgggggaaggcccaattgggccgctgaga
gaggagcccgcggcgcattagctagttggcggggtaacggcccaccaaggcga
cgatgcgtagccgacctgagagggtgaccggccacactgggactgagacacgg
```

Figure 6: Datasets of bacillus for training

$$\pi = \begin{bmatrix} 0.20562829\ 6 \\ 0.79437170\ 3 \end{bmatrix} \quad (49)$$

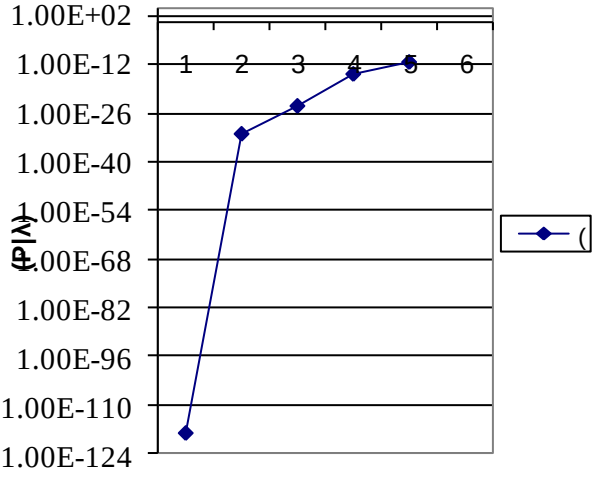

Figure 7: Estimated (P|λ) and number of iterations require converging for Bacillus dataset while training HMM

$$B = \{b_{ij}\} = \begin{bmatrix} 0.1996174657444866 & 0.1550196041515629 60 \\ 0.1996186236507420 4 & 0.1550231823246566 \\ 0.4011448890020310 0 & 0.5349388020010002 1 \\ 0.1990190162870172 4 & 0.1510184114245624 6 \end{bmatrix} \quad (50)$$

Experiment 2: In Fig: 3.3 second data set used for our simulation. We have considered here gene 16S rRNA Clostridia dataset.

Figure illustrate the tradeoff of probability of observation of given sequences P(O|λ) versus number of iteration for training of HMM. P(O|λ) is given in logarithmic scale. Figure shows that in the very first iteration, the probability of outputting the sequence is in the range of "E-124" but after second iteration onwards, it improved up to the "E-2".Here, we have experimented with the data set of clostridia. We have used the same initial parameters as used for bacillus dataset while training and estimating with HMM and Baum Welch algorithms.

Estimated parameters are as follows

$$A = \{a_{ij}\} = \begin{bmatrix} 0.49943841540733364 & 0.5005615845926663 \\ 0.5004598730458814 & 0.4095401269541187 \end{bmatrix} \quad (51$$

$$\pi = \begin{bmatrix} 0.55 \\ 0.45 \end{bmatrix} \quad (52)$$

$$B = \{b_{ij}\} = \begin{bmatrix} 0.19992936\ 61702532 & 0.15406348\ 870691322 \\ 0.19992936\ 518441037 & 0.15406348\ 844476941 \\ 0.44607772\ 1001 & 0.54038177\ 7 \\ 0.15406348\ 870691322 & 0.14406349\ 17506251 \end{bmatrix} \quad (53)$$

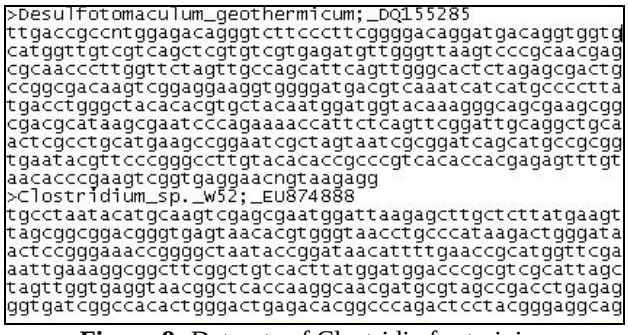

Figure 8: Datasets of Clostridia for training

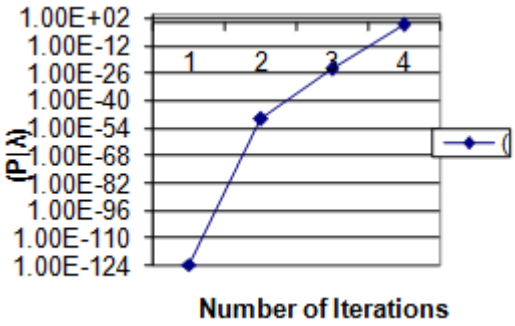

Figure 9: Estimated (P|λ) and Number of Iterations require converging for Clostridia data set

4. Conclusion

We have used the HMM in combination with the Baum Welch model successfully to classify the two highly conserved bacteria's i.e. bacillus and clostridia with the correctness of 89% and 90% respectively. We have estimated the parameters of HMM model generated for the classification of the biological sequences so that the classification process can converge in minimum time i.e. in minimum number of iterations. We can look forward the for the improvement in training phase, initial parameter setting and handling floating point underflow in a better way so that use of HMM can be made more suitable for the classification of biological sequences. Parallel execution can be done on multi-processor machine to improve performance of forward, backward and parameter estimation process. We can work on optimizing expected number of hidden states to generate HMM model.

Acknowledgment

We would like to thanks Prof. M. P. Kurhekar for his guidance and kind support for completing this paper. Special Thanks to Prof. Pranay Meshram, Namrata Agrawal, Durgesh Sharma and Bhooshan Humane. Thanks to Our families for inspiring us for writing this paper.

References

[1] Palmondon R, Srihari S N. On-line and off-line handwriting recognition: A comprehensive survey [J].IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(1):63-84.

[2] Rath T M, Kane S, Lehman A. Indexing for a digital library of George Washington's manuscripts: A study of word matching techniques[R]. Massachusetts: Center for Intelligent Information Retrieval, Computer Science Department, University of Massachusetts, 2004

[3] Itay Bar Yosef, Klara Kedem, et al. Classification of Hebrew calligraphic handwriting style[A]. In: Proceedings of the 1st International Workshop on Document Image Analysis for Libraries, Palo Alto, California, 2006.299-305

[4] Shi Baile, Zhang Liang, Wang Yong, et al. Content-based Chinese script retrieval through visual similarity criteria[J]. Journal of Software, 2001, 12(9):1336-1342(in Chinese)

[5] Zhang Xiafen, Zhuang Yueting, et al.Chinese calligraphic chracter retrieval based on shape similarity[J]. Journal of Computer-Aid Design & Computer Graphics, 2005, 17(11):2565-2569)

[6] Wu Youshou, Ding Xiaoqing. Chinese Chracter Recognition: Theory, Approach and Implementation[M].Beijing: High Education Press, 1992(in Chinese)

[7] Zhang KuangZhong. Chinese Character Recognition Technology[M].Beijing: Tsinghua University Press, 1992(in Chinese)

[8] John Canny. A computational approach to edge detection[J]. In: IEEE transactions Pattern Analysis and Machine Intelligence, 1986, 11(2), 678-698.

[9] Blum H. A transformation for extracting new description of shape [A]. In Wathen-Dunn W ed. Model for the Perception of Speech and Visual[C].Cambiride, Massachusetts: MIT Press, 1967.362-380

[10] Zhao chunjinag, Shi wenkang. A robust algorithm for distilling the skeletons of images [J]. Computaer Appliacation, 2005, 6(1), 1305-1306(in Chinese)

[11] Wan Hualin, Morshed U, Hu Hong, etal. Texture feature and its application in CBIR [J]. Journal of Computer-Aid Design & Computer Graphics, 2003, 15(2):195-199(in Chinese) products of Bessel functions, " Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[12] Haili Chui, Anad, Rangrarajan. A new point matching algorithm for non-rigid registration [J]. Computer Vision and Image Understanding archive, 2003, 89(2):114-14

[13] L u Y, Zhang H J, Yin L W, et al. Joint semantic and feature based image retrieval using relevance feedback [J]. IEEE Transaction on Multimedia, 2003, 5 (3): 3392347.

[14] Inoue T, A be S. Fuzzy support vector machines for pattern classification[C]. In: Proceedings of International Joint Conference on N eural Networks (IJCNN '01), July 2001, 2: 1449-1454.
[15] A be S, Inoue T. Fuzzy support vector machines for multiclass problems [C]. In: Proceedings of the Tenth European Symposium on Artificial Neural Networks (ESANN " 2002), Bruges, Belgium, April 2002, 113-118.
[16] Wang Shang fei, Xue J ia, W ang Xifa. Contented-based emotion image retrievalmodel[J]. Computer Science, 2004, 31 (9): 186-190.
[17] Hsia T C. A note on invariantmoments in image processing [J].IEEE Trans. On SMC, 1981, 11 (12): 831-834.
[18] Vapnik V N. Statistical learning theory [M]. John W iley &Sons, NewYork, N Y, 1998.
[19] Abe S. Analysis of multiclass support vector machines[C]. In: Proceedings of International Conference on Computational Intelligence fo rModelling Control and Automation (C IMCA '2003), Vienna, Austria, February 2003, 385-396.
[20] Xu Yang.Chinese Calligraphy Production Method Based on HMM Genetics Analogy [J].Journal of Wuhan University, 2008, 54(1):85-89.
[21] Xu Song Hua, Lau Francis, Cheung William K, Pan Yun.He.Auto matic generation of artistic Chinese calligraphy.IEEE Intelligent Systems, 2005, 20(3):32-39
[22] Zhang Jun-song, Yu Jin-hui, Mao Guo-hong, Ye Xiu-zhi. Generating Brush Texture for Cursive Style Calligraphy with Auto regressive and Stratified Sampling[J].Journal of Computer-Aided Design & Computer Graphics, 2007, 19(11): 1399-1403.
[23] Dong Jun, Xu Miao, Pan Yun-he, Ye. Statistic Model-Based Simulation on Calligraphy Creation [J].Chinese Journal of Computers, 2008, 31(7): 7720-7725

Authors' Profiles

Shitalkumar R. Sukhdeve: Post Graduated in Computer Science & Engineering from the Visvesvaraya National Institute of Technology, Nagpur.

Prof. Manish P. Kurhekar: Assistant Professor in the Visvesvaraya National Institute of Technology, Nagpur branch CSE