An EM Estimation Algorithm of Iterated Random Function System Parameters from Noisy Observations

Reda A El-Khoribi
ralkhoribi@staff.cu.edu.eg
Information Technology Department
Faculty of Computers and Information
Cairo University
Giza, Egypt

Abstract- In this paper, a new solution to the inverse problem of iterated random function systems is presented. The solution is based on a generalized hidden Markov model formalism to model the process generated by an iterated random function system. Instead of the assumption of conditional independence of observation sequence elements given the state sequence, the new model assumes the existence of short term dependency between successive observations. An expectation-maximization (EM) algorithm is derived to estimate the parameters of the transformations used in the generation of the process. Regeneration of the process is possible given the model parameters and the initial observation by using the Viterbi algorithm. This suggests the use of the model in signal compression, encoding and detection. Increasing the order of the observation process opens the door for many pattern recognition and machine learning applications.

Keywords- Iterated random function-Fractals-EM-Algorithm-IRFHMM-HMM-Veterbi-Denoising.

I. Introduction

Iterated random functions [1, 2] have been used in representing a wide range of stochastic models. Examples include fractal image generation [1], data compression [1, 3, 4], data analysis [5], human perception modelling, simulation and problem solving [6, 7], queuing theory [2, 8], modelling of digital channels [9], simulation of Ising model [2], and financial time series [10]. The basic idea behind iterated random functions is that iterates of random Lipschitz functions converge if the functions are contracting on the average [1, 2, 11]. The random generation algorithm of the iterated function systems can be understood as a form of statistical state space model [12] that simulates many natural phenomena. The importance of statistical modelling and computation for state space models has attracted a great deal of application domains like speech recognition, signal processing, ion channels, molecular biology and economics. Developing estimation algorithms for fitting these models is highly needed in these real life applications. The behaviour of the maximum likelihood estimators of state space models have been widely studied in recent literature [11, 13, 14, 15]. The cases when the state space is finite and the observations are conditionally independent or conditionally Markovian dependent gain the most focus of these studies for their simplicity and adequateness in many real life applications. The implementation of a maximum likelihood estimator in switch auto-regression with Markov regimes was considered as an extension of the inference problem to time series analysis where the state space is finite and the observations are conditionally Markovian dependent [16,17,18]. This paper introduces an EM algorithm solution to the generalized HMM where, instead of the assumption of conditional independence of observation sequence elements given the state sequence, the model assumes the existence of short term dependency between successive observations. We will call the model ‘iterated random function hidden Markov model (IRFHMM)’. A description of the model is given in the next section.

II. The proposed model

The random iterated function system can be viewed as a discrete time evolving process in which the outcome at time t is computed from the outcome at time t-1 using a randomly selected function belonging to a set of N contractive transform maps \( \{ w_1, w_2, \ldots, w_N \} \); i.e.:

\[
o_t = w_{q_t}(o_{t-1})
\]

where \( q_t \in \{1,2,\ldots,N\} \) is a discrete random variable. The idea could be generalized by letting the current outcome be computed from the previous short history of order M; i.e.:

\[
o_t = w_{q_t}(o_{t-M}, o_{t-M+1}, \ldots, o_{t-1})
\]
This process could be viewed as a composition of two sup-processes:

1. The selection process in which \( q_t \) is selected. This is a hidden random process and could be assumed Markovian with initial state distribution:
   \[
   \pi_t = P(q_t = i), \ i = 1, \ldots, N
   \]
   And state transition distribution:
   \[
   a_{ij} = P(q_t = j|q_{t-1} = i)
   \]

2. The observation process in which the selected transform is applied to the previous observation history to get the new observation. Since our objective here is to predict the model parameters from noisy observations, we’ll assume the presence of additive noise \( n_t \) in the observation. Hence, the observation at time \( t \) is given by:
   \[
   o_t = w_{q_t}(o_{t-1}, o_{t-2}, \ldots, o_{t-M}) + n_t
   \] (3)

The noise variable \( n_t \) is assumed zero mean Gaussian for the sake of simplicity; i.e. \( n_t \sim N(0, \sigma^2) \). In this case, it could be easily shown, given \( q_t \) and the previous history \( o_{t-1}, o_{t-2}, \ldots, o_{t-M} \) that \( o_t \) is Gaussian with mean \( w_{q_t}(o_{t-1}, o_{t-2}, \ldots, o_{t-M}) \) and variance \( \sigma^2 \). Hence:

\[
 p(o_t|q_t, o_{t-1}, o_{t-2}, \ldots, o_{t-M}) = N(o_t; w_{q_t}(o_{t-1}, o_{t-2}, \ldots, o_{t-M}), \sigma^2)
\] (4)

Noise variance can also be made dependent on the current state in general; i.e.:

\[
 p(o_t|q_t, o_{t-1}, o_{t-2}, \ldots, o_{t-M}) = N(o_t; w_{q_t}(o_{t-1}, o_{t-2}, \ldots, o_{t-M}), \sigma_{q_t}^2)
\] (5)

Fig. 1 shows the dependency structure of the IRFHMM for \( M=1 \) and Fig. 2 shows the dependency when \( M=2 \).

![Figure 1: IRFHMM Dependency Structure with M=1](image1)

![Figure 2: IRFHMM Dependency Structure with M=2](image2)

The transform maps can take many forms. In this paper, we’ll consider the affine maps. For scalar observation sequence, this is given by:

\[
 w_t(o_t) = b_{i0} + \sum_{j=1}^{M} b_{ij} o_{t-j}
\] (6)

### III. Parameters of the IRFHMM with affine maps

Model parameters are \( \lambda = (\pi, A, B, \sigma) \) where \( \pi = (\pi_i), \ i = 1, \ldots, N \) is a vector representing the initial state distribution, \( A = (a_{ij}), \ i = 1, \ldots, N, j = 1, \ldots, N \) is the state transition matrix, \( B = (b_{ij}), i = 1, \ldots, N, j = 1, \ldots, M \) are the coefficients of the affine maps and \( \sigma = (\sigma_i^2), i = 1, \ldots, N \) is the variance vector.

The joint probability distribution of the observation sequence \( O = [o_t] \) and state sequence \( Q = [q_t] \) where \( t = 1, \ldots, T \) is as follows

\[
 p(O, Q) = p(q_1)p(o_1|q_1) \prod_{t=2}^{T} p(q_t|q_{t-1}) p(o_t|q_t, o_{t-1}, \ldots, o_{t-M})
\] (7)

The complete data log-likelihood is thus:

\[
 \log p(O, Q) = \log p(q_1) + \log p(o_1|q_1) + \sum_{t=2}^{T} \log p(q_t|q_{t-1}) + \sum_{t=2}^{T} \log p(o_t|q_t, o_{t-1}, o_{t-2}, \ldots, o_{t-M})
\] (8)

We assumed that \( o_t = 0 \) for \( t < 1 \) in the above equation. Moreover, the conditional distribution of the initial observation given the initial state is chosen to be:

\[
 p(o_1|q_1 = i) = N(o_1; b_{i0}, \sigma_{q_1}^2)
\] (9)

These assumptions allow considering that \( p(o_1|q_1) \) a special form of \( p(o_1|q_t, o_{t-1}, o_{t-2}, \ldots, o_{t-M}) \)
when $t=1$ and hence the complete data log-likelihood is given by:

$$
\log p(O, Q) = \log p(q_1) + \sum_{t=2}^{T} \log p(q_t | q_{t-1}) + \sum_{t=1}^{T} \log p(o_t | q_t, a_{t-1}, a_{t-2}, ..., a_{t-M})
$$

Substituting the model parameters for the probabilities in (10) gives:

$$
\log p(O, Q | \lambda) = \log \pi_i + \sum_{t=2}^{T} \log a_{q_{t-1}, i} - \frac{1}{2} \sum_{t=1}^{T} \log \sigma_{q_{t}}^2 \\
- \frac{1}{2} \sum_{t=1}^{T} \left( \frac{a_{t} - b_{q_0} - \sum_{k=1}^{W} b_{q_k} a_{t-k}}{\sigma_{q_{t}}^2} \right)^2 + \text{const}
$$

(11)

This is can be rewritten using the indicator function $1_x(i)$ which gives 1 when $x = i$ and 0 otherwise as follows:

$$
\log p(O, Q | \lambda) = \sum_{i=1}^{N} 1_{x(i)}(i) \log \pi_i \\
+ \sum_{t=2}^{T} \sum_{j=1}^{N} 1_{x(t-1)}(i) 1_{x(t)}(j) \log a_{ij} \\
- \frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{N} 1_{x(i)}(i) \left( \log \sigma_{q_{t}}^2 \\
+ \left( \frac{a_{t} - b_{q_0} - \sum_{k=1}^{M} b_{q_k} a_{t-k}}{\sigma_{q_{t}}^2} \right)^2 \right) + \text{const}
$$

(12)

IV. EM Algorithm

As in HMM, model parameters can be estimated iteratively using the EM algorithm [19, 20]. Starting with the initial guess of the parameters $\lambda = \lambda_0$, the algorithm iterates as given by:

$$
\lambda_{n+1} = \arg \max_{\lambda} E_Q(\log p(O, Q | \lambda) | O, \lambda_n)
$$

Here $E_Q$ denotes the conditional expectation over Q given O and $\lambda_n$: i.e.

$$
\lambda_{n+1} = \arg \max_{\lambda} \sum_{Q} \log p(O, Q | \lambda) \ p(Q | O, \lambda_n)
$$

This optimization is subject to the following constraints:

$$
\sum_{i=1}^{N} \pi_i = 1
$$

$$
\sum_{j=1}^{N} a_{ij} = 1, i = 1, ..., N
$$

Fortunately the three terms of the complete data log likelihood in eq. (12) contain different set of parameters. So, we can optimize them separately. The constrained optimization problem in the Lagrange multiplier method settings become:

$$
J(\lambda, \eta_0, \eta, \sigma, \beta) = J_1(\pi, \eta_0) + J_2(\lambda, \eta) + J_3(\beta, \sigma)
$$

(13)

Where:

$$
J_1(\pi, \eta_0) = \sum_{i=1}^{N} \gamma_{i}(i) \log \pi_i + \eta_0 \left( 1 - \sum_{i=1}^{N} \pi_i \right)
$$

$$
J_2(\lambda, \eta) = \sum_{t=2}^{T} \sum_{i=1}^{N} \xi_{t}(i, j) \log a_{ij} + \sum_{i=1}^{N} \eta_i \left( 1 - \sum_{j=1}^{N} a_{ij} \right)
$$

$$
J_3(\beta, \sigma) = -\frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_{t}(i) \left( \log \sigma_{q_{t}}^2 \\
+ \left( \frac{a_{t} - b_{q_0} - \sum_{k=1}^{M} b_{q_k} a_{t-k}}{\sigma_{q_{t}}^2} \right)^2 \right)
$$

The variables $\eta_0$ and $\eta = \{ \eta_1, \eta_2, ..., \eta_N \}$ are the Lagrange multipliers. In each step of the EM algorithm, the variables $\gamma_{t}(i)$ and $\xi_{t}(i, j)$ are computed using the previous estimate of the model parameters ($\lambda_n$) as:

$$
\gamma_{t}(i) = P(O, q_t = i | \lambda_n)
$$

$$
\xi_{t}(i, j) = P(O, q_t = i, q_{t+1} = j | \lambda_n)
$$

As in the HMM, variables $\gamma_{t}(i)$ and $\xi_{t}(i, j)$ are related to the forward and backward probability variables $\alpha_{t}(i)$ and $\beta_{t}(i)$ through the relations:

$$
\gamma_{t}(i) = \alpha_{t}(i) \beta_{t}(i)
$$

$$
\xi_{t}(i, j) = \alpha_{t}(i) a_{ij} \beta_{t+1}(j)
$$

$$
b_{j}(a_{t}) = P(o_{t} | q_{t}, a_{t-1}, a_{t-2}, ..., a_{t-M}) = N(o_{t}; w_{j} | o_{t-1}, a_{t-2}, ..., a_{t-M}, \sigma_{j}^2)
$$

$$
\alpha_{t}(i) = P(o_{t}, a_{2}, ..., a_{t}, q_{t} = i | \lambda)
$$

$$
\beta_{t}(i) = P(o_{t+1}, a_{t+2}, ..., o_{T} | q_{t} = i, \lambda)
$$
Re-estimation of model parameters

The solution of the constrained optimization of the EM auxiliary function is to get $\lambda_{n+1} = (\hat{\theta}, A, B, \sigma)$ as a solution to:

$$\nabla f(\lambda, \eta_{\lambda}, \eta_{A}) = 0$$

Re-Estimation of $\pi$

The new estimate of $\pi$ is obtained as a solution to the following system of equations:

$$\frac{\partial}{\partial \pi_i} J_1(\pi, \eta_{\pi}) = 0, \ i = 1, 2, \ldots, N$$

$$\frac{\partial}{\partial \eta_{\pi}} J_1(\pi, \eta_{\pi}) = 0$$

It is easy to verify that this gives:

$$\hat{\pi}_i = \frac{y_i(\theta)}{\sum_{i=1}^{M} y_i(\theta)} \quad (14)$$

Re-Estimation of $A$

Solving:

$$\frac{\partial}{\partial a_{ij}} J_2(A, \eta_{A}) = 0, \ i, j = 1, 2, \ldots, N$$

$$\frac{\partial}{\partial \eta_{A}} J_2(A, \eta_{A}) = 0, \ i = 1, 2, \ldots, N$$

It is easy to verify that:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} x_t(i,j)}{\sum_{t=1}^{T} \sum_{i=1}^{N} x_t(i,j)} \quad (15)$$

Re-estimation of $B$

This comes as a solution of the system of equations:

$$\frac{\partial}{\partial b_{ik}} J_3(B, \sigma) = 0, \ i = 1, 2, \ldots, N, k = 0, 1, \ldots, N$$

For $k=0$ we have:

$$\frac{\partial}{\partial b_{ik}} J_3(B, \sigma) = \frac{1}{\sigma_i^2} \sum_{t=1}^{T} y_t(i) \left( o_t - b_{ik} - \sum_{k=1}^{M} b_{ik} o_{t-k} \right)$$

And for $k>0$:

$$\frac{\partial}{\partial b_{ik}} J_3(B, \sigma) = \frac{1}{\sigma_i^2} \sum_{t=1}^{T} y_t(i) \left( o_t - b_{ik} - \sum_{j=1}^{M} b_{ij} o_{t-j} \right) o_{t-k}$$

This reduces to the following set of algebraic equations:

$$b_{i0} \sum_{t=1}^{T} y_t(i) + \sum_{j=1}^{M} b_{ij} \sum_{t=1}^{T} y_t(i) o_{t-j} = \sum_{t=1}^{T} y_t(i) o_t$$

$$b_{i0} \sum_{t=1}^{T} y_t(i) o_{t-k} + \sum_{j=1}^{M} b_{ij} \sum_{t=1}^{T} y_t(i) o_{t-k} = \sum_{t=1}^{T} y_t(i) o_{t-k}$$

The parameters $B = (b_{ik}, i = 1..N, k = 0..M)$ are thus obtained by solving the following linear algebraic system of equations (given in matrix form):

$$\begin{bmatrix}
\sum_{t=1}^{T} y_t(i) & \sum_{t=1}^{T} y_t(i) o_{t-1} & \cdots & \sum_{t=1}^{T} y_t(i) o_{t-M} \\
\sum_{t=1}^{T} y_t(i) o_{t-1} & \sum_{t=1}^{T} y_t(i) o_{t-2} & \cdots & \sum_{t=1}^{T} y_t(i) o_{t-M-1} \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{t=1}^{T} y_t(i) o_{t-M} & \sum_{t=1}^{T} y_t(i) o_{t-M-1} & \cdots & \sum_{t=1}^{T} y_t(i) o_{t-2M} \\
\end{bmatrix} \begin{bmatrix}
b_{i0} \\
b_{i1} \\
\vdots \\
b_{iM} \\
\end{bmatrix} = \begin{bmatrix}
\sum_{t=1}^{T} y_t(i) o_{t} \\
\sum_{t=1}^{T} y_t(i) o_{t-1} \\
\vdots \\
\sum_{t=1}^{T} y_t(i) o_{t-M} \\
\end{bmatrix} \quad (16)$$

Re-estimation of $\sigma$

By setting: $\frac{\partial}{\partial \sigma} J_4(B, \sigma) = 0$

Therefore:

$$- \frac{1}{2\sigma^2} \sum_{t=1}^{T} y_t(i) + \frac{1}{2\sigma^2} \sum_{t=1}^{T} y_t(i) \left( o_t - b_{i0} - \sum_{k=1}^{M} b_{ik} o_{t-k} \right)^2 = 0$$

Hence:

$$\sigma^2 = \frac{\sum_{t=1}^{T} y_t(i) \left( o_t - b_{i0} - \sum_{k=1}^{M} b_{ik} o_{t-k} \right)^2}{\sum_{t=1}^{T} y_t(i)} \quad (17)$$

V. Vector observation elements

A straightforward extension of the EM algorithm is easy when the observation sequence elements are vectors of independent components. There would be a coefficient matrix $B$ and a variance vector $\sigma$ for each component. The conditional probability of the vector elements is the multiplication of the conditional probabilities of its components and hence the EM optimization of the parameters of every component parameters ($B$ and $\sigma$) is done separately using equations (16) and (17).
VI. Multi-Observation IRFHMM

As in the classical HMM, the model can be trained using multiple observation sequences:

\[ O^{(1)}, O^{(2)}, ..., O^{(K)} \]

In this case the forward and backward variables \( a_t^{(k)}(i) \) and \( \beta_t^{(k)}(i) \) are computed from the individual sequences; i.e.:

\[
a_t^{(k)}(i) = P(o_1^{(k)}, o_2^{(k)}, ..., o_t^{(k)}, q_t = i|\lambda) \\
\beta_t^{(k)}(i) = P(o_t^{(k)}, o_{t+1}^{(k)}, ..., o_T^{(k)}|q_t = i, \lambda)
\]

The sum of the variables \( \gamma_t^{(k)}(i) \) and \( \xi_t^{(k)}(i,j) \), over \( k \) are used instead of \( \gamma_t(i) \) and \( \xi_t(i,j) \) in the re-estimation formulae:

\[
\gamma_t^{(k)}(i) = c_k a_t^{(k)}(i) \beta_t^{(k)}(i) \\
\xi_t^{(k)}(i,j) = c_k a_t^{(k)}(i)a_{ij}b_j(o_{t+1}) \beta_{t+1}^{(k)}(j)
\]

The weighting factor \( c_k \) represents the contribution of the observation sequence \( O^{(k)} \). In this research, we used \( c_k = P(O^{(k)}|\lambda) \) as a weighting factor.

VII. Pilot study

To test the new model, we generate a set of signals using iterated random function systems as shown in the following algorithm.

Algorithm Generate(\( \pi, A, W \))
1. Set \( r \) a random sample drawn from the initial distribution \( \pi \)
2. Repeat the following for \( t=1,2,\ldots,T \)
   - Compute: \( a_t = w_r(o_{t-1}, o_{t-2}, \ldots, o_{t-M}) \) (note \( a_t = 0 \) when \( t \leq 0 \))
   - Set \( r \) a random sample drawn from the transition distribution defined in the raw \( r \) of \( A \) (i.e. \( A_r \))
3. Return \( o = (o_1, o_2, \ldots, o_T) \)

The following subsections illustrate the use of the model in classification, de-noising, modulation and demodulation.

A. Classification Problem

The generated sets are divided into two classes each of 1000 samples coinciding with two different choices of the parameters (\( \pi, A, W \)). The data of each class is divided into 10% training and 90% testing samples for cross validation. Figure 3 shows two samples one from each class and Figure 4 shows the classifier performance with different signal to noise ratios where it is clear that the accuracy is reasonably high starting from SNR of 20dB.

B. Decoding Problem:

Given a signal generated by a known model \( \lambda \), one can use the Viterbi algorithm to estimate the most probable sequence of functions applied to the initial seed point to produce the signal. Next Figure shows the Viterbi algorithm adapted to work with the proposed model.

Algorithm Viterbi( \( o, \pi, A, W, \sigma \))
1. Initialization:
   \[
   \delta_1(i) = \pi_i N(o_1; w_i(o_{0-1}, o_{0-2}, \ldots, o_{0-M+1}), \sigma_i^2) \forall i = 1, \ldots, N
   \]
2. Iteration:
   \[
   \delta_t(j) = \max_{i} \delta_{t-1}(i)a_{ij}N(o_t; w_j(o_{t-1}, o_{t-2}, \ldots, o_{t-M}) \forall t = 1, \ldots, N, \forall \delta \]
   \[
   \psi_t(j) = \arg \max_{i} \delta_{t-1}(i)a_{ij} \forall j = 1, \ldots, N, \forall t = 2, \ldots, T
   \]
3. Backtracking:
\[ q_T^* = \arg \max_i \delta_T(i) \]
\[ q_t^* = \psi_{t+1}(q_{t+1}^*), t = T - 1, \ldots, 1 \]
4. Return \( q^* = (q_1^*, q_2^*, \ldots, q_T^*) \)
End Algorithm

This encourages the promotion of an adaptive denoising technique. Let’s consider a signal generated by an unknown model and then incorporated with noise (e.g. additive white Gaussian noise). One idea to filter the signal is to estimate the model using the noisy sample as a training sample then applying the Viterbi decoding algorithm to get the state sequence and filter the signal samples by iteratively applying the set of functions indexed by the state sequence. This is illustrated in the following algorithm:

Algorithm Denoise(o, N, M)
1. \([\pi, A, W, \sigma] = Train(o, N, M)\)
2. \( q^* = Viterbi(o, \pi, A, W, \sigma) \)
3. Iteration:
\[ y_t = w_q^t(o_{t-1}, o_{t-2}, \ldots, o_{t-M}) \forall t = 1, \ldots, T \]
4. Return \( y = (y_1, y_2, \ldots, y_T) \)
End Algorithm

Table 1 shows the average improvements in the SNR gained by the proposed denoising method. Figure 5 shows a sample signal before and after denoising. On performing the paired sample t-test on the two columns, we find that the p-value is 0.00019 meaning that there is a significant difference between the two data sets.

Table 1: Improvements of the SNR

<table>
<thead>
<tr>
<th>Input SNR (dB)</th>
<th>Output SNR (dB)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2.3053</td>
<td>0.5959</td>
<td>2.9012</td>
</tr>
<tr>
<td>13.7847</td>
<td>15.7736</td>
<td>1.9889</td>
</tr>
<tr>
<td>18.9524</td>
<td>23.1807</td>
<td>4.2283</td>
</tr>
<tr>
<td>22.1038</td>
<td>26.1015</td>
<td>3.9977</td>
</tr>
<tr>
<td>31.9133</td>
<td>37.378</td>
<td>5.4647</td>
</tr>
<tr>
<td>36.5028</td>
<td>41.9635</td>
<td>5.4607</td>
</tr>
<tr>
<td>46.3231</td>
<td>51.8325</td>
<td>5.5094</td>
</tr>
</tbody>
</table>

Another application of the Viterbi algorithm is the modulation/demodulation of digital signals over iterated random functions. Both the transmitter and receiver use the same set of contractive maps \( W = \{w_i, i \in \ldots, N\} \) where \( N \) denotes the number of distinct symbols that can be transmitted. The transmitter generates a modulated signal \( x \) corresponding to a symbol sequence \( q \) using the following algorithm:

Algorithm Modulate(q, W)
1. Iteration:
\[ x_t = w_{q_t}(x_{t-1}, x_{t-2}, \ldots, x_{t-M}) \forall t = 1, \ldots, T \]
2. Return \( x = (x_1, x_2, \ldots, x_T) \)
End Algorithm

The receiver, on the other hand, receives a noisy observation sequence \( o \) related to \( x \) through: \( o = x + n \) where \( n \) is a zero mean Gaussian noise. The resulting observation sequence, clearly, follows the IRF-HMM discussed before, with the parameters of the transforms \( W \) only known to the receiver while the rest of parameters \( \lambda_r = (\pi, A, \sigma) \) are not known. Fortunately, the maximum likelihood estimation of these parameters can be estimated iteratively with the same EM re-estimations derived before (equations 14, 15, 17); i.e. \( \hat{\lambda}_r = \arg \max_q p(o|\lambda_r, W) \)

The modulating transforms can take general forms and need not be affine. To estimate \( q \) based on \( o \) and \( W \), one can use the MAP estimator:
\[ \hat{q}_{\text{MAP}}(o) = \arg \max_q p(q|o, W) \]
This can be approximated to:
\[ \hat{q}_{\text{MAP}}(o) \approx \arg \max_q p(q|o, \hat{\lambda}_r, W) \]
The Viterbi algorithm can be used to obtain this approximate solution. The overall demodulation algorithm is represented in Figure 6.

Figure 6: Demodulation Algorithm

VIII. Conclusion

The paper introduces an EM based algorithm for learning the iterated random functions with affine maps. It is an extension of the continuous HMM successfully used in pattern recognition applications. The paper showed that it could be blindly filter noisy signals using the model with an average SNR gain of 5. dB. The reconstruction of the state sequence used in generation is possible given a common set of transforms with BER depending on the noise level.

REFERENCES

