gpdsHMM: A HIDDEN MARKOV MODEL TOOLBOX IN THE MATLAB ENVIRONMENT

Sébastien David, Miguel A. Ferrer, Carlos M. Travieso, Jesús B. Alonso
Dpto. De Señales y Comunicaciones, Universidad de Las Palmas de Gran Canaria
Campus de Tafira, 35017 Las Palmas de Gran Canaria
SPAIN
Tel: +34 928 451 269 Fax: +34 928 451 243, e-mail: gpds@gi.ulpgc.es

Abstract

A Hidden Markov Model (HMM) Toolbox within the Matlab environment is presented. In this toolbox, the conventional techniques for the continuous and discrete HMM are developed for the training as well as for the test phases. The ability to make different groups of components for the vector pattern is provided. Multi-labeling techniques for the discrete HMM is also provided. The toolbox includes procedures suitable for the classical applications based on the HMM, as pattern recognition, speech recognition and DNA sequence analysis.

Key words

Pattern recognition, Hidden Markov Model, Matlab Toolbox.

1. Introduction

A Hidden Markov Model (HMM) is a type of stochastic model appropriate for non stationary stochastic sequences, with statistical properties that undergo distinct random transitions among a set of different stationary processes. In other words, the HMM models a sequence of observations as a piecewise stationary process. Over the past years, Hidden Markov Models have been widely applied in several models like pattern [1, 2], pathologies [3] or speech recognition [4, 5], and DNA sequence analysis [6, 7]. The HMMs are suitable for the classification from one or two dimensional signals and can be used when the information is incomplete or uncertain.

To use a HMM, we need a training phase and a test phase. For the training stage, we usually work with the Baum-Welch algorithm to estimate the parameters (π, A, B) for the HMM [8, 9]. This method is based on the maximum likelihood criterion.

In addition to the Baum-Welch algorithm, it is necessary to estimate the Alfa and Beta matrices thanks to the forward and backward procedures. To compute the most probable state sequence, the Viterbi algorithm is the most suitable.

In order to apply the HMM techniques, the authors have developed a HMM toolbox called gpdsHMM in the Matlab environment. Several toolbox for the HMM already exist [10]. This work was carried out in order to offer a friendlier tool through didactics and graphics examples. This toolbox also contains two new concepts developed recently in the literature: the multi-labeling and the gathering methods which, when used in suitable conditions, improve significantly the results obtained with the HMM [11].

In section 2, we introduce the classical Hidden Markov Models. Section 3 is an introduction to the key points of the HMM toolbox, and the conclusions of this paper are presented in section 4.

2. Basic HMM

A HMM model is basically a stochastic finite state automaton, which generates an observation string, that is, the sequence of observation vectors, $\boldsymbol{O}=\boldsymbol{O}_1,\ldots,\boldsymbol{O}_t,\ldots,\boldsymbol{O}_T$. Thus, a HMM model consists of a number of N states $S=\{S_i\}$ and of the observation string produced as a result of emitting a vector $\boldsymbol{O}_t$ for each successive transitions from one state S_i to a state S_j. $\boldsymbol{O}_t$ is d-dimension and in the discrete case takes its values in a library of M symbols. The state transition probability distribution between state S_i to S_j is $A=\{a_{ij}\}$, and the observation probability distribution of emitting any vector $\boldsymbol{O}_t$ at state S_j is given by $B=\{b_j(\boldsymbol{O}_t)\}$. The probability distribution of initial state is $\Pi=\{\pi_i\}$.

$$a_{ij}= P(q_{k+1} = S_j \,|\, q_k = S_i) \quad (1)$$

$$b_j(\boldsymbol{O}_t)= P(\mathbf{O}_t \,|\, q_t = S_j) \quad (2)$$

$$\pi_i = P(q_0 = S_i) \quad (3)$$

Then, given a observation sequence $\boldsymbol{O}$, and a HMM model $\lambda=(A, B, \Pi)$, we can compute $P(\boldsymbol{O} \mid \lambda)$ the probability of the observed sequence by means of the forward-backward procedure [12]. Concisely, the forward variable is defined as the probability of the partial observation sequence $\boldsymbol{O}_1, \boldsymbol{O}_2,\ldots,\boldsymbol{O}_t$ (until time t) and state S_i at time t, with the model λ, as $\alpha_t(i)$. And the backward variable is defined as the probability of the partial observation sequence form $t+1$ to the end, given state S_i at time t and the model λ, as $\beta_t(i)$. The probability of the observation sequence is calculated as:

$$P(\mathbf{O} \,|\, \lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) = \sum_{i=1}^{N} \alpha_T(i) \quad (4)$$

and the probability of being in state S_i at time t, given the observation sequence $\boldsymbol{O}$, and the model λ, as:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(\mathbf{O}\,|\,\lambda)} \tag{5}$$

The ergodic or fully connected HMM is a HMM with all states linked all together (every state can be reached from any state). The left-right (also called Bakis) is a HMM with the matrix transition defined as:

$$a_{ij} = 0 \quad if \quad j < i \tag{6a}$$

$$a_{ij} = 0 \quad if \quad j > i + \Delta \tag{6b}$$

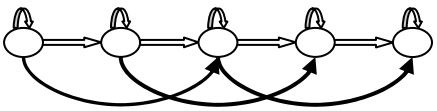

Figure 1. A bakis (or left-right) HMM

To calculate the HMM model for a given class, we need a lot of samples from this class. The characteristics for each sample are then extracted and stored in a parameter vector sequence $\boldsymbol{x}_t$. The $\boldsymbol{x}_t$ is mapped to the equivalent $\boldsymbol{O}_t$. In the continuous HMM (CHMM), the probability distribution functions are a mixture multivariate of Gaussians and so $\boldsymbol{x}_t$=$\boldsymbol{O}_t$. In the discrete HMM (DHMM), the parameters stored in $\boldsymbol{x}_t$ are quantified and assigned to a label (also called code word) $\mathbf{v}_k$, and so $\boldsymbol{O}_t$ is equal to the sequence of the index corresponding to $\mathbf{v}_k$ in the codebook. We adjust the model parameter λ=(A,B,Π) to maximize the probability of the observation sequence. Consequently, given W classes to recognize, we need to train λ^W for w=1...W HMM, one for each class, with the data set corresponding to the class w. We accomplish the above task thanks to the iterative Baum-Welch method, which is equivalent to the EM (expectation-modification) procedure.

The Baum-Welch method, developed in this toolbox, works as follows:

1. Estimate an initial HMM model as λ=(A,B,Π).
2. Given λ and the observation sequence $\boldsymbol{O}$, we calculate a new model $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\Pi})$ such as $P(\mathbf{O}\,|\,\bar{\lambda}) > P(\mathbf{O}\,|\,\lambda)$.
3. If the improvement $\frac{P(\mathbf{O}\,|\,\bar{\lambda}) - P(\mathbf{O}\,|\,\lambda)}{P(\mathbf{O}\,|\,\bar{\lambda})} < threshold$, then stop, otherwise put $\bar{\lambda}$ instead of λ and go to step 1.

In the discrete HMM case, a vector quantizer (VQ) is required to map each continuous observation vector into a discrete codebook index. The formulas of Baum-Welch method used in this toolbox to estimate the model λ=(A,B, Π) (step 2) are the following [12]:

$$\bar{\pi}_i = \gamma_1(i) \tag{7}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{8}$$

$$\bar{b}_j(\mathbf{v}_k) = \frac{\sum_{t=1}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \quad s.t. \; \mathbf{O}_t = \mathbf{v}_k \tag{9}$$

$$\text{with } \xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(\mathbf{O}_{t+1})\, \beta_{t+1}(i)}{P(\mathbf{O}\,|\,\lambda)} \tag{10}$$

Thus, it is also important to compute the forward and backward procedures. In the CHMM, the Baum-Welch algorithm estimates the means and variances for the mixture of Gaussians [12].

$$\bar{c}_{jk} = \frac{\sum_{t=1}^{T-1} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)} \tag{11a}$$

$$\bar{\mu}_{jk} = \frac{\sum_{t=1}^{T-1} \gamma_t(j,k) \bullet \mathbf{O}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{11b}$$

$$\bar{U}_{jk} = \frac{\sum_{t=1}^{T-1} \gamma_t(j,k) \bullet \left(\mathbf{O}_t - \mu_{jk}\right)\left(\mathbf{O}_t - \mu_{jk}\right)'}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{N} \alpha_t(j)\beta_t(j)}\right]\left[\frac{c_{jk}\, \aleph(\mathbf{O}_t, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm})}{\sum_{m=1}^{M} c_{jm}^{r}\, \aleph(\mathbf{O}_t, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm})}\right] \tag{11c}$$

The Viterbi algorithm can be used to obtain the estimation of the most probable state sequence. Once all the HMMs Λ=(λ^1 ...λ^W) are correctly trained, to classify a sequence for the observation $\boldsymbol{O}$, P_w=$P(O|\lambda^W)$ is calculated for all the λ^W. The unknown observation $\boldsymbol{O}$ is then classified by the process:

$$w^* = \arg\max_{1 \le w \le W} p^w \tag{12}$$

And so, w^* is the optimum class for the observation $\boldsymbol{O}$.

The initialization and stop criteria must be chosen adequately for the HMM. It directly interacts on the relevancy of the HMM [13]. Equiprobable and equal occupancy methods for the initial models are provided as well as iteration and rate of the error for the stop criterion.

3. gpdsHMM toolbox functionality

In this first version of the gpdsHMM, our goal is to propose a toolbox which contains the most usual functions for the HMM development. Both discrete HMM and continuous HMM were developed. We propose, in particular, a function to migrate from a DHMM to a CHMM and so to predict the initials parameters for the mixture of Gaussians. The generals Bakis (or left-right) and ergodic models are proposed. A multi-labeling method (for the DHMM) is also provided. The authors wished to develop some classical and useful examples in order to make this tool as didactic as possible. The functions for the CHMM use the netlab utility available from:

http://www.ncrg.aston.ac.uk/netlab/
A freely distributed version of the gpdsHMM toolbox is available from:
http://www.gpds.ulpgc.es/download/index.htm
This toolbox is distributed as binary (dll files) and source code format. For a wide promotion, we ask the users to make a reference to this paper. For any remarks about this toolbox, do not hesitate to contact the authors sending an e-mail to: gpds@gi.ulpgc.es

In addition to the classical techniques for the HMM, the authors have developed the multi-labeling and the gathering methods and proposed several functions to facilitate the use of the HMM. For further information about those functions, please refer to the appendix.

Multi-labeling

In the discrete Hidden Markov Model approach, the conventional VQ technique is applied. In our toolbox, the library of labels (also called codebook) is calculated from the training database thanks to the k-mean or the LBG algorithms. In our toolbox, for each incoming vector the quantifier performs a hard decision about which of its code word is the best match, and so the information about how the incoming vector matches other code words is discarded. Because of the discrete quantification variability, the vector of parameters can be displaced in such a way that this displacement is a potential source of misrecognition.

Unlike the conventional VQ, multi-labeling makes a soft decision about which code word is the closest to the input vector, generating an output vector whose components indicate the relative closeness of each code word to the input [11].

So, the multi-labeling codebook used in this toolbox maps the input vector $\boldsymbol{x}_t$ into an observable vector $\boldsymbol{O}_t=\{w(\boldsymbol{x}_t,\boldsymbol{v}_k)\}_{k=1,\ldots,M}$, whose components are calculated with:

$$w(\mathbf{x}_t,\mathbf{v}_k)=\frac{1/d(\mathbf{x}_t,\mathbf{v}_k)}{\sum_{m=1}^{M}1/d(\mathbf{x}_t,\mathbf{v}_m)} \qquad (13)$$

These components are positive, their sum is 1. Thus, this toolbox provides a lineal combination of code words $\boldsymbol{v}_k$ in order to improve the representations for the $\boldsymbol{x}_t$. Under the standard DHMM approach, $w(\boldsymbol{xt},\boldsymbol{v}_k)$ would take value 1 for the code word with the best match and value 0 for the rest. For the multi-labeling, the toolbox enables to use the C closest labels with C<M (number of symbols) weighted with the $w(\boldsymbol{x}_t,\boldsymbol{v}_k)$ calculated in (13) and re-scaled in order to have the sum equal to 1. The probability of an observable vector $b_j(\boldsymbol{O}_t)$ is given by:

$$b_j(\mathbf{O}_t)=\sum_{k=1}^{C}w(\mathbf{x}_t,\mathbf{v}_k)b_j(\mathbf{v}_k)$$

With respect to Baum-Welch estimation formulas for the transition probabilities a_{ij} (formula 8) and the initial state probabilities π_i (formula 7) are generalized in the same manner. Regarding the estimation of $b_j(\boldsymbol{v}_k)$, the better recognition scores were obtained just using the next heuristic estimation formula:

$$\overline{b}_j(\mathbf{v}_k)=\frac{\sum_{t=1}^{T}\gamma_t(i)\,w(\mathbf{x}_t,\mathbf{v}_k)}{\sum_{t=1}^{T}\gamma_t(i)} \qquad (14)$$

Although the above equation does not guarantee the convergence of the training process, in practice its use decrease the required number of iterations. Furthermore, to work out the observable vector $\boldsymbol{O}_t$, multi-labeling can be simplified using, only the C most significant values of $w(\boldsymbol{x}_t,\boldsymbol{v}_k)$ for each $\boldsymbol{x}_t$ (C labels), where C is lower than the codebook size M. The corresponding improvement for the recognition rate makes the multi-labeling Hidden Markov Model (MLHMM) approach extremely efficient. The MLHMM approach is closely related to the semi continuous approach.

Gathering

The gathering is the fact to group together different types of characteristics stored in a vector pattern. Instead of gathering all the parameters in one HMM, the gathering builds one HMM by group of parameters. The input $\boldsymbol{x}_t$ of d-dimension is now described as a group of vector of R characteristics with the sum of the dimension for each characteristic d(r) equal to d.
The following variables described as "y^r" stand for the variable "y" of the characteristic "r".
In the continuous case, where $\boldsymbol{O}_t=\boldsymbol{x}_t$, $b_j(\boldsymbol{O}_t)$ is calculated as:

$$b_j(\mathbf{O}_t)=\prod_{r=1}^{R}b_j(\mathbf{O}_t^r) \qquad (15a)$$

$$b_j(\mathbf{O}_t^r)=\sum_{m=1}^{M}c_{jm}^r\,\aleph(\mathbf{O}_t^r,\boldsymbol{\mu}_{jm}^r,\mathbf{U}_{jm}^r) \qquad (15b)$$

In the discrete case, we calculate a number of R vector quantifiers with code vectors $\{\mathbf{v}_k^r\}_{k=1,2,\ldots,M}^{r=1,2,\ldots,R}$, and quantify each parameter x_t^r with its vector quantifier as follows:

$$\mathbf{O}_t^r=\mathbf{v}_k^r \text{ iff } d(\mathbf{x}_t^r,\mathbf{v}_k^r)<d(\mathbf{x}_t^r,\mathbf{v}_m^r) \text{ for all } m\neq k$$

$$b_j(\mathbf{O}_t)=\prod_{r=1}^{R}b_j(\mathbf{O}_t^r) \qquad (16)$$

$$\text{with:}\quad b_j(\mathbf{O}_t^r)=b_j(\mathbf{v}_k^r) \qquad (17)$$

And the new estimation formula is:

$$\overline{b}_j(\mathbf{v}_k^r)=\frac{\sum_{t=1}^{T}\gamma_t(i)}{\sum_{t=1}^{T}\gamma_t(i)}\Bigg|_{s.t.\,\mathbf{O}_t^r=\mathbf{v}_k^r} \qquad (18)$$

Finally, in the discrete case, it is possible to combine the gathering with the multi-labeling methods.

To evaluate the efficiency of the HMM, a function calculates, thanks to the results of the HMM, a matrix of confusion for each set of parameters. The matrix of confusion is a matrix built during the test phase. It shows how and where the HMM fails. Thus, the

recognition rate and the matrix of confusion give a good idea about the pertinence for the given set of parameters within the recognition task.

To have a more didactic approach of the HMM, various functions and examples have been developed. Those functions cover manipulating sequences, probabilities matrices (like the Alfa, Beta or Gamma matrices) and maximum likelihood.

4. Conclusions

In this paper, a brief description of the gpdsHMM toolbox for Matlab is presented. This HMM toolbox is designed in order to provide the multi-labeling and gathering functionality in a didactic tool. The authors would like to exchange, through this toolbox, their experience in this area. The toolbox is structured so that everyone can customize a new HMM and extend it to fit a wide field of applications or can use the examples as a base.

5. Acknowledgement

We acknowledge the financial support form the Spanish government (TIC2003-08956-C02-02).

6. References

[1] J. A. Sánchez, C. M. Travieso, I. G. Alonso, M. A. Ferrer, *Handwritten recognizer by its envelope and strokes layout using HMM's*, 35rd *Annual 2001 IEEE Internacional Carnahan Conference on Security Technology, (IEEE ICCST'01)*, London, UK, 2001, 267-271.

[2] M. A. Ferrer, J. L. Camino, C. M. Travieso, C. Morales, *Signature Classification by Hidden Markov Model*, *33rd Anual 1999 IEEE Internacional Carnahan Conference on Security Technology, (IEEE ICCST'99), Comisaría General de Policía Científica, Ministerio del Interior, IEEE Spain Section, COIT, SSR-UPM, Seguritas Seguridad España S.A*, Madrid, Spain, Oct. 1999, 481-484.

[3] J. B. Alonso, C.Carmona, J. de León y M. A. Ferrer, Combining Neural Networks and Hidden Markov Models for Automatic Detection of Pathologies, *16_th Biennial International Eurasip Conference Biosignal 2002*, Brno, Check Republic, June 2002.

[4] Renals, S., Morgan, N., Bourlard, H., Cohen, M. & Franco, H. (1994), Connectionist probability estimators in HMM speech recognition, *IEEE Transactions on Speech and Audio Processing 2(1)*, 1994, 161-174.

[5] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, Maximum mutual information estimation of HMM parameters for speech recognition,. *In Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing,* , Tokyo, Japan, December 1986, 49-52

[6] Yin, M.M., Wang, J.T.L., Application of hidden Markov models to gene prediction in DNA, *Information Intelligence and Systems, 1999.*] *Proceedings. International Conference on*, 1999, 40 – 47.

[7] Cohen, A., Hidden Markov models in biomedical signal processing, *Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conf. of the IEEE, Vol. 3*, 1998, 1145 – 1150.

[8] L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probablistic functions of Markov chains. *The Annals of Mathematical* Statistics*, 41(1),* 1970, 164-171.

[9] L. Baum, An inequality and associated maximization technique in statistical estimation for probalistic functions of Markov processes. *Inequalities*, *3*, 1972, 1-8.

[10] Al-Ani, T.; Hamam, Y., An integrated environment for hidden Markov models, a Scilab toolbox, *Computer-Aided Control System Design, 1996., Proceedings of the 1996 IEEE International Symposium on*, Sept. 1996, 446 – 451.

[11] J.Hernando, C.Nadeu, José B. Mariño, Speech recognition in a noisy environment based on LP of the one-sided autocorrelation sequence and robust similarity measuring techniques, *Speech communications*, *vol. 21*, 1997, 17-31.

[12] L. R. Rabiner. Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition *Readings in Speech Recognition, chapter A*, 1989, 267-295.

[13] M.A. Ferrer, I. Alonso, C. Travieso, Influence of initialization and Stop Criteria on HMM based recognizers, Electronics letters of IEE, *Vol. 36*, June 2000, 1165-1166.

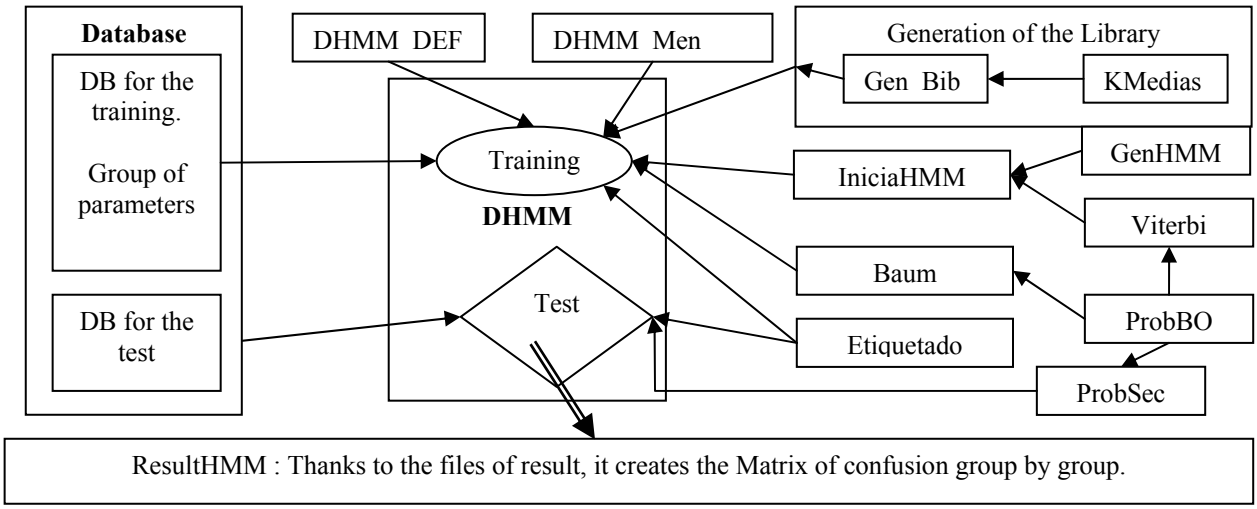

Figure 2: Block-diagram for a DHMM

Appendix:

The main change between the block-diagram for the DHMM (figure 2) and the CHMM, is the function "etiquetado" that is the quantification for the vector of characteristic. Those functions are more detailed on the web.

Alfa: this function calculates the Alfa from the HMM defined with the matrix A, B and Pi. The Alfa is scaled to avoid the problem of the precision.
AlfaBeta: this function calculates the Alfa and Beta from the HMM defined with the matrix A, B and Pi. The Alfa is scaled to avoid the problem of the precision.
Baum: this function computes the Baum Welch algorithm in order to estimate the HMM parameters, (formulas 7, 8 and 9).
DHMM_DEF: this script defines the parameters of the discrete HMM. The type of the HMM and the method to quantify the library are chosen.
DHMM: this is a script to train and test the HMM.
DHMM_MEN: this script is a small program that prints some messages to describe the computation of the HMM.
Etiquetado: this function implements the multi-labeling for the discrete HMM. To do that, we use a clustering algorithm like the k-mean to quantify the library VQ. The multi-labeling enables to give various labels given a parameter. The possible labels are directly linked to the number of symbol by state, (formulas 16, 17, and 18).
formato_lectura_secuencial: this function splits the database to smaller files (located in the c:\temphmm directory) in order to decrease the HMM computational time and the memory requirement.
genHmm: this function generates a HMM with N states and M symbols by state.
gen_bib: this function computes a library for all the different models of parameters based on the training vectors. According to the set up made in the DHMM_DEF, the library is generated with a LBG algorithm or with a k-means algorithm.
iniciaHMM: this function calculates the HMM optimal initial model for the Baum-Welch training.
kMedias: this function computes a k-mean algorithm for the library VQ.
prodBO: this function calculates the product of the probability to monitor a sequence $\boldsymbol{O}$ with the probability of the backward B distribution.
ProbSec: this function estimates the log of the probability to monitor a sequence $\boldsymbol{O}$ given a HMM. We use the log to avoid numerical problem, formulas (4).
Resulhmm: this function calculates the confusion matrix group by group according to the gathering made in the DHMM_DEF.
ROC: this function analyses the results from the HMM. It particularly deals with the problem of false acceptance rate (FAR called here FMR false match rate) and false rejection rate (FRR called here FNMR false non match rate).
Viterbi: this function calculates the sequence of the most probable states given the HMM and the sequence monitored $\boldsymbol{O}$. We use the algorithm of Viterbi with the log for the numerical precision problems.

To implement the continuous HMM, this toolbox uses the functions from the netlab utility, as mentioned before (http://www.ncrg.aston.ac.uk/netlab/), and some of the functions described for the discrete HMM. Moreover, the following functions were designed to manage a CHMM.

AlfaBetac: this function is equivalent to the AlfaBeta function used for the DHMM.
Baumc: it is equal to the function Baum but used for a CHMM instead of a DHMM, (formulas 11a, 11b, 11c).
CHMM_DEF: this function defines the parameters for the CHMM and in particular the mixture of Gaussians used in this case.
CHMM: this is a script to train and test the HMM.
CHMM_MEN: it is equal than the DHMM_MEN.
DHMM2CHMM: this function generates a CHMM from a DHMM.
GencHMM: it is equivalent to the genHMM but used for a CHMM.
IniciacHMM: it is equivalent to the genHMM but used for a CHMM.
Probsimb: this function calculates the probability for a vector $\boldsymbol{O}$ to be generated from any state S_i.
Viterbic: Function to calculate the most probable state sequence given a CHMM and a sequence monitored $\boldsymbol{O}$.