

HMMSim: An Interactive Simulation Tool for Teaching Discrete Hidden Markov Models

J. F. Vélez, A. Sánchez and A. B. Moreno

Abstract—This paper presents the educational software tool *HMMSim* (Discrete Hidden Markov Model Simulator) for the definition, training and simulation of Discrete Hidden Markov Models (HMM). This software allows to create a HMM for a given pattern recognition problem, to train the HMM adjusting its parameters according to the training process, and to simulate the behaviour of an implemented HMM. The tool also automatically generates the equivalent C++ program corresponding to a trained HMM, which enables to use this code as a part of a program having the functionality of the trained HMM for a particular recognition task. The first motivation for the development of this system has been to create a visual software for helping the students to understand how a HMM works and how it can be used in pattern recognition problems. *HMMSim*, which is inspired in the SNNS environment for neural networks, has been successfully used in our Digital Image Processing and Pattern Recognition course of a Computer Science technical degree at the Universidad Rey Juan Carlos. This work describes the main components of *HMMSim*, summarizes its functionality, and shows an educational application of this tool to a speech recognition course project.

Index Terms— Interactive Learning Environment, Simulation Software, Graphical User Interface, Hidden Markov Models, Pattern Recognition, Object Oriented Programming.

I. INTRODUCTION

Pattern Recognition is usually taught as a course or as a part of a more general course (for example Artificial Intelligence or Computer Vision) in many Computer Science degrees. Classical pattern recognition techniques such as the Euclidean classifier, Bayes classifier or clustering algorithms, are usually explained in these courses. These topics are presented in an introductory elective subject on *Digital Image Processing and Pattern Recognition* in the 3rd year of a Computer Science technical degree at Universidad Rey Juan Carlos. Other more complex trainable classifiers such as Neural Networks (NN) or Discrete Hidden Markov Models (HMM) are also explained in this course.

Due to the inherent complexity of NN and HMM systems, the use of appropriate computer-based simulation tools for

teaching purposes is desirable and even necessary. This is specially important when assignments or computer projects using these classifiers are proposed as exercises to the students. The course needs from visual interactive simulation tools which assist the students to understand how these complex classifiers work, and also how can they be applied to practical recognition problems.

A GNU visual software environment for the definition and simulation of neural networks is SNNS (Stuttgart Neural Network Simulator) [1] developed at the University of Stuttgart since 1989. This tool provides an efficient and flexible simulation for many different types of neural nets (i.e. perceptrons, self-organizing maps or recurrent networks). SNNS consists of two main components: the simulator kernel, written in C, which operates on the internal data structures of neural networks and performs all operations on them; and the graphical user interface, which gives a graphical representation of the networks and controls the kernel during the simulation run. Our students also worked with SNNS v.4.2 in the course for the simulation of all neural networks examples. Most of the proposed problems require the design and implementation of multilayer perceptrons.

Similarly, a free visual software tool for the definition, training and simulation of HMM in related pattern recognition problems was also needed in our course. Resemblance of the chosen HMM software with SNNS is considered important for pedagogical purposes due to the wide diffusion of SNNS, its simplicity of use and other interesting capabilities, and finally because the students' effort to learn a HMM simulator is greatly reduced in this way.

We have knowledge of other two software tools for the representation and simulation of HMM. HMMPro [2] is a complete but complex environment, developed by Net-ID, and specifically oriented for protein and DNA sequences analysis. HTK [3], developed at Cambridge University, is a general set of library modules written in C for building and manipulating HMM. However, neither visual interfaces nor other desirable features (such as the equivalent high-level code corresponding to a trained HMM) are available for this last tool. In conclusion, we have not found any appropriate HMM simulation educational software for our course. This is the main motivation, as well as the objective of adding to a HMM environment the most salient features of SNNS, which stimulated us for developing our own educational HMM simulation software. The implemented tool, called *HMMSim* (Discrete Hidden Markov Model Simulator), is GNU

The authors are with the Department of Ciencias Experimentales e Ingeniería, Universidad Rey Juan Carlos, Campus de Móstoles, c/ Tulipán s/n, 28933 Móstoles (Madrid), Spain (A. Sánchez is the corresponding author, phone: +34-91-6647452; fax: +34-91-6647490; e-mail: an.sanchez@escet.urjc.es).

Licensed and available at: <http://www.escet.urjc.es/~visionc>.

The rest of the paper is organized as follows. Section II outlines some applications of HMMs, and gives an overview of *HMMSim* (i.e. its main capabilities and user interfaces). An illustrative educational example showing the use of *HMMSim* in a course project proposed to our students appears in Section III. Finally, Section IV resumes the conclusions and future extensions of *HMMSim*.

II. *HMMSim* OVERVIEW.

As it was pointed out, the motivation behind the development of this software system is to make available a simple and visually intuitive environment which enables to build, train and simulate a discrete HMM for its application in pattern recognition problems. This tool can assist the students to achieve an understanding of these trainable pattern classification models, and to use them in related exercises introduced during the course. The design and implementation of *HMMSim* was carried out using Object Oriented Programming principles, and the tool was inspired in the SNNS software. Due to SNNS is also used in our course laboratory assignments, it is easier for the students to learn the main features of *HMMSim*. Thus, they can concentrate more into the application of HMM to the specific pattern recognition problem.

Pattern Recognition is a scientific area with the aim to classify objects (or patterns) into a number of categories or classes [4]. These objects are usually represented as feature vectors with a fixed length, and each feature contained in the vector is generally independent from the others. However, some pattern recognition problems need from variable length feature vectors. One of the most extended models to represent and solve these pattern recognition problems are the Discrete HMM [5]. Some important applications of the HMM are: speech recognition [6][7], handwriting recognition [8], face identification [9], or signature classification [10]. In general, all of those applications can be modeled and solved using the HMM approach, when they require the recognition of variable length sequences in which the components are temporally related. For a formal definition of HMM and the parameters involved in this model see for example [5]. Next, we resume the main components of *HMMSim*.

A. *HMMSim* features and user interfaces.

As mentioned above, the design of *HMMSim* user interfaces is intentionally inspired in the neural simulator SNNS [1]. The *HMMSim* capabilities integrated in the proposed software tool are:

- loading and saving a HMM file,
- loading pattern files to train a HMM using the specified parameters,
- displaying a given HMM,
- adjusting the patterns to a HMM according to the training process,

- getting help about the available operations defined for *HMMSim*,
- generating an equivalent C++ source code file which represents the functionality of a trained HMM,
- estimating the probability to happen a particular sequence of observations (known as *HMM Problem 1*), and
- finding the parameters of the HMM which maximize the probability to happen certain sequences of observations (known as *HMM problem 3*). In this way, a HMM to recognize a particular set of sequences of observations can be determined.

All the previous operations are initiated from the *HMMSim* Manager Panel, shown in Fig. 1. Other graphical menus are accessible from this window, for example the training interface (Train HMM), or the menu for displaying a defined HMM (Display HMM).



Fig. 1. Manager panel showing the functionality of *HMMSim* (loading a HMM, opening a training window, loading training patterns, displaying a HMM, saving a HMM, generating C++ code for a trained HMM and receiving help about the tool).

- HMM display window.

Fig. 2 shows the display or visualization window for a defined HMM. This interface window is designed to represent from left to right a sequential HMM, which are mostly used in pattern recognition applications. The graph which shows a particular HMM is changing during the training process as this stage modifies the considered HMM. Probabilities of transitions among different states of the graph are represented according to a color code.

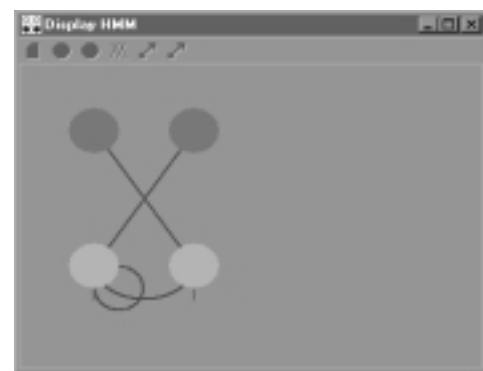


Fig. 2. Display window. Dark nodes represent observable states, and light ones represent hidden states of a HMM. The arc colors indicate the probability of a path representing a sequence of observations.

- Training window.

Training evolution of a given HMM can be observed in this window. Probability of a HMM to generate the training sequences is displayed at each simulation time unit. A training example is shown in Fig. 3. Another functionalities offered by this interface window are to select the pattern files used for training or testing the considered HMM, to stop the training process, and to check how a trained model behaves for a new pattern.

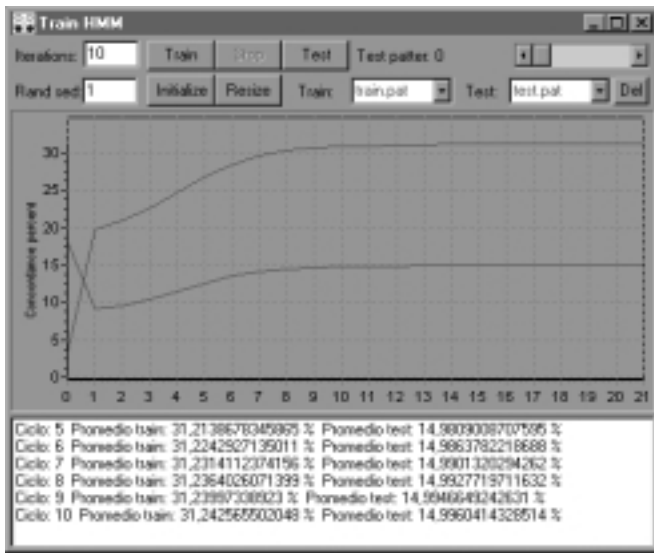


Fig. 3. Training window. Buttons to start and stop the training process, and to select train and test sets for a HMM are represented on the top of this window. Displayed graphics on the middle and text area on the bottom of this window permit us to observe the training evolution. Note that probability values of the HMM change during the training task, and training and test curves are dynamically generated.

- HMMSim auxiliary files.

HMMSim offers the options of storing into a file and loading from a file the configuration of a HMM and the training patterns for it, respectively. Fig. 4 shows the aspect of two respective ASCII files containing the configuration of a HMM and the training patterns for a considered example.

<pre>#HIDDEN STATES 4 #OBSERVABLE STATES 5 #A_Matrix 0.7 0.1 0.1 0.1 0.7 0.1 0.1 0.1 0.7 0.1 0.1 0.1 0.7 0.1 0.1 0.1 #B_Matrix 0.6 0.1 0.1 0.1 0.1 0.6 0.1 0.1 0.1 0.1 0.6 0.1 0.1 0.1 0.1 0.6 0.1 0.1 0.1 0.1 #Pi_Matrix 0.7 0.1 0.1 0.1</pre>	<pre>0 1 0 1 2 1 0 1 3 3 2 1 3 2 3 3 2 1 0 3 1 0 3 4 1 0 2 1 0 2 2 1 0 1 2 1 0 0 2 4 0 2 0 2 4 0 2 3 3 4 1 2 1 0 1 2 3 3 3 2 1 3 3 3 2 2 2 4 0 3 4 2 0 0 0 0 2 2 2 1 0 0 0 1 3 1 3 3 1 2 3 4 1 2 1 0 2 0 3 1 2 4 0 0 1 0 1 3 1 3 4 1 3 3 2 1 0 2 1 0 0 1 0 2 4 0 2 4 2 1 0 2 4 0 2 4 2 1 3 3 3 1 0 3 1 2 1 2 1 2 2 4 0 0 2 3 1 2 2 0 2 4 1 2 0 1 3 4</pre>
---	---

Fig. 4. Left figure shows the text file used to store the configuration of a given HMM. Right figure shows the corresponding sample pattern file used for a HMM training.

B. HMMSim components.

HMMSim has been designed and implemented according to Object Oriented Programming. Fig. 5 shows the relation among the different HMMSim components. The separation of these components makes possible to generate HMMs which are available to be used once they are trained for a particular task. This feature makes the tool very useful for pattern recognition software development. Automatically generated C++ code for a trained HMM, is ready to be used in a particular pattern recognition task.

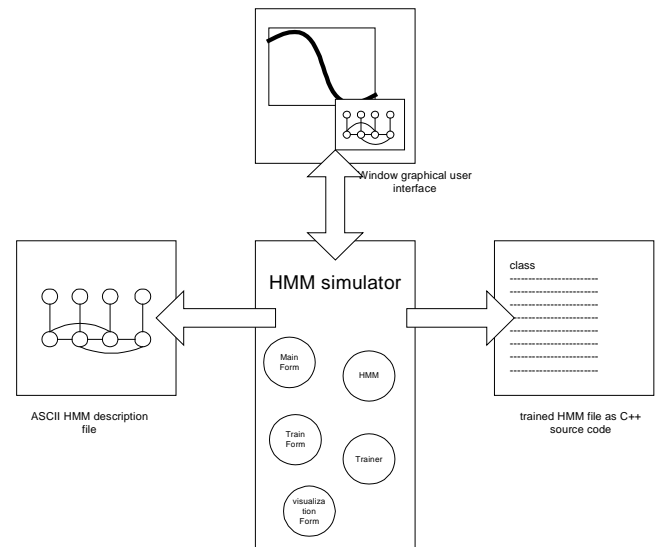


Fig. 5. HMMSim components: simulator, trainer, graphical user interface and model compiler *hmm2c++*.

The internal structure of the tool could be described by its UML Static Structure Diagram [11] in Fig. 6. The main classes are: *HMM* (represents the description of a particular HMM), *TrainingForm* (shows the training evolution form), *HMMTrainer* (trains a given HMM according to Baum-Welch algorithm [7]), *DisplayForm* (shows the visualization form of a HMM), and *MainForm* (shows the main form of HMMSim which respectively enables to load, store, train, display or produce C++ code for a given HMM).

III. EDUCATIONAL APPLICATION EXAMPLE OF HMMSim: SPEECH RECOGNITION PROJECT.

To illustrate the course application of HMMSim in a practical pattern recognition example, we shortly describe a course project proposed to the students this year. The project consists in a simple speech recognition problem. The solution must be developed using HMMSim.

The goal of this project is the implementation of a system to recognize isolated words of a small vocabulary (i.e. about tens of words). This is the simplest form of speech recognition to be performed because the end points of the words are easier to find and the pronunciation of each word does not affect the others. Continuous speech is much more difficult to handle [7].

The first step in the resolution of this problem is to build a feature vector for each speech wave corresponding to a word.

This previously requires to capture the particular word in the form of a WAV file. Students speak the required words to a microphone in front of their mouth in a natural manner. Fig. 7 represents a sample sound wave obtained from the pronunciation of the Spanish word “uno”.

Next, the word wave is decomposed into parts to locally study the dominant frequencies. This task is performed by using, for example, the *Fast Forier Transform* (FFT), the *Linear Predictive Coding Model* or any other similar model with the help of filters to eliminate high frequencies [7]. After the application of these transformations on the wave file, a sequence of N -dimensional vectors is obtained. The same procedure is applied for different voices of the same word. Then, the k -means clustering algorithm is applied to obtain the k most representative centroids of all vectors (in our example the value of k is 5). In this way, a method to reduce a voice file to a sequence of N integer values in the range 1..5 (for $k=5$) is available.

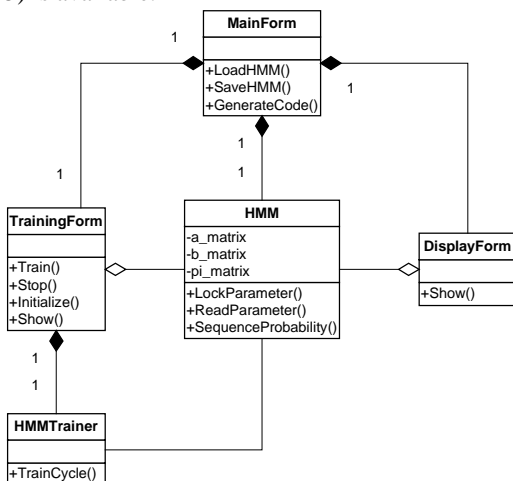


Fig. 6. UML Static Structure Diagram of *HMMSim*.

At this point, students define a HMM to recognize each particular word. This model has been created and trained with our tool *HMMSim*. For this task, they need to capture a set of wave files corresponding to different repetitions of the same word. With these files, students define and train a HMM using *HMMSim*, which enables to recognize the particular word. The same procedure is repeated for all the words belonging to the vocabulary to be recognized and a particular HMM is built for each different word. The recognition task consists in determining the pattern word (or particular trained HMM) of the considered vocabulary which corresponds to a test word (if it happens).

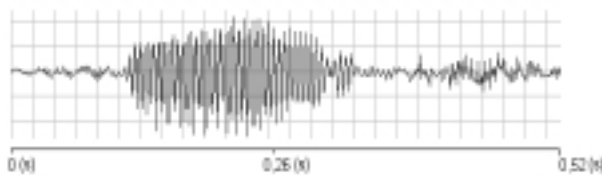


Fig. 7. Wave representation corresponding to the Spanish word “uno”.

IV. CONCLUSIONS AND FUTURE WORK.

HMMSim is an educational and easy-to-learn software environment which intends to help the students in the task of understanding how Discrete Hidden Markov Models work. The tool provides an interactive graphical user interface and a help menu to assist the students and teachers about the definition, training and simulation of HMMs for its possible application in pattern recognition problems. Another interesting feature of *HMMSim* is the automatic generation of equivalent C++ code for a trained HMM. Therefore, this code can be easily integrated into a C++ pattern recognition application. *HMMSim* is conceptually similar in many aspects to SNNS, although our framework is more simple.

The use of *HMMSim* in our *Digital Image Processing and Pattern Recognition* course has resulted very positive. The tool was evaluated by the course students (about 25) and by a group of teachers in our department. Students considered that the use of this software helped them to understand the application of HMMs in related recognition problems. They also pointed out that the similarities between *HMMSim* and SNNS have reduced the learning effort of both course software environments. Teachers suggested some improvements (mainly related with interfaces and some help menus) which will get better the tool and its application in our course.

Actually, we are also working in the improvement of *HMMSim* with the inclusion of the Viterbi decoding algorithm [7]. The code of *HMMSim* is also been optimized to increase its efficiency and the accuracy of datatype values. This last point is very important since the computed probability values are close to zero in a few iterations of a HMM training, even when working with double precision data.

REFERENCES

- [1] Andreas Zell et al., Stuttgart Neural Network Simulator: User Manual, University of Stuttgart, Technical Report n° 6/95, 1995.
- [2] Net-ID Inc., HMMPro Reference Manual, available in: <http://www.netid.com/html/hmmpro.html>.
- [3] S.J. Young et al, *The HTK Hidden Markov Model ToolKit Book*, Entropic Cambridge Research Laboratory, 1995. URL: <http://www.entropic.com/htk.html>.
- [4] S. Theodoridis y K. Koutroubas, *Pattern Recognition*, Academic Press, 1999.
- [5] L. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition”, *Proceedings of IEEE* 77(2): 257-286, 1993.
- [6] D. Guillevic y C.Y. Suen, “HMM Word Recognition Engine”, CENPARMI, Suite GM-606, Concordia University, Montreal, Canada.
- [7] L. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.
- [8] H. Bunke and T. Caelli (editores), “Hidden Markov Models in Vision”, Special Issue, *International Journal of Pattern Recognition and Artificial Intelligence* 15(1), 2001.
- [9] F. Samaria, “Face segmentation for identification using hidden Markov models”, *Proc. British Machine Vision Conference '93*, BMVC Press, 1993.
- [10] J.L. Camino et al., “Signature Classification by Hidden Markov Model”, *Proc. 33rd Annual 1999 Intl. IEEE Carnahan Conference on Security Technology*: 481-484, 1999.
- [11] G. Booch, J. Rumbaugh and I. Jacobson, *The Unified Modelling Language User Guide*, Addison Wesley, 1999.