An Evolutionary Programming Based Algorithm for HMM training

Ewa Figielska,Wlodzimierz Kasprzak

Institute of Control and Computation Engineering, Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
efigielska@poczta.wwsi.edu.pl, W.Kasprzak@ia.pw.edu.pl

Abstract. In this paper, we propose an evolutionary programming (EP) based algorithm for the training of hidden Markov models (HMMs), which are applied to automatic speech recognition. This algorithm (called the EP algorithm) uses specially designed operators of mutation and selection to find the HMM parameters and the number of states. In order to evaluate the recognition capability of the HMMs trained by the EP algorithm, a computational experiment is carried out using speech samples for 20 words. The results indicate that the proposed EP algorithm is a very promising tool for HMM training - the EP-HMM approach achieves an average recognition rate of 97.92% and the training process requires quite a small number of iterations of the algorithm.

1 Introduction

Statistic-based approaches, in which variations in speech are modeled statistically and which use automatic learning procedures, represent the current state of the art in speech recognition [11, 7]. An example of a probabilistic model for dynamic systems is the Hidden Markov Model (HMM), which has been successfully used in many applications, especially in speech recognition.

The topology of the HMM is described by the number of states used in it and by the connections between these states. The term hidden refers to the type of the Markov model in which the observation is a probabilistic function of the state. An HMM is usually created for each spoken word (characterized by a sequence of feature vectors) in a given lexicon. The recognition procedure consists of a feature sequence extraction step (e.g. the MFCC-based features) and a search step (e.g. dynamic programming search) for the best path in the HMM, that is most likely to generate such a sequence. Several different ways of modeling the distribution of the observed variable are: discrete, multi-dimensional Gaussian, mixture of Gaussians or tied mixture of Gaussians (also called semi-continuous model) [6]. The choice of specific distribution depends on the expected nature of real-world variable distribution and also on the size of a training set.

HMM and its extensions can be modeled using more general probabilistic modeling framework called Dynamic Bayesian Networks [10]. DBNs offer a possibility to track multiple input, output and hidden variables simultaneously and thus to model conditional probabilities in more flexible way (while retaining the

first-order Markov assumption). A widely used model for general speech recognition is the Hierarchical HMM (HHMM). This topology is based on "nesting" the simpler HMMs in larger ones, so as to give the possibility to interpret a speech signal at different levels - e.g. phoneme, word and sentence level [4].

Therefore, we ask for a procedure that finds an HMM which best describes a spoken word. The main problems in creating such a procedure are: to define the topology of the HMM and to determine the stochastic model parameters. For isolated word recognition the left-to-right topology is generally used. The classic approach to stochastic model parameter estimation is the Baum-Welch algorithm [2]. The two drawbacks of this algorithm are: the number of states must be known in advance, and a local optimization is achieved due to its hill-climbing property. To overcome these drawbacks, it was proposed to use genetic algorithms with their global optimization assumption [3]. But a large number of iterations are required to achieve a globally optimized model. In order to estimate the number of states and the model parameters in one computation step and to speed-up the search, the genetic algorithm was tied with the Baum-Welch algorithm [9].

In this paper, for training HMMs we propose an algorithm which is based on evolutionary programming (EP). This algorithm (called EP algorithm) finds both the stochastic HMM parameters (such as transition probability matrix, mixture coefficient matrix, the mean vector and covariance matrix for Gaussian distribution) and the best number of HMM states (we assume the left-to-right HMM topology). As the proposed EP algorithm is based on mutation and selection operators (evolutionary programming uses no recombination), the main effort is directed to design such a mutation operator, which ensures a good performance of the algorithm. We develop two mutation operators (the first one changes the number of HMM states, whereas the second one changes the values of HMM parameters). As the search process proceeds, the values of the mutation parameters (associated with the mutation operators) are progressively reduced so that the mutation is performed less frequently and the changes in the values of HMM parameters lessen. This mutation mechanism aims at the exploration of the search space at the beginning of the search process and the exploitation of the solutions at its end.

In order to evaluate the recognition capability of the HMMs trained by the proposed EP algorithm, a computational experiment is carried out using real features of 20 words. The results demonstrate a high word recognition rate, hence they verify a high performance of the EP-HMM approach. The results also indicate that the proposed EP-HMM approach is superior to the GA-HMM approach proposed in [3] and seems to be comparable with the hybrid GA-HMM approach [9], combining GA with the Baum-Welch method.

2 Hidden Markov Model

A HMM is usually represented as a tuple $\lambda = \{\mathbf{S}, \pi, \mathbf{A}, \mathbf{B}\}$, where [11]:

$\mathbf{S}$ - the set of N states,

π - the initial state probability vector: $\pi_j = P[s_1 = i]$, $1 \leq i \leq N$, s_1 is the initial state,

$\mathbf{A} = \{a_{ij}\}$ - the transition probability matrix, where $a_{ij} = P[s_{t+1} = j | s_t = i]$, s_t is the state at time t, and a_{ij} has the following properties:

$$a_{ij} \geq 0, \quad 1 \leq i, j \leq N, \tag{1}$$

$$\sum_{j=1}^{N} a_{ij} = 1, \quad 1 \leq i \leq N, \tag{2}$$

$\mathbf{B} = \{b_j(\mathbf{o})\}$ - the observation probability vector, where $b_j(\mathbf{o})$ is a random function associated with state s_j. In the most general model, it is a mixture of M Gaussian distributions:

$$b_j(\mathbf{o}) = \sum_{m=1}^{M} c_{jm} G(\mathbf{o}, \mu_{jm}, U_{jm}), \quad 1 \leq j \leq N, \tag{3}$$

where $\mathbf{o}$ is the observation vector, c_{jm} is the mixture coefficient, μ_{jm} is the mean vector and U_{jm} is the covariance matrix for the mth component in the Gaussian distribution mixture in state j. Coefficients c_{jm} satisfy the following constraints:

$$c_{jm} \geq 0, \quad 1 \leq j \leq N, \ 1 \leq m \leq M, \tag{4}$$

$$\sum_{m=1}^{M} c_{jm} = 1, \quad 1 \leq j \leq N, \tag{5}$$

3 The EP algorithm for HMM training

3.1 Evolutionary Programming

Evolutionary programming (originally introduced in [5]) is a search technique inspired by the principles of genetics and Darwinian natural selection. It is considered as one of the methods (next to genetic algorithms, evolutionary strategies and genetic programming) of the general approach called evolutionary computing. Its applications focus on continuous parameter optimization problems [1]. Evolutionary programming differs from the other evolutionary computing methods mainly by never using the crossover operator. It uses only inheritance from one parent: it creates an offspring only by mutation.

3.2 EP-based HMM training

The proposed EP-based algorithm operates on a population of individuals, each one representing a single HMM model. The process of looking for new solutions uses mutation operators. Two mutation operators are introduced with the purpose of (i) changing the number of states (MUT_ST operator), and (ii) changing the values of the HMM parameters, such as the transition probability matrix, the

mixture coefficient matrix, the mean vector and covariance matrix for the Gaussian distribution (MUT_VAL operator). With each mutation operator some mutation parameters are associated (the mutation probability, P_{st}, with MUT_ST; the mutation probability, P_{val}, and mutation variance, V, with MUT_VAL). As the search process proceeds, the values of the mutation parameters are progressively reduced so that the mutation is performed less frequently and the changes in the values of HMM parameters lessen. This mutation mechanism aims at the exploration of the search space at the beginning of the search process and the exploitation of the solutions at its end.

The proposed EP based algorithm developed for HMM training can be outlined as follows.

1. Set the initial values of the mutation parameters.
2. Set $t = 0$ (t is the generation index).
3. Generate and evaluate the initial population of individuals $P(t)$.
4. Repeat the following steps until stopping criterion is satisfied.
   4.1. Repeat the following loop G times (G is the population size).
      4.1.1. Select a parent from $P(t)$.
      4.1.2. Apply the mutation operators over the parent individual and produce an offspring.
      4.1.3. Copy the offspring to population $P(t+1)$.
   4.2. Evaluate $P(t+1)$.
   4.3. Replace the worst individual of $P(t+1)$ by the best individual found so far.
   4.4. Set $t = t + 1$.
   4.5. Update the values of the of the mutation parameters.
5. Return the best individual found.

The factors which characterize the EP algorithm are determined as follows.

Solution representation The solution is represented by a data structure shown in Figure 1. Because we generate the HMM model of the left-to-right type, for the representation of transition probability matrix we use a matrix tr of dimension $N \times 2$ (instead of a matrix of dimension $N \times N$) where element $tr[i][j]$ corresponds to the transition probability $a_{i,i+j}$ (the other values of a_{ij}, not represented in tr, are equal to 0).

In the decoding process, the HMM model parameters are set to values of their representatives, and then transition probabilities, a_{ij}, and mixture coefficients, c_{jm}, are normalized to satisfy constraints (2) and (5), respectively, so that:

$$a_{ij} = \tilde{a}_{ij} / \sum_{k=1}^{N} \tilde{a}_{ik}, \quad 1 \leq i \leq N, \tag{6}$$

and

$$c_{jm} = \tilde{c}_{jm} / \sum_{k=1}^{M} \tilde{c}_{jk}, \quad 1 \leq j \leq N, \tag{7}$$

where $\tilde{a}_{ij}$, $\tilde{c}_{jm}$ are the values of a_{ij} and c_{jm} before normalization.

```
struct {
   int N   // represents the number of states in the model
   int M   // represents the number of the components in the mixture of Gaussian distributions
   double tr[N][2] // represents the transition probability matrix, element
                   // tr[i][j] (i=1,..,N, j=0,1) corresponds to the transition probability a_{i,i+j}
   struct { // for every state
      double vc[M]     // represents the mixture coefficients for a given state
      double mi[M][F_dim]  //  represents the M mean vectors in the feature space, where
                           //  F_dim is the size of the feature vector (observation)
      double cov[M][F_dim]  // represents the diagonal values of M covariance matrices
   } vb[N]    // represents parameters of the Gaussian distribution mixture
} individual  // represents the HMM model
```

Fig. 1. Data structure representing the HMM model

Initial population An initial population of individuals is randomly generated. The limits for minimal parameter values (described in Section 4.2) are taken into consideration when a population is generated as well as when new solutions are created by the mutation operators.

Evaluation To measure the fitness f_g of an individual g, the value of an objective function is used which is defined as follows [9]:

$$f_g = \frac{1}{K} \sum_{i=1}^{K} \log(P(\mathbf{O}_i|\lambda_g)), \tag{8}$$

where λ_g are parameters of the HMM model represented by individual g, K is the number of observation sequences in training data. The likelihood $P(\mathbf{O}|\lambda)$ is calculated by the forward procedure [11]. The logarithm of the likelihood is computed, as $P(\mathbf{O}|\lambda)$ is often very small and exceeds the precision range of a computing machine.

Parent selection The binary tournament selection method is used. In a binary tournament selection, two individuals are randomly chosen. The better fitting one (with a greater objective function value) is then taken as a parent individual.

Mutation Two mutation operators are used: the state mutation operator MUT_ST and the value mutation operator MUT_VAL.

The operator MUT_ST changes the number of states in an individual with probability P_{st}. If the state mutation is to be performed, the index, ϑ ($N_{min} \leq \vartheta \leq N_{max}$), of the mutated state is randomly generated and then state ϑ is either

cloned (i.e. a new state has the same parameters as state ϑ and is put next to state ϑ) or removed with probability equal to 0.5. Cloning a state is introduced in our algorithm rather than including a state with randomly generated parameters because, from our experience, the latter causes a significant decrease in the fitness value of the individual.

The operator MUT_VAL changes the value of a single parameter x (in the model representation) with probability P_{val} according to the following equation:

$$x' = x \cdot G(1.0, V), \tag{9}$$

where x' is a new value of parameter x and $G(1.0, V)$ is a Gaussian random number generated with mean 1.0 and variance V.

The values of mutation parameters P_{st}, P_{val}, and V are progressively reduced (starting from their initial values) using reduction factors κ_{st}, κ_{val} and κ_V, respectively, until they achieve some minimal values. This reduction takes place when the number of successive iterations without any improvement of the best solution found so far is greater than 1.

Stopping condition The search process terminates after a predetermined number of generations is performed.

4 Experiments

In order to evaluate the recognition rate of the HMMs trained by the proposed EP algorithm, computational experiments were carried out using 20 words. The training set contains 18 utterances of every word, whereas the testing set has 12 utterances, which are different than those used for training.

Before training the HMMs by the EP algorithm, the feature sequence extraction for the words to be used in training and testing is made.

4.1 Speech features

Twelve mel-frequency cepstral coefficients (MFCC) are computed for every signal window of M (=512) samples each, with a 50% overlapping of two consecutive windows, according to the standard equations shown below [11, 7, 8] (see Figure 2).

1) The $M/2$ Fourier power coefficients ($k = 1 \ldots M/2$), for every window τ under a Hamming window function $w_r(t)$ are:

$$FC(k, \tau) = |F(k, \tau)|^2 = \left| \frac{1}{M} \sum_{t=0}^{M-1} \left[x(\tau + t) e^{\frac{-i2\pi kt}{M}} \cdot w_\tau(t) \right] \right|^2 \tag{10}$$

2) The Mel-spectral coefficients by collecting the FCs in L (=32) triangle sub-band filters $D(l)$:

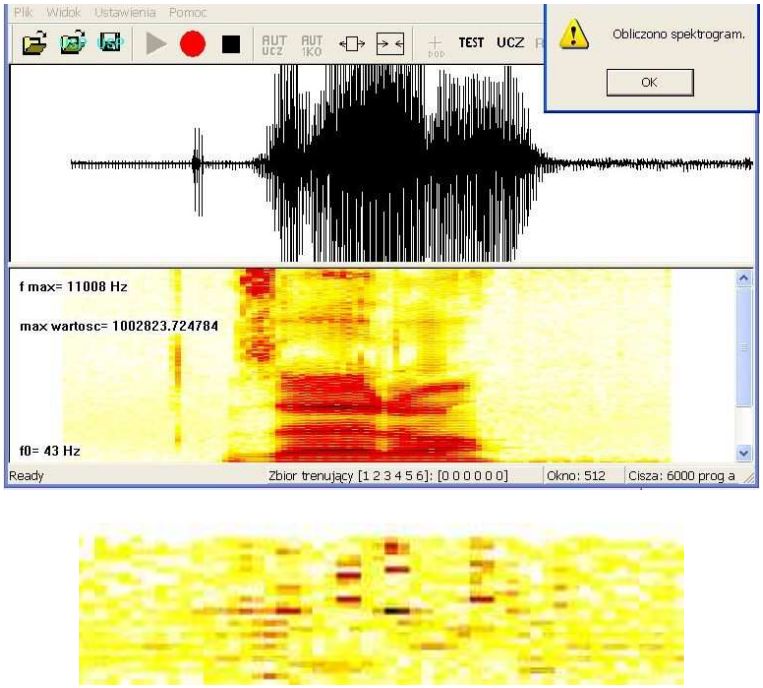

Fig. 2. Example of a speech signal (top window), its spectrogram (middle) and the set of MFCC features (bottom)

$$MFCL(l,\tau) = \sum_{k=0}^{M-1} [D(l,k) \cdot FC(k,\tau)] \tag{11}$$

distributed according to the *Mel scale*:

$$f_{mel} = 2595 \log\left(1 + \frac{f}{700[Hz]}\right) \tag{12}$$

3) The MFCC coefficients, $k = 1, \ldots, 12$, (see Figure 3) are:

$$MFCC(k,\tau) = \sum_{l=0}^{L-1} \left[\log MFC(l,\tau) \cdot \cos\left(\frac{k(2l+1)\pi}{2L}\right)\right] \tag{13}$$

4.2 EP algorithm parameterization

The bounds for the number of states of every HMM word model, N_{min} and N_{max}, are set to 5 and 20, respectively. We always assume that the model is in state s_1 at time 1. The number of components in the Gaussian distribution mixture is

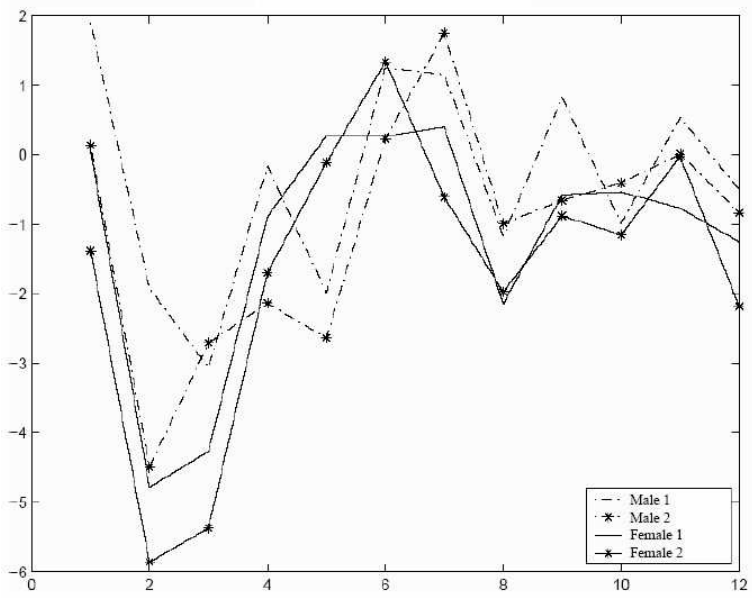

Fig. 3. Example of average MFCC vectors (with 12 coefficients each) for the vowel /a/ and for 4 speakers

fixed to 6. The mixture coefficients c_{jm} and diagonal covariance coefficients U_{jm} are constrained to be greater than or equal to 10^{-4}. Additionally, the smallest value of the transition probability, a_{ij}, for $j \in \{i, i+1\}$ is set to 10^{-6}, while for $j \notin \{i, i+1\}$ the values of a_{ij} are set to 0 according to the left-right type of the model.

To ensure a reasonable computation time of the EP algorithm working with individuals of great size (each of which representing all HMM parameters), the population size and the allowed number of iterations must not be too great. In our algorithm, the population size G is equal to 10 and the search process terminates after 2500 iterations.

The initial values of P_{st}, P_{val} and V are equal to 0.6, 0.1 and 0.5, respectively. The process of the reduction of the values of P_{st}, P_{val} and V terminates when they become equal to 0.2, 0.001, and 0.01, respectively.

Three experiments were performed with the following values of the reduction factors:

- Experiment I with $\kappa_{st} = \kappa_{val} = \kappa_V = 1.0$;
- Experiment II with $\kappa_{st} = \kappa_{val} = 0.990$, and $\kappa_V = 0.999$;
- Experiment III with $\kappa_{st} = \kappa_{val} = \kappa_V = 0.990$;

For each of the experiments 5 simulation runs were carried out.

4.3 Performance of the EP HMM training

The results of the experiment are presented in Table 1 and Figure 4.

Table 1 shows the recognition rates of HMMs obtained in each simulation of the EP algorithm for each of the experiments. In this table, we can see that the best results were obtained in experiment II. In this case, the average recognition rate has the greatest value. Moreover, the recognition rate is not less than 95% in all the simulations of this experiment what indicates that the EP algorithm with $\kappa_{st} = \kappa_{val} = 0.990$, and $\kappa_V = 0.999$ behaves more stably than the other two algorithms (used in experiments I and III). In experiment I, in one of the simulations the recognition rate of 100% was achieved.

Figure 4 shows the average (over 5 simulations) recognition rates vs. the number of generations for experiments I, II and III. In this figure we can observe, that in the first 1000 generations of the EP algorithms in experiment II and III, the average recognition rate of 95% is achieved, and that in the succeeding generations of the algorithm in experiment II, the quality of the produced HMMs still improves. In experiment I, the recognition rate of 95% was achieved after 2250 generations.

Table 1. Recognition rates (%)

	Experiment I	Experiment II	Experiment III
Simulation 1	98.75	95.00	92.92
Simulation 2	92.92	99.17	97.50
Simulation 3	92.08	99.17	99.58
Simulation 4	100.00	97.50	96.67
Simulation 5	94.59	98.75	95.00
Average	95.67	97.92	96.33

The obtained results indicate that the recognition performance of our EP-HMM approach is good. The average recognition rate obtained in experiment II is equal to 97.92%.

The proposed EP based training of HMM, regarding the computational effort (it required 2500 iterations to achieve the recognition rate of 97.92%), is superior to the genetic algorithm based training proposed in [3], which reportedly required 20000 generations to achieve the recognition rate of around 96%. The quality of results seems to be comparable with the hybrid GA approach proposed in [9].

5 Summary and concluding remarks

In this paper we have proposed the evolutionary programming based algorithm for training hidden Markov models applied to speech recognition. On basis of the

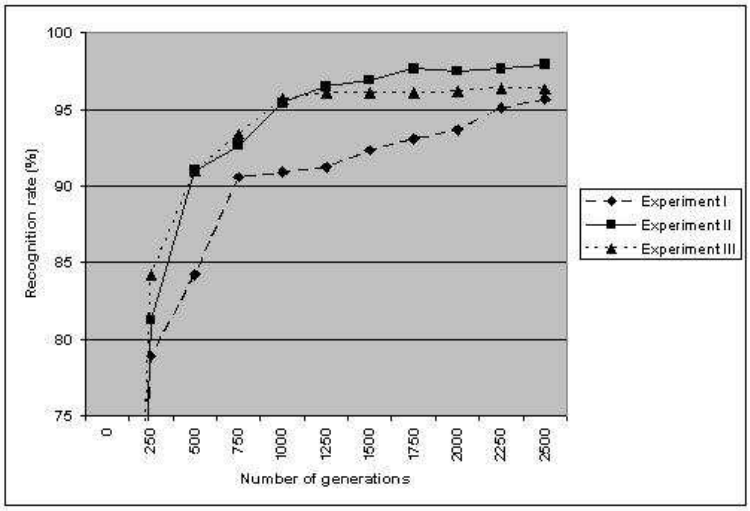

Fig. 4. The average recognition rates (over 5 simulations) vs. the number of generations for experiments I, II and III

performed experiment we can conclude that the performance of the EP-HMM approach is relatively high. The achieved average recognition rate is equal to 97.92%.

Further research should include training with a greater number of words.

Acknowledgments

The support by a research grant from the Faculty of Electronic Engineering and Information Technology of Warsaw University of Technology is gratefully acknowledged.

References

1. Back T., H.-P. Schwefel, Evolutionary computation: an overview, Proceedings of the Third IEEE Conference on Evolutionary Computation, Piscataway, NJ, 1996, pp. 20-29.
2. Baum L.E., T. Petrie, G. Soules, N. Weiss, A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, Ann. Math. Statistics, 41 (1), 1970, pp. 164-171.
3. Chau, C.W., S. Kwong, C.K. Diu, W.R. Fahrner, Optimisation of HMM by a genetic algorithm, Proceedings ICASSP, 1997, pp. 1727-1730.
4. Fine S., Y. Singer, N. Tishby: The hierarchical Hidden Markov Model: Analysis and applications. Machine Learning, 1998, pp.32-41.
5. Fogel L, A.J. Owens, M. J. Walsh, Artificial Intelligence for Simulated Evolution, 1966.

6. Gales M.J.F.: Semi-Tied Covariance Matrices for Hidden Markov Models. IEEE Transactions on Speech and Audio Processing, Vol. 2, 1999.
7. Junqua J.-C., J.-P. Haton, Robustness in automatic speech recognition, Kluwer Academic Publications, Boston etc., 1996.
8. Kasprzak W., F.A. Okazaki, R. Seta: A common feature representation scheme for speech and contour recognition, In: K. Wojciechowski et al. (eds.): Computer Vision and Graphics, ICCVG 2004. Series: Computational Imaging and Vision, vol. 32, Springer 2005, pp. 463-468.
9. Kwong S., C.W. Chau, K.F. Man, K.S. Tang: Optimisation of HMM topology and its model parameters by genetic algorithms. Pattern Recognition 34, 2001, pp. 509-522.
10. Murphy K.P.: Dynamic Bayesian Networks: Representation, Inference and Learning. Ph.D. Thesis, UC Berkeley, 2002.
11. Rabiner L. and B. Juang, Fundamentals of Speech Recognition, Prentice Hall, 1993.