GMM Parameter Estimation by Means of EM and Genetic Algorithms

Sergey Zablotskiy, Teerat Pitakrat, Kseniya Zablotskaya, and Wolfgang Minker

Department of Information Technology
University of Ulm, Germany
{segey.zablotskiy,teerat.pitakrat,
kseniya.zablotskaya,wolfgang.minker}@uni-ulm.de

Abstract. Most of the state-of-the-art speech recognition systems use Hidden Markov Models as an acoustic model, since there is a powerful Expectation-Maximization algorithm for its training. One of the important components of the continuous HMM we focus on is an emission probability which can be approximated by the weighted sum of Gaussians. Although, EM is a very fast iterative algorithm it can only guarantee a convergence to a local result. Therefore, the initialization process determines the final result. We suggested here two modifications of genetic algorithms for the initialization of EM. They are compared to the results of the EM with the same number of local multi-starts.

Keywords: Hidden Markov Model, Gaussian Mixture Model, Expectation-Maximization.

1 Introduction

The Hidden Markov Models (HMM) are commonly exploited in most speech recognition systems [1] because of their efficiency and the existence of a powerful algorithm for their learning - the Baum-Welch algorithm [2]. It is a kind of the Expectation-Maximization (EM) algorithm specially designed for HMM parameters estimation.

Acoustic features extracted from an audio signal (observations in terms of HMM) have naturally a continuous distribution. The use of vector codebooks simplifies the problem of HMM training by transforming the continuous space into vectors from a codebook. However, such conversion decreases the quality of an acoustic model and consequently the performance of the whole speech recognition system. The use of continuous HMMs is therefore strongly preferable in spite of their complexity.

One of the important HMM components we are going to focus on is an emission probability density for each state (normally, one part of a phoneme). According to the central limit theorem [3] whenever the observation is the sum of a large number of random mutually independent events having the same distribution with its own mean and variance it tends to be normally distributed. All the random variables should have approximately equal distribution scale. This implies that there should not be any dominating random variables between them. Unfortunately, this is not mostly the case when the data are coming from different sources. Therefore, for better approximation

J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2011, LNCS 6761, pp. 527–536, 2011.

of the multivariate acoustic feature density distribution it is common practice to use a weighted sum of Gaussians (GMM) instead.

EM is a very fast iterative algorithm commonly used for GMM parameters (weights, means and covariance matrices) estimation [4]. However, it can only guarantee a convergence to a local result [5] - local maximum of the log-likelihood. The EM algorithm (as any other local optimization algorithms) strongly depends on the starting procedure. Any unsuccessfully initialized local optimization algorithm gets stuck in a sub-optimal solution, which may be significantly worse than the existing best (global) one.

Quite a standard approach to get a better modeling result is the multi-start from different initial points in the search space. The use of the stochasticity is the only known acceptable way to find the global (or at least better than one of the local solutions) extremum (we do not take into account the examination of the whole search space with a certain small step, because it needs an extremely huge computational power for a multi-dimensional space).

There is a variety of stochastic algorithms for global optimization. However, all of them find the best solution in a local extremum area not faster and not more precisely than the most local search algorithms (gradient or direct search-for-optimum methods).

Since EM is a very powerful and fast local optimization algorithm for HMM parameters estimation it is worthy to use it in a combination with a global stochastic algorithm for its initialization.

In this paper we therefore describe a hybrid approach for the GMM training based on a global optimization genetic algorithm, which generates a new population of starting points for the local EM.

The paper is organized as follows. Sections 2 and 3 outline the use of the EM algorithm for GMM parameters estimation. Section 4 explains the standard genetic algorithm and Section 5 describes its implementation for EM algorithm initialization. Section 6 presents the results. Finally, the conclusions are made in Section 7.

2 The EM Algorithm for GMM Parameters Estimation

The common idea of the EM local optimization method [4] can be described in the following way. At first, the starting point for the algorithm is chosen randomly or according to some a priori information. Then, the iterative process is performed (the recursive formulas are specially designed for different tasks). At each iteration it finds a new function which is a lower bound for an objective function anywhere in the neighborhood of the current point and the value of this function is equal to the value of the objective function in the same current point. For the EM algorithm there is a convergence guarantee to a local solution or at least that it doesn't go downhill. The success of the EM algorithm depends on the objective function. Sometimes it is very hard to find such a lower bound function, but for the density estimation problem there is a good practical solution, which makes the idea of EM work very well [6]. Moreover, one of the important properties of the EM algorithm for density parameter

estimation is that it guarantees, in contrast to many other optimization algorithms, an increase of the likelihood function at each iteration, but only until a local maximum is reached.

GMM is a probabilistic model for density estimation using the Gaussian mixture distribution:

$$f(\mathbf{x} \mid \Theta) = \sum_{k=1}^{M} \alpha_k N(\mathbf{x}, \mu_k, \Sigma_k), \tag{1}$$

where

$\Theta = \{\alpha_k, \mu_k, \Sigma_k\}$, $k = 1, \ldots, M$ are parameters of GMM to be estimated,

α_k are Gaussian mixture weights $(\sum_{k=1}^{M} \alpha_k = 1)$,

μ_k, Σ_k are the mean vectors and covariance matrices,

M is the number of mixture components,

$N(\mathbf{x}, \mu_k, \Sigma_k)$ is the probability density function of the *d*-dimensional Gaussian distribution:

$$N(\mathbf{x}, \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{d}{2}} \left|\Sigma_k\right|^{\frac{1}{2}}} \exp\left(\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k)\right). \tag{2}$$

The new parameters Θ^{new} are estimated in terms of old parameters Θ^g as follows:

$$\alpha_k^{new} = \frac{1}{N} \sum_{i=1}^{N} p(k \mid \mathbf{x}_i, \Theta^g), \tag{3}$$

$$\mu_k^{new} = \frac{\sum_{i=1}^{N} \mathbf{x}_i \, p(k \mid \mathbf{x}_i, \Theta^g)}{\sum_{i=1}^{N} p(k \mid \mathbf{x}_i, \Theta^g)}, \tag{4}$$

$$\Sigma_k^{new} = \frac{\sum_{i=1}^{N} p(k \mid \mathbf{x}_i, \Theta^g)(\mathbf{x}_i - \mu_k^{new})(\mathbf{x}_i - \mu_k^{new})^T}{\sum_{i=1}^{N} p(k \mid \mathbf{x}_i, \Theta^g)}, \tag{5}$$

where

N is the training set size,

$p(k \mid \mathbf{x}_i, \Theta^g)$ is a probability of data sample $\mathbf{x}_i$ to be emitted from the k^{th} mixture component with parameters Θ^g :

$$p(k \mid \mathbf{x}_i, \Theta^g) = \frac{\alpha_k^g N(\mathbf{x}_i, \mu_k^g, \Sigma_k^g)}{\sum_{l=1}^{M} \alpha_l^g N(\mathbf{x}_i, \mu_l^g, \Sigma_l^g)}. \tag{6}$$

The above formulas perform expectation and maximization steps of EM at the same time. The results from the previous iteration become guess parameters for the next iteration. The algorithm repeats these steps until a stop condition is satisfied.

3 Initialization of the EM Algorithm

Quite a trivial approach for the initialization of parameters Θ^g is a generation of the uniformly distributed random samples in the range of the available data. However, it often leads to the underflow problem (the logarithm is calculated for a number close to 0) or simply does not find a good result.

Therefore, we used another initialization method. We will further refer to this method as a "local initialization".

The guess values for all the weights are equal to: $\alpha_k^g = \frac{1}{M}$, where $k = 1, \ldots, M$, M is the number of mixture components.

M initial mean vectors are randomly chosen without repeating and modification from the available data set for all the Gaussian elements in the mixture. Suppose that we have an observation sequence

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nd} \end{bmatrix}, \tag{7}$$

where d is the number of dimensions of the mixture model and N is the number of generated data from each dimension, the mean vectors are chosen by generating a set of random values $R = \{r_1, \ldots, r_M\}, r_k \in [1, N]$:

$$\mu^g = \begin{bmatrix} x_{r_1 1} & x_{r_1 2} & \cdots & x_{r_1 d} \\ x_{r_2 1} & x_{r_2 2} & \cdots & x_{r_2 d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{r_M 1} & x_{r_M 2} & \cdots & x_{r_M d} \end{bmatrix}. \tag{8}$$

All covariance matrices are the same diagonal with non-zero elements obtained by estimation of the variances from the data separately for each dimension:

$$\Sigma^g = \begin{bmatrix} \sigma^2(x_{i1}) & 0 & \cdots & 0 \\ 0 & \sigma^2(x_{i2}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2(x_{iD}) \end{bmatrix}, \tag{9}$$

where $i = 1, \ldots N$. This covariance matrix is used as an initial parameter for all the Gaussian elements in the mixture model.

4 Genetic Algorithm

A genetic algorithm [7] is aimed to mimic the evolution of the nature. We assume that the parameters of the problem can be encoded into chromosomes. Each chromosome represents one individual which adapts and evolves in a given environment. Stronger and better individuals tend to survive and reproduce the next generation. With the help of crossover and mutation in the reproduction process, chromosomes of the offspring inherit partially the properties of both parents and get new properties. The nature-like process will then select a better offspring for the next generation. This makes the species better and better over time.

In order to utilize the genetic algorithm, the encoding scheme of the chromosomes must be chosen. The classical encoding method is to encode them into a binary string. Each parameter is converted to a binary form and placed into a specific position in the long chromosome. Fig. 1 shows an example of two chromosomes that are encoded into a binary string, where p_i , $i = 1, \ldots n$ are parameters of the problem. All of them are encoded with a 4-bit binary code.

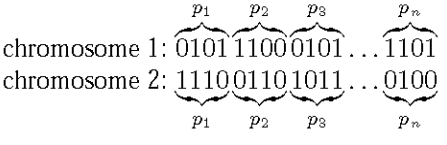

Fig. 1. Chromosome Encoding in binary

Advantages of this method are its simplicity and traceability. Gray code can also be applied to this type of encoding which makes Hamming distance between each value constant. One of the drawbacks of the binary encoding is the encoding time. All parameters must be converted from real values to binary, processed in the evolution and converted back to real values again to be evaluated for the fitness. This conversion takes time and slows down the algorithm when used with very long chromosomes.

The second drawback is the resolution of real values when converted back from binary. As a binary number is discrete, it cannot represent the precision of a floating point in real values. Hence, the number of bits needs to be considered. With an

increasing number of bits, the accuracy increases as well. However, the speed mentioned previously and the memory consumption must be taken into account as most compilers allocate the same amount of memory for one bit and one character.

Another method is to use real values directly as parameters of the algorithm (see Fig. 2). This method draws considerable interest since there is no base conversion, therefore the speed is improved. However [8] states that a real-value genetic algorithm may not always give good results depending on the application.

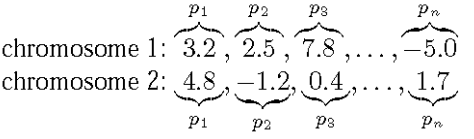

Fig. 2. Chromosome Encoding in real values

In this experiment, we use a binary encoding scheme for the parameters of each Gaussian element which are weights (α), means (μ) and covariance matrices (∑). These three parameters are converted to binary and concatenated to form a binary string as shown in Fig. 3.

α_k	$\mu_{k1}, \mu_{k2}, \ldots, \mu_{kd}$	$\sigma_{k_{11}}, \sigma_{k_{12}}, \ldots, \sigma_{k_{dd}}$

Fig. 3. Encoded parameters of one Gaussian element

Binary strings from all the Gaussians are again concatenated into a long chromosome of the Gaussian Mixture Model that will be used in the evolution process of the genetic algorithm.

Gaussian 1.	Gaussian 2.	...	Gaussian M.

Fig. 4. Encoded chromosome containing all elements of a Gaussian Mixture Model

During the evolution process chromosomes have to pass through three steps which are selection, crossover and mutation. The selection process chooses a number of chromosomes in the current generation to produce offspring. The selection criterion is primarily based on the fitness of each chromosome. Fitter individuals are more likely to be chosen for reproduction.

Crossover or recombination is a very important process that allows the chromosomes of the new offspring to have some parts of both parents. This is based on the assumption that chromosomes of the parents may contain both good and bad parts. If chromosomes of the new offspring can take some good parts from the first parent and some good parts from the second parent, they tend to become better. Crossover is a probabilistic method that may result in a worse chromosome than those of the parents. The fitness function handles it by biasing good chromosomes so that they are more likely to be selected for the next generation.

Mutation improves the genetic algorithm by altering some parts of the chromosomes that prevents the algorithm from getting stuck in a local extremum. The mutation rate should be properly determined (automatically or through experiments) since too small value may lead to the early local convergence and too large value turns the algorithm into a pure random search. Mutation causes a flip of each bit of the chromosome from zero to one and vice versa with probability equal to the mutation rate.

5 EM Multi-Start by Means of a Genetic Algorithm

The standard genetic algorithm is a global stochastic search which looks for the best solution in the whole search space. However, for high-dimensional tasks where the search space is extremely large it may fail to converge to the global solution in a limited time frame. This is called a genetic drift problem [9] and caused by the accumulation of stochastic errors due to a finite population. The problem occurs when the best solution lies in the vicinity of the found solutions but the genetic algorithm misses or fails to pick up this point.

Since the EM algorithm is very fast and powerful but is only able to find a local extremum in the area of the starting point, we decided to use a genetic algorithm for the initialization as it tends to find the global solution due to its stochastic evolutionary nature. The hybrid EM and GA method is described as follows:

```
Algorithm
  Initialize (Population);
  Evaluate_EM (Population);
  While (Generation no. < Max Number)
        Selection;
        Crossover;
        Mutation;
        Reinsertion;
        Evaluate_EM (New population);
  End while;
End
```

The first generation of GA is initialized by the method described in Section 3. Each individual (see Fig. 3 and 4) is decoded and used as an initial parameter for the Gaussian mixture model. The EM algorithm iterates until convergence to a local maximum and the found likelihood is assigned to the chromosome's fitness value. GA then performs its normal operations by creating a new generation taking into account the fitness value of each individual. These steps are continued until the maximum number of generations is reached.

6 Experiment Results

The data for the experiments (observations) were generated from a given Gaussian mixture model. These parameters were not known to the optimization algorithms and were used only at the generation step. The task was to estimate the parameters of GMM, given the data set.

The algorithms investigated in this work were multi-start EM with local initialization (see Section 3) and two hybrid Genetic Algorithms in combination with EM: randomly initialized GA and locally initialized one.

As the following step, the multi-start EM algorithm was performed. The starting point was determined by the local initialization. The algorithm iterated until a convergence to a local maximum of the likelihood was reached. Then, the best result was saved and a new initial model was again initialized locally, independently from the previous initialization and results. Finally, the best model with the highest likelihood among these multi-starts was accepted as the solution. The randomly initialized GA created the first generation randomly in the search space, regardless of the available data. Each individual (initial parameters of GMM) was passed to the EM algorithm. It then performed a standard procedure to find a local maximum of this model. The resulted likelihood was assigned to the individual as the fitness value. The next generation was a product of selection of the parent individuals, their crossover and mutation. The best individual in the last generation was then chosen as the best found model (taking into account elitism property).

Locally initialized GA performed the evaluation process with the help of EM in a similar way as random initialized GA did. The only difference was the initialization of the first GA generation. Individuals were chosen from the data by the local initialization method (the same with the EM Multi-Start initialization).

All three algorithms were set to run 1000 times of EM restarts. That is, for the standard EM multi-start, 1000 restarts were performed. For genetic modifications, where each individual was equal to one EM restart, the total number of individuals (number of individuals per generation × number of generations) had to be 1000. Furthermore, since the initialization process in EM and GA operations were all stochastic, the final results were averaged by 100 same experiments.

Fig. 5 shows the simulation results for a case, when the real number of components in the mixture was known or correctly determined (by clustering or as a priori information) and equal to 6. The dimension of the task was chosen to be 39 similarly to the dimension of SR task. The audio features extracted from a speech signal are usually the energy and 12 Mel-frequency cepstrum coefficients [10], their first and second order derivatives, i.e. 39-dimensional feature vectors.

As can be seen, the locally initialized GA performed better than randomly initialized one (for both cases: 20 and 50 individuals per generation). Locally initialized GA gave a better result (smaller value of the negative log-likelihood) than the standard EM multi-start after 200-250 restarts (the total number of individuals).

Fig. 6 shows the simulation results for a case, when the real number of components in the mixture was not known. The data was generated from GMM, consisting of 6 components and modeled by a mixture with 4 components. This is quite a standard situation for speech recognition tools (and not only) that the number of Gaussians in the model differs from the real data distribution (because of a lack of knowledge). It can be seen that both GA modifications performed significantly better than EM multi-start from the very beginning.

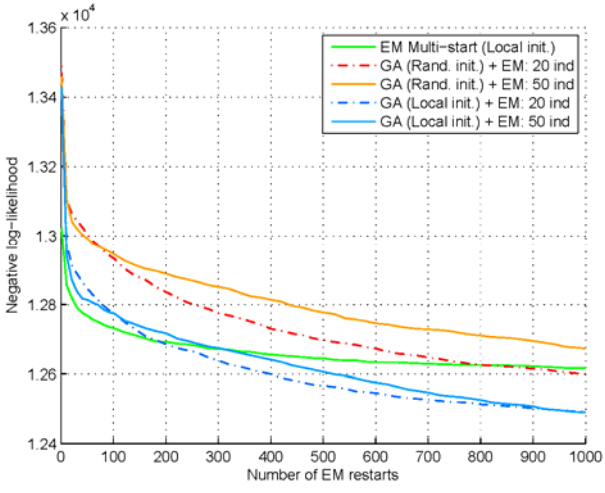

Fig. 5. EM Multi-start vs. Hybrid GA: 6 Mixture Components modeled by 6 Components

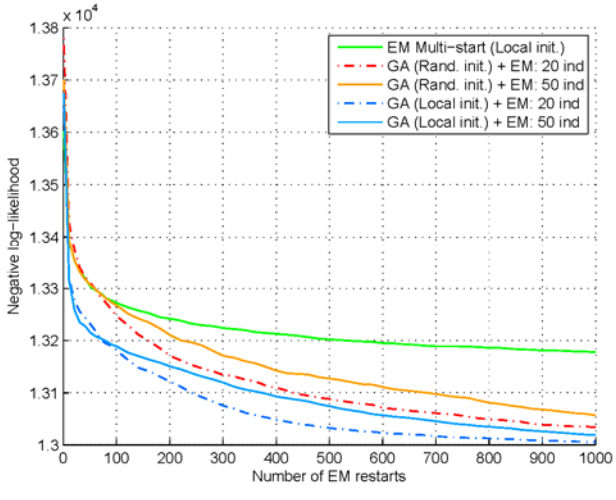

Fig. 6. EM Multi-start vs. Hybrid GA: 6 Mixture Components modeled by 4 Components

In these two simulations, a small mutation rate 0.005 (for each bit) was used which gave better results than higher mutation rates, 0.01 and 0.05.

Other types of selection and crossover were also investigated and the best methods were stochastic universal sampling and double-point crossover. However, they did not have significant influence on the likelihood value compared to the mutation rate.

The number of individuals also influenced the result. In most cases 20 individuals per generation (50 generations) led to a better result than 50 individuals (20 generations). A larger number of individuals in one population increases the chances of the

algorithm to get closer to the global minimum at the initialization step. However, at the same time it reduces the number of GA iterations for processing these starting points. Hence, the achieved result means that the GA operators outperform the pure EM algorithm restart from randomly chosen points.

7 Conclusion

The employment of the genetic algorithms for the initialization of the Expectation-Maximization algorithm mostly (and significantly) increases the likelihood of the data being generated by the found Gaussian Mixture Model. It is shown that the local initialization allows us to avoid the underflow problem and leads usually to a better result.

However, the use of genetic algorithms is worth only if there is an opportunity (computational power and time) to restart EM algorithm for a certain number of times. It does not bring us any significant effect when only few generations of GA can be performed.

References

1. Peinado, A., Segura, J.C.: Speech Recognition over Digital Channels: Robustness and Standards. John Wiley and Sons Ltd, Chichester (2006)
2. Rabiner, L., Juang, B.-H.: Fundamentals of Speech Recognition. Prentice-Hall, Englewood Cliffs (1993)
3. Rice, J.: Mathematical Statistics and Data Analysis, 2nd edn. Duxbury Press, Boston (1995)
4. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society 39(1), 1–38 (1977)
5. McLachlan, G., Krishnan, T.: The EM Algorithm and Extensions. Wiley, New York (1997)
6. Redner, R., Walker, H.: Mixture densities, maximum likelihood and the EM algorithm. Society for Industrial and Applied Mathematics (SIAM) 26(2), 195–239 (1984)
7. Davis, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
8. Goldberg, D.E.: Real-coded genetic algorithms, virtual alphabets, and block. University of Illinois, Urbana (1990)
9. Nolle, L., Battersby, A., El-Mihoub, T.A., Hopgood, A.A.: Hybrid Genetic Algorithms: A Review (2006)
10. Davis, S.B., Mermelstein, P.: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Transactions on Acoustics, Speech and Signal Processing 28(4) (1980)