**A NONLINEAR AUTOREGRESSIVE MODEL
FOR SPEAKER VERFICATION1**

Sundararajan Srinivasan2, Tao Ma3, , Georgios Lazarou4 and Joseph Picone5

*Abstract*— Gaussian Mixture Modeling (GMM) has been the most popular approach in speaker verification for over two decades. The inefficiencies of this model for signals such as speech are well documented and include an inability to model temporal dependencies that result from nonlinearities in the speech signal. The resulting models are often complex and overdetermined, which leads to a lack of generalization and overtraining. In this paper, we present a nonlinear mixture autoregressive model (MixAR) that attempts to directly model nonlinearities in the trajectories of the speech features. We apply this model to the problem of speaker verification. Experiments with synthetic data as well as with standard speech databases, including TIMIT, NTIMIT, and NIST-2001, demonstrate that MixAR, using only half the number of parameters and only static features, can achieve a lower equal error rate when compared to GMMs, particularly in the presence of noise. Performance as a function of the duration of both the training and evaluation utterances is also analyzed.

*Keywords*— Gaussian mixture models, mixture autoregressive model, nonlinear statistical models, speaker verification

Manuscript submitted February 1, 2013.

1. This material is based upon work supported by the National Science Foundation under Grant No. IIS-0414450. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
2. S. Srinivasan is with Nuance Communications Inc., 1198 East Arques Avenue  Sunnyvale, CA 94085, USA (phone: 408-992-6243; email: sundararajan.srinivasan@gmail.com).
3. T. Ma is with Siri at Apple Inc., 2 Infinite Loop, mailstop 302-4APP, Cupertino, California 95014, USA (phone: 408-643-5909; email: tma@apple.com).
4. G. Lazarou is with The New York City Transit Authority, 30-74 38th Street, Apt 1A, Astoria, New York, New York, USA 11103 (phone: (662) 617-2064; email: glaz@ieee.org).
5. J. Picone is with the Department of Electrical and Computer Engineering at Temple University, 1947 North 12th Street, Philadelphia, Pennsylvania 19027 USA (phone: 215-204-4841; fax: 215-204-5960; email: joseph.picone@isip.piconepress.com).

# Introduction

The goal in speaker verification is to accept or reject the identity claim made by a speaker. This is widely used in a variety of applications ranging from secured access and surveillance to multimodal verification. A challenge for statistical modeling in speaker verification is to accurately and efficiently represent the probability distribution of speaker features so that even similar sounding speakers can be distinguished. The majority of speaker recognition systems today utilize Gaussian Mixture Models (GMMs) either entirely or as part of a hybrid model (Beigi, 2011).

There are two well-known drawbacks of the GMM model. The first involves statistical independence – there are obviously dependencies between absolute, static, and acceleration feature coefficients. Constructing a GMM from standard features decorrelated using only a diagonal covariance matrix does not adequately model these dependencies. Use of full covariance matrices results in models with an extremely large number of parameters and creates parameter estimation problems. Performance improvements with such techniques have been minimal, and often increase the system’s sensitivity to mismatched channel conditions. A major overarching goal in our work is to improve performance when training and evaluation conditions are mismatched, and the evaluation data contains previously unseen noise and channel conditions.

The second more serious drawback, which is the focus of this work, is the implicit assumption of linearity in the MFCC dynamics. The derivatives of the cepstral features are only a linear approximation of the actual dynamics of the static features. However, a survey of studies on the subject shows that the speech signal contains significant nonlinear information, and using only derivative features to represent speech MFCC dynamics with GMM modeling is tantamount to discarding any nonlinear information present in the signal (May, 2008; Kokkinos & Maragos, 2005).

An obvious solution to this problem is to add features that can represent the nonlinear dynamic information. However, adding nonlinear invariants as features has not improved the robustness of speech and speaker recognition technologies in harsh or mismatched environments. The reasons for these failures include (1) it is difficult to estimate invariants reliably from speech, resulting in parameter estimation algorithms that need to be extensively tuned; (2) these estimation algorithms typically require an acoustic event to have a long duration (Petry et al., 2002), and this gravely undermines the applicability of invariant features for a short-term stationary signal like speech; and (3) invariants only quantify the degree of nonlinearity and do not characterize the nature of the dynamics completely.

The primary goal of this work is to approach the information representation problem at the acoustic modeling level using a nonlinear mixture autoregressive model (MixAR) (Zeevi et al., 2000), thereby accounting for the nonlinear dynamics of speech in the base model and minimizing the dimensionality of the feature space. This model is shown in Previous work on mixture autoregressive modeling for speech has been in the context of hidden Markov models for speech recognition (Juang & Rabiner, 1985). A more recent investigation of AR-HMMs (Ephraim & Roberts, 2005) used a switching autoregressive process to capture signal correlations during state transitions. Results on speech recognition showed that at best the model was only comparable to an MFCC-based HMM using a GMM observation model. Another model considered speech features as a GMM white noise process filtered through an autoregressive signal for speaker identification (Ayadi, 2008).

A more sophisticated model (Wong & Li, 2000) considers a mixture of autoregressive filters (MAR) for the observation model. Our earlier work (Srinivasan et al., 2008) considered this model for phone classification. MixAR is a generalization of MAR, where the mixture weights are allowed to be time‑varying and data-dependent. In this work, we apply the MixAR model to feature vectors in a speaker recognition task, and demonstrate improved performance over the more limited MAR model.

The rest of the paper is organized as follows: Section  formally defines the MixAR model, explains some of the relevant properties of this model and discusses the parameter estimation problem. Results of experiments using synthetic data are included in Section  and speaker verification experiments with real speech data are presented in Section . Experiments documenting variation in performance with the duration of training and evaluation utterances are also discussed in Section . Finally, in Section  we present our conclusions and discuss future directions for this research.

# The MIXAR MODEL

A mixture autoregressive process (MixAR) of order *p*with *m* components, *X*={*x*[*n*]}, is defined as (Zeevi et al., 2000):



where *ε*i is a zero-mean Gaussian random process with a variance of , “w.p.” denotes “with probability” and the gating weights, *W*i sum to 1. The linear prediction coefficients, {*a*i}, represent the dynamic model, where *a*i,0 are the component means, while *wi* and *gi* are called gating coefficients. It is apparent that an *m*-mixture MixAR process is the weighted sum of *m* Gaussian autoregressive processes, with the time-dependent weights depending on previous data and the gating coefficients.

One insightful way of viewing this model is as a process in which each data sample at any one point in time is generated from one of the component AR mixture processes chosen randomly according to its weight *W*i, as depicted in . One property of MixAR that is of particular relevance here is the ability of MixAR to model nonlinear time series (Zeevi et al., 2000; Wong & Li, 2000). Though the individual component AR processes are linear, the probabilistic mixing of these AR processes constitutes a nonlinear model. Even when the mixture weights are fixed, the model reduces to MAR, which is still nonlinear. The addition of a gating system layer for weight generation increases the flexibility of the model even further, allowing us to model distributions as a function of past data.

Several other properties of MixAR, including a mathematically rigorous proof of the asymptotic performance of a MixAR model for stochastic processes are derived in (Zeevi et al., 2000). The same work also discusses the problem of parameter estimation; however some practical implementation issues remain and we discuss them in the next. Note that in the original formulation, both the gate and prediction orders were constrained to be equal. In this paper, we restrict ourselves to MixAR models of order one to avoid difficulties during parameter estimation. We used the ISIP public domain speech recognition software (Huang & Picone, 2002) to implement the MixAR model as well as integrate it into an existing speaker verification system.

## Estimation of the Prediction and Variance Parameters

Similar to the well-known training procedure for GMM, maximum likelihood estimates for MixAR prediction and variance parameters can be calculated using the Expectation-Maximization (EM) algorithm Error! Reference source not found.. Given the order, p, the parameter set for each of the m components of a MAR model consists of *p*+1 predictor coefficients (including the mean), the error variance, and mixing weight:

|  |  |
| --- | --- |
|  |  |

To estimate these parameters, we first need an initial guess for these parameters and then we iterate with EM to successively refine the estimates. An initialization strategy that we found to work reasonably well was to first train a GMM with the same number of mixtures and then set each component of the MixAR to have the same mean, variance, and weight as the GMM model. We initialize the predictor coefficients and the data-dependency gating coefficients, {*Ai*} of MixAR to zero. These initial parameters can be then refined recursively using an E-step **Error! Reference source not found.**:

|  |  |
| --- | --- |
|  |  |

where

|  |  |
| --- | --- |
|  |  |

is the probability a sample was generated from component *l* at time instant *n*. The corresponding M-step is given by:

|  |  |
| --- | --- |
|  |  |
|  |  |

where

|  |  |
| --- | --- |
|  |  |
|  |  |

and

|  |  |
| --- | --- |
|  |  |

Refer to comments on estimation of predictor coefficients and variances for MixAR and MAR in **Error! Reference source not found.Error! Reference source not found.** for further details.

However, a complication arises with respect to the estimation of gating coefficients for MixAR. There is no closed-form solution for these, and hence a Newton gradient-ascent approach must be used:

|  |  |
| --- | --- |
|  |  |
|  |  |

where *Q* denotes the log-likelihood of the MixAR model for the training data. β and Δ are design parameters to be chosen empirically. The expression for computing Q is:

|  |  |
| --- | --- |
|  |  |

Due to this complication in the updates for the gate coefficients, the training procedure outlined above is not in the realm of strict EM algorithm but falls under a class of algorithms called as generalized EM algorithms (GEM) **Error! Reference source not found.**. For both EM and GEM algorithms, the E- step is similar. However, while an EM algorithm actually maximizes the expectation during each M-step, a GEM algorithm only guarantees that parameters that increase the model likelihood for the data is increased but does not guarantee that his is maximized at each M-step. This could mean that a GEM algorithm could take more number of iterations for training than an EM algorithm for the same or a comparable problem.

Drawing parallels with the choice of adaptation factor μ in adaptive filter theory, we can envisage that quick and smooth convergence of the GEM algorithm can be achieved by starting with a relatively high value for β and then reducing this value with successive iterations. In our experiments, we found that fixing Δ = 0.01 and running 10 iterations each with β = 0.9, β = 0.5, and β = 0.2 in succession provided a reasonably smooth and quick convergence. However, such convergence is not guaranteed in general and this poses a problem to the application of this model for real-life signals.

Fortunately, we can do better than guessing an appropriate value for β. We can use the secant method for root-finding and maximization **Error! Reference source not found.**. In general, to find the maxima using Newton’s method, the iteration is:

|  |  |
| --- | --- |
|  |  |

In the secant method, the double derivative in the denominator is estimated numerically using the secant at the point. Thus, we estimate the scaling factor β as the inverse of double derivative of the log-likelihood w.r.t. the gate parameters:

|  |  |
| --- | --- |
|  |  |

During implementation, this scheme amounts to finding for each gate coefficient *wl*, the value of Q at three different points, Q(*wl*), Q(*wl* +Δ), Q(*wl* - Δ), and then using the following update equation:

|  |  |
| --- | --- |
|  |  |

Similarly, the update equation for gate coefficients *gl* is:

|  |  |
| --- | --- |
|  |  |

Using this method, we obtain convergence curve as in Fig. 2 for the same data from speaker *4516* of NIST-2001 database **Error! Reference source not found.** used for the previous method. We find that that this method is more reliable and quick - three GEM iterations were sufficient. We use this method in future experiments.

pppppppppppp

del:





where  is a *q*-dimensional internal state vector, is a *p*-dimensional observation vector, *F* is the state evolution matrix and *H* is the observation transformation matrix. The variables ω*t* and ν*t* are assumed to be uncorrelated white Gaussian noise with covariance matrices *Q* and *R*, respectively.

The sequence of observations, , and underlying states, , are finite dimensional and are assumed to follow multivariate Gaussian distributions for every time *t*. The first equation can be viewed as an autoregressive state process that describes how states evolve from one time frame to the next. The second equation maps the output observations to the internal states. The system’s hidden states, , are the deterministic characteristic of an LDM that are also affected by random Gaussian noise. The state and noise variables can be combined into one single Gaussian random variable (Frankel & King, 2007).

Based on Figure 1, conditional density functions for the states and output can be written as follows:





According to the Markovian assumption, the joint probability density function of the states and observations becomes:



We need to estimate the hidden state evolution given  and the model parameters. This can be accomplished using a Kalman filter combined with a Rauch Tung Striebel (RTS) smoother (Frankel & King, 2007). The Kalman filter provides an estimate of the state distribution at time *t* given the previous observations. The RTS smoother gives a corresponding estimate of the underlying state conditions over the entire observation sequence. For the smoothing part, a fixed interval RTS smoother is used to compute the required statistics once all data has been observed.

The RTS smoother adds a backward pass that follows the standard Kalman filter forward recursion. In addition, in both the forward and the backward pass, we need some additional recursions for the computation of the cross-covariance. The corresponding RTS equations are:









A synthetic LDM model with two-dimensional states and one-dimensional observations was created to demonstrate the contribution of RTS smoothing. In **Error! Reference source not found.**, we show the state predictions of this LDM model using a traditional Kalman filter. In **Error! Reference source not found.**, the performance of the Kalman filter with RTS smoothing is shown. In both figures, the true state evolutions for our synthetic LDM model are compared to a scatter plot of the noisy observations of the LDM model and the RTS smoothed data. RTS smoothing produces significantly better prediction for the system’s internal states.

The Expectation-Maximization (EM) algorithm (Digalakis et al., 1993) is used to find maximum likelihood estimates of parameters for a specific word or phone, where the model depends on unobserved latent variables. The relevant equations are:







The E-step algorithm consists of computing the conditional expectations of the complete-data sufficient statistics for standard ML parameter estimation. Therefore, the E-step involves computing the expectations conditioned on observations and model parameters. The RTS smoother described previously can be used to compute the complete-data estimates of the state statistics. EM for LDM then consists of evaluating the ML parameter estimates by replacing  and  with their expectations.

The EM algorithm converges quickly and is stable for our synthetic LDM model of two-dimensional states and one-dimensional observations. After initilizing this LDM model with an identity state transition matrix and random observation matrix, the first iteration of ML parameter estimation was applied to update the model parameters. Log-likelihood scores of observation vectors were calculated and saved in order to perform further analysis.

EM training was applied for 30 iterations. After the training recursion, intermediate log-likelihood scores of observation vectors for all iterations of LDM were plotted as a function of the number of iterations. This plot is referred to as the EM evolution curve. We explored 1-, 4-, 6-, and 10-dimensions for each state in the LDM approach, and applied EM training for each specified dimension. In Figure 4, the EM evolution curve is shown as a function of the state dimension. The training procedure converges quickly, requiring no more than 10 iterations.

# Synthetic Data Experiments

One significant drawback of LDMs is that, they are inherently static classifiers — they are not capable of implicitly modeling the temporal evolution of a speech signal. Static classifiers are not designed to find the optimal start and stop times for a phone hypothesis. HMMs, on the other hand, are very good at optimizing segmentations while performing classification. Based on our previous work integrating a Support Vector Machine into a speech recognition (Ganapathiraju et al., 2004), we employed a similar two-pass hybrid HMM/LDM recognizer. This system, shown in **Error! Reference source not found.**, leverages the temporal modeling and *N*‑best list generation capabilities of the traditional HMM architecture in a first‑pass analysis, and uses a second pass to re-rank candidate sentence hypotheses with a phone-based LDM model.

Since the hybrid architecture postprocesses *N*-best lists, high performance *N*-best list generation is critical to achieving good performance. In our research, a word graph is generated and converted to an *N*‑best list using a stack-based word graph to *N*-best list converter. Word lattices or word graphs are a condensed representation of the search space. Word graphs are an intermediate representation commonly used in a multi-pass speech recognition system. Typically, a word graph contains word labels, start and stop times, a language model score and an acoustic score. To convert the word graph to an *N*-best list, a stack is initialized with the start node of the graph. A recursive procedure is then used to grow partial paths according to the word graph and to re-rank the stack to find the best partial path. During this procedure, beam pruning is applied to maintain the *K*‑best partial paths in the stack. Upon completion, the *N*‑best partial paths (*N* < *K*) are traced to produce the final *N*-best sentence hypotheses.

Once this list is produced, along with the corresponding segmentations for the acoustic units, LDM classifiers are used in a second pass to estimate the likelihood scores. In this work, a transformation-based score combination scheme is applied for simplicity. The LDM likelihood scores are first normalized (transformed) to match the range of the HMM scores, and then a weighted combination of these two scores is used. Choice of the normalization scheme and combination weight is data-dependent and requires empirical evaluation. Alternate approaches such as classifier-based score fusion and density-based score fusion could be used, but our experience was that the overall results are not sensitive to the type of score fusion used.

# Speaker Verification Experiments

In order to evaluate the hybrid HMM/LDM recognizer, the Aurora-4 Corpus (Parihar et al., 2004) was chosen because it contains mismatched training and evaluation conditions, which is a fundamental problem addressed in this work. The Aurora-4 Corpus consists of the original WSJ0 data with digitally-added noise and is divided into two training sets and 14 evaluation sets. Training Set 1 (TS1) and Training Set 2 (TS2) include the complete WSJ0 training set known as SI-84. TS1 consists of the original WSJ recordings, while TS2 contains various digitally-added noise conditions. The 14 evaluation sets are derived from data defined by the November 1992 NIST evaluation set. Each evaluation set consists of a different microphone or noise combination. In this work, we use only TS1 dataset for training and use the 14 evaluation sets for performance analysis.

Traditional 39 dimensional MFCC acoustic features (12 cepstral coefficients, absolute energy, and first and second order derivatives) were computed from each of the signal frames within the phoneme segments. Before extraction, each feature dimension was normalized to the range [-1,1] to improve the convergence behvaior of our LDM training. A total of 40 phonemes are used for acoustic modeling, so there are 40 LDM classifiers in the hybrid decoder.

The evaluation results for the clean dataset and six noisy evaluation sets are presented in Table 1. The results for the hybrid HMM/LDM decoder for the condition labeled “Clean,” which represents matched training and testing in a noise-free environment, are encouraging. The hybrid HMM/LDM system achieves an 11.6% WER which represents a 12.8% relative WER reduction compared to a comparably configured HMM baseline. The hybrid decoder also achieves 13.2% relative WER reduction for the babble noise evaluation dataset, and smaller improvements for a majority of the other conditions, which represent mismatched training and evaluation conditions. The overall results are promising given that the segmentations have not been optimized for the LDM system and confirms LDM’s capability to model speech dynamics in a manner that is complementary to a traditional HMM.

# Summary

In this paper, we proposed a hybrid framework to integrate LDMs within the framework of an HMM for large vocabulary continuous speech recognition tasks. The theoretical foundation of the linear dynamic model is discussed and an EM-based training paradigm is introduced. The hybrid decoder architecture is an off-line processing mechanism and is bootstrapped using a baseline HMM system. Several issues related to applying an LDM in a hybrid system have been addressed: modifications to the HMM system; implementation of the *N*-best list generation; and development of an *N*-best rescoring paradigm using HMM and LDM score fusion. Results on the Aurora-4 Corpus are encouraging.

In this work, the LDM postprocesses segmentations derived from the first pass of an HMM-based recognition. It is well known that segmentation plays a major role in high performance speech recognition systems. Future work will be focused on closely integrating the LDM into the core search loop of a speech recognizer, providing acoustic scores at the frame level that can be directly integrated into the Viterbi search, alleviating the need to do *N*-best rescoring. This would allow a deeper analysis of the utterance and improve performance beyond that achievable with *N*-best rescoring and fixed segmentations.

# ReferenceS

Digalakis, V., Rohlicek, J., & Ostendorf, M. (1993). ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition. IEEE Transactions on Speech and Audio Processing, 1(4), 431–442.

Frankel, J., & King, S. (2007). Speech Recognition Using Linear Dynamic Models. *IEEE Transactions on Speech and Audio Processing*, 15(1), 246–256.

Frankel, J. (2003). Linear Dynamic Models for Automatic Speech Recognition. University of Edinburgh. Retrieved from http://homepages.inf.ed.ac.uk/joe/pubs/2003/Frankel\_thesis2003.pdf.

Ganapathiraju, A., Hamaker, J., & Picone, J. (2004). Applications of Support Vector Machines to Speech Recognition. *IEEE Transactions on Signal Processing*, 52(8), 2348–2355.

Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallet, D., Dahlgren, N., & Zue, V. (1993). TIMIT Acoustic-Phonetic Continuous Speech Corpus. *The Linguistic Data Consortium Catalog*. Philadelphia, Pennsylvania, USA: The Linguistic Data Consortium. ISBN:1-58563-019-5.

Liang, F. (2003). An Effective Bayesian Neural Network Classifier with a Comparison Study to Support Vector Machine. *Neural Computation*, 15(8), 1959–1989. doi:10.1162/08997660360675107

Parihar, N., Picone, J., Pearce, D., & Hirsch, H.-G. (2004). Performance Analysis of the Aurora Large Vocabulary Baseline System. Proceedings of the European Signal Processing Conference (pp. 553–556). Vienna, Austria.

Tsontzos, G., Diakoloukas, V., Koniaris, C., & Digalakis, V. (2007). Estimation of General Identifiable Linear Dynamic Models with an Application in Speech Recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vol. 4, pp. IV–453–IV–456). doi:10.1109/ICASSP.2007.366947

Wöllmer, M., Klebert, N., & Schuller, B. (2011). Switching Linear Dynamic Models for Recognition of Emotionally Colored and Noisy Speech. *Sprachkommunikation 2010* (ITG-FB 225) (pp. 1–4). Bochum, Deutschland: Springer.

# LIST OF FIGURES

**Error! Reference source not found.**

**Error! Reference source not found.Error! Reference source not found.**

**Error! Reference source not found.Error! Reference source not found.**

**Error! Reference source not found.**

**Error! Reference source not found.**

# Figures

Figure 2. Performance of (Generalized) EM using the secant method as a function of the number of iterations for an 8-mixture MixAR model is shown (speaker *4516* from the NIST-2001 database).

(a)



(b)



Figure 1. An overview of the (a) GMM and (b) MixAR approaches. The MixAR model is a weighted sum of Gaussian autoregressive models with time-dependent weights.

.

# LIST OF TABLES

Table 1. Experimental results for the hybrid HMM/LDM system are compared to a conventional HMM system. Substantial improvements were obtained on the clean and babble noise conditions.

# Tables

|  |  |  |  |
| --- | --- | --- | --- |
| **Condition** | **HMM****Baseline** | **Hybrid****LDM** | **RelativeReduction** |
| Clean | 13.3 | 11.6 | **12.8%** |
| Airport | 53.0 | 50.3 | 5.09% |
| Babble | 55.9 | 48.5 | **13.2%** |
| Car | 57.3 | 59.8 | -4.4% |
| Restaurant | 53.4 | 50.6 | 5.2% |
| Street | 61.5 | 59.4 | 3.4% |
| Train | 66.1 | 63.4 | 4.1% |

Table 1. Experimental results for the hybrid HMM/LDM system are compared to a conventional HMM system. Substantial improvements were obtained on the clean and babble noise conditions.