# AUTOREGRESSIVE HMM SPEECH SYNTHESIS†

*Carl Quillen*

MIT Lincoln Laboratory,
Lexington MA 02420, USA
cbq@ll.mit.edu

**ABSTRACT**

Autoregressive HMM modeling of spectral features has been proposed as a replacement for standard HMM speech synthesis. The merits of the approach are explored, and methods for enforcing stability of the estimated predictor coefficients are presented. It appears that rather than directly estimating autoregressive HMM parameters, greater synthesis accuracy is obtained by estimating the autoregressive HMM parameters by using a more traditional HMM recognition system to compute state-level posterior probabilities that are then used to accumulate statistics to estimate predictor coefficients. The result is a simplified mathematical framework that requires no modeling of derivatives and still provides smooth synthesis without unnatural spectral discontinuities. The resulting synthesis algorithm involves no matrix solves and may be formulated causally, and appears to result in quality very similar to that of more traditional HMM synthesis approaches. This paper describes the implementation of a complete Autoregressive HMM LVCSR system and its application for synthesis, and describes the preliminary synthesis results.

## 1. INTRODUCTION

Recently autoregressive modeling has been proposed as an alternative to the standard HMM approach[1]. It was proposed as a simpler alternative to standard HMM synthesis techniques that use parameters derived from a standard HMM, e.g. [2, 3], or the more complicated HMM trajectory model[4]. The autoregressive formulation generatively models an observation distribution that is properly normalized and that can directly be used to synthesize speech parametrically without any audible discontinuities. Nevertheless while this formulation performs better than an ordinary HMM system implemented without making use of feature derivatives, it generally seems to be inferior to HMM systems trained with them, and on its own it results in an inferior synthesis system. Approaches to overcoming these problems are explored below.

In its most basic form, autoregressive modeling uses a set of predictor coefficients $a_1 \ldots a_r$ to predict an observation $x(t)$ at time $t$ given the previous $r-1$ observations $x(t-i)$, for $i \in 1 \ldots r-1$:

$$x(t) \sim \sum_{i=1}^{r-1} a_i x(t-i) + a_r. \tag{1}$$

Note the use of a constant offset $a_r$ here. The observations could be scalar or vector, but for the purposes of this paper, we will only consider separate autoregressive coefficients for each feature dimension, modeling prediction error in a diagonal Gaussian approach. (The presentation is somewhat simplified below in places by presenting only one scalar element of the feature vector.) One should also note that the synthesis feature vectors will be derived from LSF parameters, and are not the raw waveform itself.

In synthesis, the prediction coefficients are used to directly generate each new $x(t)$. If the coefficients form a stable recursion then we can view the constant offset $a_r$ as being related to a *target* $\vec{g}$ through the equation $a_r e_0 = (I - A)\vec{g}$, where $e_0 = (1, 0, \ldots 0)^T$. $\vec{x}(t)$ will eventually approach $\vec{g}$ as time $t$ increases. This can be understood by writing (1) in matrix-vector form. Let us denote by $\vec{x}(t)$ the vector composed of the (scalar) observation at current and previous times:

$$\vec{x}(t) = (x(t), x(t-1), \ldots, x(t-r+1))^T .$$

Then $\vec{x}(t) - \vec{g} = A(\vec{x}(t-1) - \vec{g})$ where

$$A = \begin{pmatrix} a_1 & a_2 & \ldots & a_{r-1} \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}. \tag{2}$$

If $A$ has spectral radius less than one then we can expect $\vec{x}(t)$ synthesized from the prediction will eventually converge stably to the target $\vec{g}$. Synthesizing speech then is a matter of picking the correct regression coefficients and targets for each phonetic unit needed in the desired utterance. Then the synthesis algorithm will (schematically) be something like the following:

- Given written text for a desired utterance, convert the words using pronunciation dictionaries and letter-to-sound rules to a list of desired phonetic units ordered by time.
- For each phonetic unit HMM state, assign a duration to it.
- For each frame in each HMM state, find the right set of prediction coefficients to use. If Gaussian mixture models were used to estimate prediction coefficients, this will involve a choice of which Gaussian mixture was optimal in the context of the utterance.
- Starting from an initial $x(r-1) \ldots x(0)$, iterate using the assigned $a_1(t) \ldots a_r(t)$ to generate all the $x(t)$ across the utterance.

This sequence of steps is causal and involves no matrix solves across the utterance, although finding a good selection of Gaussian mixture elements may require some lookahead.

† This work was sponsored by the United States Air Force Research Laboratory under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

## 2. DIAGONAL GAUSSIAN AUTOREGRESSIVE MODELING

Suppose we have a $d$-dimensional Gaussian mixture HMM observation density $P$ for each HMM state $s$ of the form

$$P(x(t)|x(t-1)\dots x(t-r+1), s) = \sum_m w_m \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi v_i^m}} \exp\left(-\frac{1}{2v_i^m}(\vec{a}_i(m)\cdot\vec{x}_i(t))^2\right) \quad (3)$$

where the vector of of $r+1$ predictor coefficients $\vec{a}_i(m) = (a_i^0, a_i^1, \dots a_i^r)$, with $a_i^0 \equiv 1$ and $\vec{x}_i(t)$ now denotes the $i$-th dimension of the observations at current and previous times concatenated with 1:

$$\vec{x}_i(t) = (x_i(t), x_i(t-1), \dots, x_i(t-r+1), 1)^T .$$

$w_m$ are scalar mixture weights $\sum_m w_m = 1$. Deriving update equations via the EM algorithm for this model is straightforward. The E-step results in the update

$$\vec{a}_i', v_i' = \arg\max_{\vec{a}_i, v_i} -\sum_t \frac{1}{2v_i}\vec{a}_i^T R_i \vec{a}_i + \frac{1}{2}\log v_i - \lambda(a_i^0 - 1),$$

where $R_i(m) = \sum_t \gamma_t(m)\vec{x}_i\vec{x}_i^T / \sum_t \gamma_t(m)$ is an autocorrelation matrix. Maximizing this equation results in the following equations for the updated values $\vec{a}_i', v_i'$ :

$$\vec{a}_i' = \frac{R_i^{\div} e_0}{e_0^T R_i^{\div} e_0},\ v_i' = \frac{1}{e_0^T R_i^{\div} e_0} \quad (4)$$

Here $R_i^{\div}$ denotes the pseudo-inverse of $R_i$, which is the same as the inverse when $R_i$ is non-singular, and $e_0$ is again the elementary vector $(1, 0\dots 0)^T$. In the singular case where $e_o^T R_i^{\div} e_0 = 0$, the variance $v_i$ may be assigned a minimum value as a variance 'floor'. However this problem did not occur in any of the training sets explored here.

### 2.1. Controlling Stability of the Prediction Coefficients

Equation (4) contains no mechanism for controlling the spectral radius of the prediction matrix $A$ in (2), and this is inadequate for synthesis. If unstable coefficients are estimated in any phonetic unit, during synthesis an exponential divergence in a feature parameter may occur and it may well result in audible artifacts generated for that unit. However $A$ has a simple form, allowing the characteristic polynomial to be calculated by inspection, which results in the following equation for its eigenvalues $\lambda$ :

$$\lambda^{r-1} - \sum_{i=1}^{r-1} a_i \lambda^{r-1-i} = 0. \quad (5)$$

Because the desired regression order is typically rather low, ($r \sim 3$) calculating the roots of this polynomial is very easy, and the spectral radius may be directly estimated at low cost.

We propose two different methods for controlling the spectral radius. One is to constrain a weighted norm of $\vec{a}$, as suggested in [5]. This amounts to adding a constant to the diagonal of the autocorrelation matrix $R$. A shooting method can be used to adjust the constant amount added until the spectral radius $\leq 1$. This approach isn't guaranteed to result in an iteration that increases likelihood, but in practice it seems to work well. Another approach is to calculate all roots of (5) and then normalize the magnitude of any that lie outside the unit circle. The coefficients of the resulting polynomial then provide the updated $\vec{a}$. Again this may result in EM iterations which lower likelihood, but nevertheless it appears to work well.

### 2.2. Front End Features for Synthesis

Synthesis and speech recognition are fundamentally different problems, and so we might expect that the optimal analysis features for recognition and computing state/frame alignments would be different than the ones best suited for synthesis. Fortunately, there is no need to compromise in this regard. We can compute state and Gaussian posterior probabilities for each Gaussian of a good recognition system, and then use these posterior probabilities to accumulate counts for different analysis features. These counts can then be used to estimate an observation probability model for the synthesis HMM system. This is the approach used here. The synthesis features used are described in [5]. They are log-differences of LSF's, pitch and a voicing log probability parameter. These parameters are readily used to synthesize waveforms and appear to be well-modeled in a diagonal-Gaussian framework.

### 2.3. Gaussian Selection

We attempted to implement Gaussian selection using a simplified version of the method presented in [3]. This involved an iterative selection similar to EM, but in the version that was tried, time alignments were chosen a-priori and not allowed to change. The end result was a visually sharper and more pleasing spectrum, but unfortunately it was less intelligible and resulted in a degradation in measured PESQ MOS-LQO scores[6]. (See section 4.1 for a description for how PESQ was used to evaluate the synthesis system.) Therefore the results below are presented only in the single-Gaussian case. It seems likely that a better Gaussian selection algorithm would ultimately produce gains from mixture modeling.

### 2.4. Phone Duration

One approach to computing phone duration is to explicitly model it in the HMM framework as in [3]. Because the experiments reported here relied on a heavily modified HMM training and recognition system that did not support this kind of duration modeling we created a separate explicit duration model for each desired phonetic unit in the test transcript. We carried out forced alignment of our training corpus with a standard HMM system and then trained models from the measured durations. Because most possible triphones are not observed in training corpora of reasonable size, some form of interpolation is needed to provide duration estimates for unseen triphones. Several different approaches to this were tried: matrix factorization, by the Tucker tensor decomposition and by state clustering.

The state clustering approach used the same software for state-clustering as the HMM recognition system, and involved single-Gaussian duration models for each training triphone. A cluster tree was grown using a list of phonetic questions for the triphone context in a greedy fashion in a way that minimized entropy of the resulting triphone clusters. Unseen triphone duration may then be estimated by walking this tree asking the relevant questions about the phoneme context until the correct leaf cluster is found. The duration of that cluster can then be used as the mean duration estimate.

The Matrix factorization approach calculates approximations to the full positive duration matrix $D(p)$ for each center phoneme $p$.

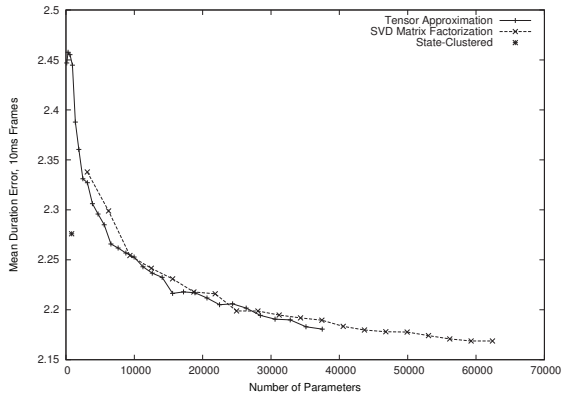


**Fig. 1**. Mean Duration Error in Frames, TIMIT Database

If we assign an integer index $i$ for each phoneme $p$, where $i = f(p)$, for some indexing function $f$, then $D_{f(o)f(q)}(p)$ which is the $f(o), f(q)$ entry of $D(p)$ is defined to be equal to the average duration estimate of the triphone $o, p, q$ seen in training and zero for triphones not seen. We can compute estimates for triphones that we have not seen in training by finding a low-rank factorization of $D(p)$, so that $D(p) \approx A(p)B(p)$. If there are ~40 phonemes in the system then $D(p)$ is a $40 \times 40$ matrix. If we choose $A$ to have much lower rank (e.g. a $40 \times 10$, and so $B$ is $10 \times 40$) then $A(p)$ and $B(p)$ will have many fewer parameters than $D$ and may be trained robustly using the amount of data available in the training set. There are several approaches one might take to doing this factorization. We tried non-negative matrix factorization[7] as well as rank-reduced SVD (singular value decomposition, also known as latent semantic analysis) of $D(p) - \bar{p}$, where $\bar{p}$ denotes the average duration of the phoneme $p$.

The Tucker tensor decomposition[8] is a rather similar concept to the matrix factorization approach, based on approximating a tensor using iterative application of SVD to the individual tensor indices, which in this case were the three phoneme labels of the triphone. We tried this using various choices for the core tensor dimensions. Results were rather similar to the SVD matrix factorization.

The results of evaluating these different duration models are explored in section 3.1 below.

## 3. EXPERIMENTS

### 3.1. Duration Model Evaluation

Figure 1 presents the results of evaluating the three different methods of section 2.4 on the TIMIT database. The models were evaluated by training on the 173490 triphones of training from TIMIT and tested on 50124 triphones of held-out test. The test triphones excluded triphones with a center phoneme of silence. The phoneme set used was a reduced 39 phoneme one, plus a boundary phone context used at utterance beginnings and endings.

For the state-clustered duration model, various choices of count and likelihood thresholds were chosen for the clusters, and the point plotted was the one for which the best test-set performance was observed. This occurred for 833 state clusters. The resulting mean error of 2.28 frames is obviously much better than the matrix factorization approach at the same number of parameters. Matrix approaches can train models with very large numbers of parameters without over-training and they ultimately achieve significantly better performance. For this training there are only $40 \times 39 \times 40 = 62400$ possible triphones (including utterance boundary context), which is about equal to the number of parameters for the largest model trained.

| System | Clusters | Gaussians | WER |
|---|---|---|---|
| Standard ML | 4625 | 38644 | 7.2% |
| Autoregressive | 4625 | 35292 | 13.4% |
| Autoregressive, spectral radius control | 4625 | 35187 | 13.1% |
| Standard ML, no derivatives | 4647 | 33941 | 35.4% |

**Table 1**. WSJ Word Error Rate Comparison

The Matrix Factorization approach presented here is the SVD one. We also tried a maximum-entropy NNMF approach, but the results were much worse.

The duration model was also tested on a 195305 phoneme single-speaker corpus and tested on 7486 triphones of test data. The results were similar, but unsurprisingly the estimated duration error was lower, by about 0.4 frames.

It is of course possible to fuse these different duration estimation systems together in order to get a more accurate estimate. Simple averaging did not seem to result in a meaningful improvement in performance, but this possibility may merit further exploration.

In order to employ this duration model in synthesis, we actually require state durations for each state of the synthesis HMM. We used a state-clustered approach to modeling this in the experiments reported below:

- The synthesis HMM system was created using a state cluster tree that was constrained to be identical across each state.
- Forced alignment of training data was extended to provide state-level alignments.
- The fraction of the total phoneme duration that was expended in each state of each triphone cluster was estimated by simple averaging of the times measured in the training data.

Triphone state durations during test were computed by estimating a particular triphone duration using one of the previous techniques, then looking up the state time fractions for the triphone state cluster that contains that triphone. Synthesis state time durations were then computed as the simple product of these results.

## 4. RECOGNITION AND SYNTHESIS EXPERIMENTS

We created a baseline ML-trained unadapted non-crossword triphone recognition system and trained it on the WSJ0 corpus, testing on the 1992 evaluation corpus. The recognition system was a simple FST-based trainer and LVCSR engine that performs about as expected on this corpus. This system used a standard mel-cepstral front end, using 13 mel-cepstral features with first and second derivatives and causal cepstral mean removal. State-clusters were extracted from this system and the same state clustering used to create a single-Gaussian-per-state autoregressive system that used just the 13 mel-cepstral coefficients as features. The regression order $r$ for this system was 3. It was trained using iterative Gaussian splitting and ML training. The performance of these two systems is compared with a standard ML system initialized in the same way as the Autoregressive system used in table 1.

| System Training | MOS-LQO | State Clusters |
|---|---|---|
| EM trained | 2.16 | 3756 |
| Direct Estimation | 2.34 | 3751 |
| Direct Estimation | 2.36 | 4709 |
| Direct Estimation | 2.36 | 6481 |

**Table 2**. Narrow-Band PESQ MOS-LQO Measurements, 53 Utterances Read Speech

There is obviously a performance degradation due to using the the autoregressive modeling, but it is still much better than the identical system created without derivatives. There are also many avenues available for improving performance of the autoregressive system, but absolute performance of it as a recognition system may well not be very important. Synthesis quality seems to be dominated by the performance of the recognition system used to estimate state posterior probabilities in training, and this can be a different (and better) recognition system.

### 4.1. Synthesis Experiments

We used an approach which is somewhat similar to the way bluescores are used in the machine translation community to measure synthesis accuracy to enable rapid development and optimization of system parameters. This was accomplished by prerecording test utterances using a live test speaker, doing forced alignment of these utterances relative to the transcript and then using the derived time marks to synthesize a comparison waveform. The reference recorded waveform was then compared acoustically with the synthesized one using the PESQ algorithm[6], which is an automatic perceptually-weighted tool in wide use in the speech coding community commonly used for measuring speech coder quality. While this approach is not perfect, we observed that improvements in modeling sophistication (and code bug fixes) that resulted in gains in recognition performance on WSJ tended to be highly correlated with PESQ score improvements in synthesis. The PESQ results (MOS-LQO numbers) are normalized to attempt to match human subject MOS measures of quality rated on a 1 to 5 scale. Improvements of the order of 0.1 point in MOS-LQO tend to result in quite noticeable audible improvements. The PESQ best scores reported here

The HMM synthesis system was a fully crossword state-clustered triphone system. State clustering was constrained so that the same cluster tree was used across all three states of all triphone clusters. Single-Gaussian-per-state-cluster systems were created in two different ways. One was by direct estimation of statistics using posterior probabilities estimated from a standard HMM system that used traditional cepstral features and their derivatives. In another approach, this autoregressive system was trained from a flat start using several passes of EM. An initial evaluation was carried out on the same four-hour single-speaker database used in [5]. A separate read speech test corpus was collected for PESQ score measurements. PESQ MOS-LQO measurements were carried out with voicing forced on and with a constant fixed pitch. An analysis-synthesis system was used to resynthesize the reference test utterances so that they also had the same constant pitch and voicing, and no post-filtering was applied. Thus the comparison only measured coarse spectral fidelity.

Table 2 presents the results of PESQ measurement as the number of triphone clusters increased. The results were were still generally quite intelligible and this improved substantially when voicing and pitch were added back to the synthesis.

We also implemented a full synthesis system using pronunciation dictionaries and the duration model presented in 2.4, training on voices from the CMU Arctic database. The results were natural sounding and very intelligible, but probably were not as good as the freely available HTS system on the same utterances. Adding post-filtering would improve the perceived quality substantially. The results appear to be promising and there are many avenues for improvement, including using Gaussian mixtures, mixed excitation speech models, more tightly integrated duration models, etc.

## 5. CONCLUSIONS

The results presented here are obviously preliminary and do not make a strong case for use of this formulation. Nevertheless, because statistics for the synthesis HMM may be estimated using posterior state probabilities from a better performing recognition system, it seems reasonable to expect that a full synthesis system implemented in an autoregressive framework with all the same features of existing HMM systems would perform just as well, and with a considerable potential reduction in computational complexity. It is also possible that the autoregressive formulation would have advantages in a adaptive framework, given that the regression coefficients can easily be adapted by the MLLR algorithm. Finally we found that matrix and tensor approximation approaches to duration modeling provided a substantially more accurate estimate of average triphone duration than a more traditional state-clustered approach.

## 6. REFERENCES


[1] M. Shannon and W. Byrne, "Autoregressive HMMs for speech synthesis," *Proc. Interspeech*, 2009.

[2] T. Masuko, K. Tokuda, T. Kobayashi, and S. Imai, "Speech synthesis using HMMs with dynamic features," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 1, pp. 389–392, IEEE, 1996.

[3] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[4] K. Zen, H. Tokuda and T. Kitamura, "Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences," *Computer Speech and Language*, vol. 21, no. 1, pp. 153–173, 2006.

[5] C. Quillen, "Kalman filter based speech synthesis," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 4618–4621, IEEE, 2010.

[6] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *Proceedings ICASSP 2001*, vol. 2, pp. 749–752, 2001.

[7] D. Lee and H. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, 2001.

[8] L. R. Tucker, "Some mathematical notes on three-mode factor analysis"," *Psychometrica*, vol. 31, pp. 279–311, 1966.