UNIVERSITY OF
CAMBRIDGE
Department of Engineering

A formulation of the autoregressive HMM for speech synthesis

Matt Shannon
sms46@eng.cam.ac.uk

William Byrne
bill.byrne@eng.cam.ac.uk

Technical Report
CUED/F-INFENG/TR.629

31 August 2009

Cambridge University Engineering Department
Trumpington Street
Cambridge, CB2 1PZ
U.K.

Abstract

We present a formulation of the *autoregressive HMM* for speech synthesis and compare it to the standard HMM synthesis framework and the trajectory HMM. We give details of how to do efficient parameter estimation and synthesis with the autoregressive HMM and discuss consequences of the autoregressive HMM model.

There are substantial similarities between the three models, which we explore. The advantages of the autoregressive HMM are that it uses the same model for parameter estimation and synthesis in a consistent way, in contrast to the standard HMM synthesis framework, and that it supports easy and efficient parameter estimation, in contrast to the trajectory HMM.

1 Introduction

It has been shown that it is possible to synthesize natural sounding speech with HMMs and the quality of the best HMM-based synthesis systems now rivals the best unit selection synthesis systems [1]. A breakthrough that helped make this possible was realizing how to use dynamic feature information during synthesis by respecting the constraints between static and dynamic features [2].

However the established approach to HMM-based synthesis is inconsistent in the enforcement of these constraints [3]. During synthesis we take the constraints between static and dynamic features into account, whereas during parameter estimation we assume the static and dynamic feature sequences are independent.

This is a recognized problem and has been addressed previously. Zen showed how a *trajectory HMM* [3] could be employed so that the same model is used for both parameter estimation and synthesis in a consistent way. Synthesis quality improved as a result [3]. However parameter estimation for the trajectory HMM is more complicated than for the standard HMM, requiring alignment with a delayed-decision Viterbi algorithm and gradient-based parameter re-estimation procedures [3]. The challenge remains to find a model which can easily and consistently be used for both parameter estimation and synthesis.

In this technical report we give a formulation of the *autoregressive HMM* [4, 5, 6, 7] for speech synthesis. The autoregressive HMM relaxes the traditional HMM conditional independence assumption, allowing state output distributions which depend on past output as well as the current state. In this way the autoregressive HMM explicitly models some of the dynamics of speech.

Autoregressive HMMs have been used before for speech recognition [4, 5, 6, 8] but to our knowledge they have not been previously investigated for speech synthesis. Note that for the autoregressive HMM considered here the observations are acoustic feature vectors. This is distinct from the *hidden filter HMM* (also sometimes called the autoregressive HMM) [9, 10] for which the observations are waveform samples.

In §2 we specify the autoregressive HMM model, show how to do efficient parameter estimation and synthesis, and investigate aspects of the autoregressive HMM model. In §3 we review the standard HMM synthesis framework and the trajectory HMM. In §4 we compare the autoregressive HMM to the standard HMM synthesis framework and the trajectory HMM. Finally in §5 we give conclusions.

2 Autoregressive HMM

In this section we specify the autoregressive HMM model, show how to do efficient parameter estimation and synthesis, and investigate aspects of the autoregressive HMM model.

2.1 Model

We first describe a general generative model for sequences of acoustic feature vectors. Conceptually we first generate a *hidden* state sequence $\theta = \theta_{1:T}$ and then generate an *observed*

or *output* feature vector sequence $c = c_{1:T}$ given this state sequence. We consider models with a joint probability distribution of the form:

$$P(c, \theta) = \prod_t P(\theta_t|\theta_{t-1})P(c_t|c_{1:t-1}, \theta_t) \tag{1}$$

The *state transition probabilities* $P(\theta_t|\theta_{t-1})$ are conditioned only on the previous state. The *state output distributions* $P(c_t|c_{1:t-1}, \theta_t)$ are conditioned on both the current state and all past output. This is in contrast to the standard HMM assumption that the state output distribution $P(c_t|\theta_t)$ is conditionally independent of past output.

The *autoregressive HMM with summarizers* specializes the above to a particular form of output distribution $P(c_t|c_{1:t-1}, \theta_t)$. We assume c_t is conditionally Gaussian, with covariance depending only on the state θ_t. Rather than allowing the mean for each state to be an arbitrary function of past output $c_{1:t-1}$, we restrict it to be an affine function of a fixed set of *summarizers* of past output. Each summarizer f^d is a function that takes the entire past output $c_{1:t-1}$ and produces a vector-valued *summary* $f^d(c_{1:t-1})$. We consider state output distributions of the form:

$$P(c_t|c_{1:t-1}, \theta_t) = \mathcal{N}\left(c_t|\mu_{\theta_t}(c_{1:t-1}), \Sigma_{\theta_t}\right) \tag{2}$$

$$\mu_q(c_{1:t-1}) = \sum_{d=1}^{D} A_q^d \left(f^d(c_{1:t-1}) - \mu_q^d\right) + \mu_q^0 \tag{3}$$

where Σ_q is a state-dependent covariance matrix, A_q^d is a rectangular matrix for each summary d and state q, μ_q^0 is a state-dependent bias vector, and following Woodland [6], we have introduced redundant bias vectors μ_q^d for each summary d and state q as a trick to make re-estimation easier. The set of parameters specifying the autoregressive HMM is therefore $(A_{qij}^d, \mu_{qi}^d, \mu_{qi}^0, \Sigma_{qij})$, where q ranges over states, i and j range over feature vector components, d ranges over summarizers, and A_{qij}^d is the (i, j)-component of the matrix A_q^d in (3).

Using diagonal square matrices in (3), that is $A_{qij}^d = a_{qi}^d \delta_{ij}$ and $\Sigma_{qij} = \sigma_{qi}^2 \delta_{ij}$ for some a_{qi}^d and σ_{qi}^2, our state output distributions become:

$$P(c_t|c_{1:t-1}, \theta_t) = \prod_i \mathcal{N}\left(c_{ti}|\mu_{\theta_t i}(c_{1:t-1}), \sigma_{\theta_t i}^2\right) \tag{4}$$

$$\mu_{qi}(c_{1:t-1}) = \sum_{d=1}^{D} a_{qi}^d \left(f_i^d(c_{1:t-1}) - \mu_{qi}^d\right) + \mu_{qi}^0 \tag{5}$$

The set of parameters specifying the autoregressive HMM is now $(a_{qi}^d, \mu_{qi}^d, \mu_{qi}^0, \sigma_{qi}^2)$.

We further assume that the i^{th} summarizer $f_i^d(c_{1:t-1})$ depends only on the i^{th} feature vector component $c_{(1:t-1)i}$, and so $P(c|\theta) = \prod_i P(c_i|\theta)$, i.e. the feature vector sequence components are independent given the state sequence. This is a common assumption when modelling speech using HMMs. We generally refer to this model simply as the *autoregressive HMM*.

We are free to choose the summarizers (f^d) to be anything which might distill useful information about past output. However for simplicity we usually take each f^d to be a fixed linear combination of the past K feature vectors[1]:

$$f^d(c_{1:t-1}) = \sum_{k=-K}^{-1} w_k^d c_{t+k} \tag{6}$$

[1] see §2.5.1 for a discussion of what to do for the initial frames $t \leq K$

window	offset			
	-3	-2	-1	0
$w_\cdot^1$			1.0	
$w_\cdot^2$		-1.0	1.0	
$w_\cdot^3$	1.0	-2.0	1.0	

(a) typical autoregressive

window	offset			
	-3	-2	-1	0
$w_\cdot^1$			1.0	
$w_\cdot^2$		1.0		
$w_\cdot^3$	1.0			

(b) canonical autoregressive

window	offset						
	-6	-5	-4	-3	-2	-1	0
$w_\cdot^1$						1.0	
$w_\cdot^2$				1.0			
$w_\cdot^3$	1.0						

(c) fixed offset autoregressive

window	offset		
	-1	0	+1
$w_\cdot^1$	-0.5	0.0	0.5
$w_\cdot^2$	1.0	-2.0	1.0

(d) standard HMM synthesis

Table 1: examples of window coefficients

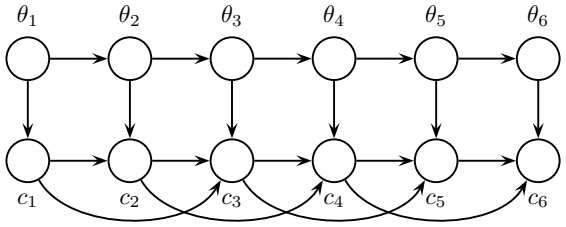


Figure 1: graphical model for a simple autoregressive HMM

We call the linear summarizers *windows*, with *window coefficients* w_k^d. These window coefficients are only non-zero in the past ($k < 0$). An example of autoregressive window coefficients is shown in Table 1(a).

By setting the windows to be $w_k^d = \delta_k^{-d}$ as in Table 1(b) we recover a canonical autoregressive HMM [4, 5, 7]. By setting the windows to be delta functions at fixed offsets from the current time as in Table 1(c) we obtain the form of model used by Woodland [6] and Chin [8].

Note that we only explicitly deal with the *static* feature vector sequence c for the autoregressive HMM. However the role played by linear summarizers here is somewhat similar to that of *dynamic features* in the standard HMM framework. For comparison standard HMM synthesis windows are shown in Table 1(d).

The graphical model for the autoregressive HMM with linear summarizers of depth $K = 2$ is shown in Figure 1.

2.2 Parameter estimation

In this section we cover maximum likelihood estimation of the parameters $(a^d_{qi}, \mu^d_{qi}, \mu^0_{qi}, \sigma^2_{qi})$ of the autoregressive HMM. We use *expectation maximization (EM)* [11] for parameter re-estimation, and show how to use decision tree clustering to cope with data sparsity.

Nothing in this section depends on the summarizers being linear. Indeed the training procedure needs to know only the *value* $f^d(c_{1:t-1})$ of each summarizer at each time t, which we write more concisely as $f^d(t)$ or f^d_t.

2.2.1 Forward-Backward algorithm

Define:

$$\alpha_q(t) \triangleq P(c_{1:t}, \theta_t = q)$$
$$\beta_q(t) \triangleq P(c_{t+1:T}|c_{1:t}, \theta_t = q)$$

Then we have the following recursions:

$$\alpha_q(t) = \sum_p \alpha_p(t-1) u_{pq} P(c_t|c_{1:t-1}, \theta_t = q) \tag{7}$$

$$\beta_q(t) = \sum_r u_{qr} P(c_{t+1}|c_{1:t}, \theta_{t+1} = r) \beta_r(t+1) \tag{8}$$

where $u_{pq} \triangleq P(\theta_t = q|\theta_{t-1} = p)$ is the state transition probability. This allows us to efficiently compute α and β, and thus the *state occupancies* $\gamma_q(t) \triangleq P(\theta_t = q|c)$:

$$\gamma_q(t) = \frac{\alpha_q(t)\beta_q(t)}{\sum_q \alpha_q(t)\beta_q(t)}$$

The above is very general, and would work for any output distributions of the form $P(c_t|c_{1:t-1}, \theta_t)$. In particular it does not depend on the specific form of output distribution given in (2) and (3).

Note the similarity to Forward-Backward for the standard HMM framework. We can use the same implementation as for the standard case, replacing only the state output probability calculation.

We can extend the above algorithm to the case of a *hidden semi-Markov model (HSMM)* with an explicit duration model by using an expanded state space, as for the standard HMM synthesis framework [12, 13, 14].

2.2.2 Parameter re-estimation overview

In this section we give an overview of the parameter re-estimation procedure for the autoregressive HMM. Technical details and proofs are given in §2.2.3.

We use the notation:

$$\langle g \rangle_q \triangleq \frac{\sum_t \gamma_q(t) g(t)}{\sum_t \gamma_q(t)}$$

to denote the weighted average of an arbitrary real-valued function $g(t)$ with respect to the occupancies $\gamma_q(t)$ of state q. To efficiently compute the *statistic* $\langle g \rangle_q$ we first *accumulate* $\sum_t \gamma_q(t) g(t)$ then divide by the state occupancy $\sum_t \gamma_q(t)$.

The re-estimation formulae giving the updated parameter values $(\widehat{a}^d_{qi}, \widehat{\mu}^d_{qi}, \widehat{\mu}^0_{qi}, \widehat{\sigma}^2_{qi})$ are:

$$\widehat{\mu}^0_{qi} = \langle c_i \rangle_q \tag{9}$$

$$\widehat{\mu}^d_{qi} = \langle f^d_i \rangle_q \tag{10}$$

$$\sum_{e=1}^{D} R^{de}_{qi} \widehat{a}^e_{qi} = r^d_{qi} \tag{11}$$

$$\widehat{\sigma}^2_{qi} = r^0_{qi} - \sum_{d=1}^{D} \widehat{a}^d_{qi} r^d_{qi} \tag{12}$$

where

$$R^{de}_{qi} \triangleq \langle f^d_i f^e_i \rangle_q - \langle f^d_i \rangle_q \langle f^e_i \rangle_q$$
$$r^d_{qi} \triangleq \langle c_i f^d_i \rangle_q - \langle c_i \rangle_q \langle f^d_i \rangle_q$$
$$r^0_{qi} \triangleq \langle c_i c_i \rangle_q - \langle c_i \rangle_q \langle c_i \rangle_q$$

and where q ranges over states, i ranges over feature vector components, and $1 \leq d, e \leq D$.

If $R^{(1:D)(1:D)}_{qi}$ is not invertible then (11) has multiple equally-good solutions and we may pick arbitrarily between them, for instance by using the Moore-Penrose pseudo-inverse $\widehat{a}_{qi} = R^{+}_{qi} r_{qi}$, which can be efficiently computed by singular value decomposition. We use variance floors on σ^2_{qi} to avoid singularities in the likelihood function.

Note that computing the $(\widehat{a}^d_{qi})$ using (11) involves inverting a $D \times D$ matrix for each q and i. For typical cases with $D = 3$ or $D = 4$ summarizers this is not computationally intensive.

We need to accumulate various quantities for the re-estimation formulae. Defining a dummy summarizer $f^0(t) \triangleq c_t$, we need the statistics $\langle f^d_i \rangle_q$ and $\langle f^d_i f^e_i \rangle_q$ with corresponding accumulators:

$$\sum_t \gamma_q(t) f^d_i(t) \tag{13}$$

$$\sum_t \gamma_q(t) f^d_i(t) f^e_i(t) \tag{14}$$

where q ranges over states, i ranges over feature vector components, and $0 \leq d, e \leq D$.

2.2.3 Details of parameter re-estimation

For clarity, in this section we will leave the index i implicit, so c is a sequence of scalars, μ_q is a scalar, etc.

From (1) and (4):

$$\begin{aligned} \log P(c|\theta) &= \sum_t \log P(c_t | c_{1:t-1}, \theta_t) \\ &= -\tfrac{T}{2} \log 2\pi + \tfrac{1}{2} \sum_t \log \tau_{\theta_t} - \tfrac{1}{2} \sum_t \tau_{\theta_t} \left(c_t - \mu_{\theta_t}(c_{1:t-1}) \right)^2 \end{aligned}$$

where $\tau_q \triangleq 1/\sigma_q^2$. Adding a dummy summarizer $f_t^0 \triangleq c_t$ and setting $a_q^0 = -1$ to simplify writing subsequent expressions, from (5) we can write:

$$\log P(c|\theta) = C + \tfrac{1}{2}\sum_t \log \tau_{\theta_t} - \tfrac{1}{2}\sum_t \sum_{d,e=0}^{D} \tau_{\theta_t} a_{\theta_t}^d a_{\theta_t}^e (f_t^d - \mu_{\theta_t}^d)(f_t^e - \mu_{\theta_t}^e)$$

where $C = -\frac{T}{2}\log 2\pi$.

For expectation maximization we want to consider the expectation of $\log P(c|\theta)$ with respect to some distribution Q over θ (typically the posterior over θ given the current model parameters). We get the following EM auxiliary function:

$$\begin{aligned}
\mathbb{E}_Q \log P(c|\theta) &= \sum_\theta Q(\theta) \log P(c|\theta) \\
&= C + \tfrac{1}{2}\sum_{t,q} \gamma_q(t) \log \tau_q - \tfrac{1}{2}\sum_{t,q}\sum_{d,e=0}^{D} \gamma_q(t)\tau_q a_q^d a_q^e (f_t^d - \mu_q^d)(f_t^e - \mu_q^e) \\
&= C + \tfrac{1}{2}\sum_q \gamma_q \log \tau_q - \tfrac{1}{2}\sum_q \gamma_q \tau_q \left[\sum_{d,e=0}^{D} R_q^{de} a_q^d a_q^e + \left(\sum_{d=0}^{D} a_q^d (\widehat{\mu}_q^d - \mu_q^d) \right)^2 \right]
\end{aligned}$$

where

$$\begin{aligned}
\gamma_q(t) &\triangleq Q(\{\theta_t = q\}) = \sum_{\theta:\theta_t=q} Q(\theta) \\
\gamma_q &\triangleq \sum_t \gamma_q(t) \\
R_q^{de} &\triangleq \left\langle f^d f^e \right\rangle_q - \left\langle f^d \right\rangle_q \left\langle f^e \right\rangle_q \\
\widehat{\mu}_q^d &\triangleq \left\langle f^d \right\rangle_q
\end{aligned}$$

We can see that maximizing $\mathbb{E}_Q \log P(c|\theta)$ with respect to the parameters $(\mu_q^{0:D}, a_q^{1:D}, \tau_q)$ is equivalent to maximizing the following expression separately for each q with $\gamma_q \neq 0$:

$$\log \tau_q - \tau_q \left[\sum_{d,e=0}^{D} R_q^{de} a_q^d a_q^e + \left(\sum_{d=0}^{D} a_q^d (\widehat{\mu}_q^d - \mu_q^d) \right)^2 \right] \tag{15}$$

If $\gamma_q = 0$ then the parameters for that state q don't affect the likelihood, and we choose them to keep their previous values.

Considering (15) as a function of $\mu_q^{0:D}$ given $a_q^{1:D}$ and τ_q, we can see that this achieves its maximum whenever $\sum_{d=0}^{D} a_q^d(\widehat{\mu}_q^d - \mu_q^d) = 0$ for all q, and in particular when $\mu_q^d = \widehat{\mu}_q^d$ for all d and q.

The fact the maximum in $\mu_q^{0:D}$ isn't unique is due to the redundant means $\mu_q^{1:D}$ we introduced in §2.1. We can see now how this trick makes re-estimation easier. Without loss of generality we could set $\mu_q^{1:D} = 0$, but then the maximum value of μ_q^0 would depend on $a_q^{1:D}$ and the maximum value of $a_q^{1:D}$ already depends on μ_q^0, so we would need to jointly re-estimate them. This could be done without too much trouble, for example by treating the mean μ_q^0 as the coefficient a_q^{const} of a constant summarizer $f^{\text{const}}(c_{1:t-1}) \triangleq 1$. However using the above trick is arguably easier, and saves us a dimension on the matrix inversion.

In fact the above trick is exactly what results from treating the mean as a constant summarizer and using the block matrix inversion lemma to reduce the dimensionality of the required inverse by 1.

Plugging $\sum_{d=0}^{D} a_q^d(\widehat{\mu}_q^d - \mu_q^d) = 0$ into (15), we now want to maximize the following expression with respect to $a_q^{1:D}$ given τ_q, subject to $a_q^0 = -1$:

$$\log \tau_q - \tau_q \sum_{d,e=0}^{D} R_q^{de} a_q^d a_q^e \tag{16}$$

This is equivalent to minimizing the following expression with respect to $a_q^{1:D}$:

$$\sum_{d,e=0}^{D} R_q^{de} a_q^d a_q^e = \sum_{d,e=1}^{D} R_q^{de} a_q^d a_q^e - 2\sum_{d=1}^{D} r_q^d a_q^d + r_q^0 \tag{17}$$

where

$$\begin{aligned} r_q^d \triangleq R_q^{d0} &= \left\langle cf^d \right\rangle_q - \left\langle c \right\rangle_q \left\langle f^d \right\rangle_q \\ r_q^0 \triangleq R_q^{00} &= \left\langle cc \right\rangle_q - \left\langle c \right\rangle_q \left\langle c \right\rangle_q \end{aligned}$$

The derivative of (17) with respect to $a_q^{1:D}$ is 0 if and only if $R_q^{(1:D)(1:D)} a_q^{1:D} = r_q^{1:D}$. However it is not immediately obvious that this equation always has a solution, or that any solution is a global minimum.

To see this, note that $R_q^{(0:D)(0:D)}$ is positive semi-definite since it is a convex combination of positive semi-definite matrices, and so $\sum_{d,e=0}^{D} R_q^{de} a_q^d a_q^e$ is a positive semi-definite quadratic form in $a_q^{0:D}$. It can be shown that any affine map from $\mathbb{R}^n$ to $\mathbb{R}^m$ induces a pullback taking positive semi-definite quadratic forms on $\mathbb{R}^m$ to translated positive semi-definite quadratic forms on $\mathbb{R}^n$, where a *translated* quadratic form is anything of the form $Q'(x) = Q(x-\mu)+c$ for some quadratic form Q. Therefore (17) is a translated positive semi-definite quadratic form in $a_q^{1:D}$. In particular this means (17) is bounded below, that the global minimum is attained somewhere, and that any point where the derivative is zero is a global minimum.

Therefore there is guaranteed to be a solution to $R_q^{(1:D)(1:D)} \widehat{a}_q^{1:D} = r_q^{1:D}$, or more concisely $R_q \widehat{a}_q = r_q$, and any such $\widehat{a}_q$ is a global minimum. For instance we may choose the solution with minimum Euclidean norm, corresponding to the Moore-Penrose pseudo-inverse $\widehat{a}_q = R_q^+ r_q$. In the common case where R_q is invertible the unique minimum is at $\widehat{a}_q = {R_q}^{-1} r_q$. The global minimum value of (17) is $r_q^0 - \sum_{d=1}^{D} r_q^d \widehat{a}_q^d$. This value is independent of the minimum $\widehat{a}_q$ chosen, and is non-negative since (17) is non-negative.

Note also that since (17) is bounded below, (15) cannot be made arbitrarily large for fixed τ_q. Thus there is no need to use flooring or other techniques to avoid singularities in the likelihood function when choosing $\widehat{a}_q^{1:D}$.

Finally, plugging an optimal value $\widehat{a}_q^{1:D}$ into (16), we want to maximize the following expression with respect to τ_q:

$$\log \tau_q - \tau_q \widehat{\sigma}_q^2 \tag{18}$$

where

$$\widehat{\sigma}_q^2 \triangleq r_q^0 - \sum_{d=1}^{D} r_q^d a_q^d$$

We saw above that $\widehat{\sigma}_q^2 \geq 0$. If $\widehat{\sigma}_q^2 > 0$ then the unique maximum in τ_q is at $1/\tau_q = \widehat{\sigma}_q^2$, that is $\sigma_q^2 = \widehat{\sigma}_q^2$.

If $\widehat{\sigma}_q^2 = 0$ then we can make the likelihood arbitrarily large by choosing small σ_q^2. Singularities like this in the likelihood function are a weakness of maximum likelihood estimation in general. A standard hack to get around this in the case of variances is to set a minimum value for variances, called a *variance floor* [15]. If $\widehat{\sigma}_q^2$ is less than the variance floor then σ_q^2 is set to the floor. For the standard HMM framework the variance floor for each feature vector component is often set to a constant fraction of the global variance of that component over the whole training corpus, and this is the approach we adopt here.

There are three special cases above: $\gamma_q = 0$, R_q not invertible and $\widehat{\sigma}_q^2 = 0$. We might ask when these occur. A key quantity is the non-zero occupancy number $N_q \triangleq \#\{t : \gamma_q(t) > 0\}$. Note that $N_q = 0$ if and only if $\gamma_q = 0$. If $N_q < D + 1$ then R_q is always non-invertible and $\widehat{\sigma}_q^2$ is typically 0. If $N_q = D + 1$ then R_q is typically invertible and $\widehat{\sigma}_q^2$ is typically 0. If $N_q > D + 1$ then R_q is typically invertible and $\widehat{\sigma}_q^2$ is typically non-zero. Note that $N_q \geq \gamma_q$ and N_q may be much larger if many times contribute small occupancies, so EM's soft assignment helps to avoid these special cases compared with hard assignment. Intuitively we can say that these special cases tend to occur only for states which definitely aren't seen much in the training corpus.

The maximum value of the auxiliary function is:

$$\mathbb{E}_Q \log P(c|\theta) = C + \tfrac{1}{2} \sum_q \gamma_q \left(-\log \widehat{\sigma}_q^2 - 1 \right) \tag{19}$$

$$= -\tfrac{T}{2} \left(\log 2\pi + 1 \right) - \tfrac{1}{2} \sum_q \gamma_q \log \widehat{\sigma}_q^2 \tag{20}$$

since $\sum_q \gamma_q = T$.

2.2.4 Decision tree clustering

In systems with large state spaces there is typically not enough data to robustly ML-estimate the output distributions for every state – indeed the majority of states may never be seen in the training corpus. A standard solution is to *cluster* the states and use one set of shared parameters for all the states in each cluster, and *decision tree clustering* [16] is a standard clustering method. In this section we show how to do decision tree clustering for the autoregressive HMM.

A clustering is represented by a partition $\mathcal{C}$ of state space, where we constrain the states $q \in C$ in a cluster $C \in \mathcal{C}$ to have identical output distributions. Given state-level accumulators (13) and (14) for an unclustered system we can compute accumulators for an arbitrary state-clustered system just by summing, for each cluster, the state-level accumulators for that cluster. Thus we can compute the maximum likelihood before and after a hypothesized split. From (22) the change in likelihood for a split of cluster C into two pieces C_1 and C_2 is:

$$\Delta(C \to C_1, C_2) \triangleq \tfrac{1}{2} \gamma_C \sum_i \log \widehat{\sigma}_{Ci}^2 - \tfrac{1}{2} \gamma_{C_1} \sum_i \log \widehat{\sigma}_{C_1 i}^2 - \tfrac{1}{2} \gamma_{C_2} \sum_i \log \widehat{\sigma}_{C_2 i}^2 \tag{21}$$

Note that this depends only on C, C_1 and C_2 and not on other details of the clustering $\mathcal{C}$.

To compute the change in likelihood for a hypothesized split we therefore need to sum the accumulators for all the states in C_1 and use these to compute the optimal values $(\widehat{\sigma}_{C_1 i}^2)$,

and then do the same for C_2 (or slightly more efficiently the accumulators for C_2 can be computed directly from the accumulators for C and C_1 by subtraction).

Thus the decision tree clustering procedure is as follows. We perform Forward-Backward with an unclustered system and compute state-level accumulators (13) and (14). Starting with all states in one cluster, we recursively split each leaf node according to the allowed question that maximizes the change in likelihood. We don't split a node when the maximum change in likelihood obtainable by splitting falls below a pre-specified threshold, or when there are no allowed questions. We have a pre-specified minimum occupancy for a cluster, and an allowed question is one that doesn't violate this minimum occupancy constraint. Note that the order in which we choose nodes to split doesn't affect the outcome.

Note that for the autoregressive HMM the updated parameter values $(\widehat{a}^d_{qi}, \widehat{\mu}^d_{qi}, \widehat{\mu}^0_{qi}, \widehat{\sigma}^2_{qi})$ together with state occupancies (γ_q) are *not* sufficient to recover the accumulators (13) and (14). This is in contrast to the standard HMM framework where there is a simple correspondence between re-estimated parameter values and accumulators (given the state occupancies). Furthermore we believe there is no way to deduce $\widehat{\sigma}^2_{Ci}$ from the re-estimated parameter values for each $q \in C$. This means we must pass the decision tree clustering algorithm the accumulators themselves, and not just the re-estimated parameter values together with occupancies as for the standard HMM.

Given a clustering constructed as above we can easily adapt the procedure in §2.2.2 to re-estimate the shared parameters. For each cluster C we just use a shared accumulator for all the states in the cluster, then re-estimate the shared set of parameters $(\widehat{a}^d_{Ci}, \widehat{\mu}^d_{Ci}, \widehat{\mu}^0_{Ci}, \widehat{\sigma}^2_{Ci})$ using the same procedure as in §2.2.2. The maximum likelihood (20) becomes:

$$\mathbb{E}_Q \log P(c_i|\theta) = -\tfrac{T}{2}\left(\log 2\pi + 1\right) - \tfrac{1}{2}\sum_{C\in\mathcal{C}} \gamma_C \log \widehat{\sigma}^2_{Ci} \tag{22}$$

2.3 $P(c|\theta)$ is Gaussian

For the autoregressive HMM with linear summarizers, the distribution $P(c|\theta)$ over output sequences c given a state sequence θ is a multidimensional Gaussian. In this section we show this, and derive an explicit form for the mean and precision matrix in terms of the state sequence.

As elsewhere we assume the feature vector sequence components are independent given the state sequence, though in fact $P(c|\theta)$ is still Gaussian in the general vector autoregressive HMM with linear summarizers case. Thus we want to show $P(c_i|\theta)$ is a multidimensional Gaussian for each static sequence component i. For clarity, in this section we will leave the index i implicit, so c is a sequence of scalars, μ_q is a scalar, etc.

From (1) and (4):

$$\begin{aligned}
\log P(c|\theta) &= \sum_t \log P(c_t|c_{1:t-1}, \theta_t) \\
&= -\tfrac{T}{2}\log 2\pi - \tfrac{1}{2}\sum_t \log(\sigma^2_{\theta_t}) - \tfrac{1}{2}\sum_t \left(\frac{c_t - \mu_{\theta_t}(c_{1:t-1})}{\sigma_{\theta_t}}\right)^2 \\
&\stackrel{c}{=} -\tfrac{1}{2}\sum_t z_t^2
\end{aligned}$$

where the constant does not depend on c and the *residual* z_t is defined as:

$$z_t \triangleq \frac{c_t - \mu_{\theta_t}(c_{1:t-1})}{\sigma_{\theta_t}}$$

Now μ_q is an affine function of the summarizers as in (4), and the summarizers are linear functions of the past output $c_{1:t-1}$ as in (6). Therefore z_t is an affine combination of $c_{1:t}$, that is:

$$z_t = \sum_s L_{ts} c_s - b_t$$

for some sequence b_t and lower triangular matrix L_{ts}. In fact:

$$L_{ts} = l^{s-t}_{\theta_t}$$
$$b_t = b_{\theta_t}$$

where

$$l^s_q \triangleq \frac{-1}{\sigma_q} \sum_{d=0}^{D} a^d_q w^d_s$$
$$b_q \triangleq \frac{-1}{\sigma_q} \sum_{d=0}^{D} a^d_q \mu^d_q$$

and where we have introduced a dummy summarizer $f^0_t \triangleq c_t$ with window coefficients $w^0_k = \delta^0_k$ and autoregressive coefficient $a^0_q = -1$ to simplify writing the expressions. Considering the sequences c, z and b as vectors over time, we have $z = Lc - b$. Therefore:

$$\sum_t z^2_t = \sum_t z^\mathsf{T} z = c^\mathsf{T}(L^\mathsf{T} L)c - 2c^\mathsf{T}(L^\mathsf{T} b) + b^\mathsf{T} b$$

We can see that $P(c|\theta)$ is Gaussian, with natural parameters $\overline{P} = L^\mathsf{T} L$ and $\overline{b} = L^\mathsf{T} b$. By definition the mean of the Gaussian $\overline{\mu}$ satisfies $\overline{P}\overline{\mu} = \overline{b}$ so $L^\mathsf{T} L\overline{\mu} = L^\mathsf{T} b$. But L is invertible, since it is lower triangular with diagonal elements $(1/\sigma_{\theta_t})$, and so we get $L\overline{\mu} = b$.

Note that L is a Cholesky-like decomposition of the precision matrix $\overline{P}$, the only difference being that the conventional Cholesky decomposition results in a lower triangular matrix times its transpose, rather than the transpose times a lower triangular matrix. Furthermore, since L is band diagonal, so is $\overline{P}$. If K is the maximum window depth as in (6) then we can see that L is $(K+1)$-diagonal, so $\overline{P}$ is $(2K+1)$-diagonal.

We note in passing that the residual sequence $z|\theta \sim \mathcal{N}(0, I)$, since $c|\theta \sim \mathcal{N}(\overline{\mu}, \overline{\Sigma})$, so $(Lc - b)|\theta \sim \mathcal{N}(L\overline{\mu} - b, L\overline{\Sigma}L^\mathsf{T})$, which is just $\mathcal{N}(0, I)$.

The above leads to a nice decomposition of the overall precision matrix $\overline{P}$ as the sum over time of *local contributions* that depend only on the state at that time. Similarly the overall b-value $\overline{b}$ decomposes as the sum over time of local contributions. Specifically, if we define:

$$\overline{P}^{st}_q \triangleq l^s_q l^t_q = \tau_q \left(\sum_{d=0}^{D} a^d_q w^d_s \right) \left(\sum_{d=0}^{D} a^d_q w^d_t \right) \tag{23}$$

$$\overline{b}^s_q \triangleq l^s_q b_q = \tau_q \left(\sum_{d=0}^{D} a^d_q w^d_s \right) \left(\sum_{d=0}^{D} a^d_q \mu^d_q \right) \tag{24}$$

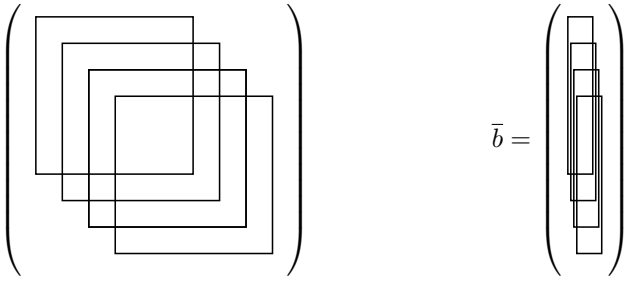
then

$$\overline{P}^{st} = \sum_u \overline{P}_{\theta_u}^{(s-u)(t-u)} \tag{25}$$

$$\overline{b}^{s} = \sum_u \overline{b}_{\theta_u}^{(s-u)} \tag{26}$$

Thus the overall precision matrix $\overline{P}$ is the sum of overlapping local contributions $\overline{P}_{\theta_u}$ for each time u, where each local contribution is a $(K+1)\times(K+1)$ matrix which depends only on the state θ_u at that time. Similarly the overall b-value $\overline{b}$ is the sum of overlapping local contributions $\overline{b}_{\theta_u}$ for each time u, where each local contribution is a $(K+1)$-dimensional vector which depends only on the state θ_u at that time. These decompositions into local contributions given by (25) and (26) are shown schematically below:

$$\overline{P} = \quad \overline{b} = \tag{27}$$



We can see from (23) that each local contribution $l_q l_q^{\mathsf{T}}$ to the precision matrix is the outer product of a $(K+1)$-dimensional vector with itself, where this vector is a state-dependent linear combination of window coefficients.

2.4 Synthesis

During synthesis we produce an output feature sequence c for a given word sequence. From the point of view of synthesis there is a strong similarity between the the standard HMM synthesis framework, the trajectory HMM and the autoregressive HMM with linear summarizers. As we show in §2.3 and §3.4, in all three cases $P(c|\theta)$ is a multidimensional Gaussian over vector sequences with band diagonal precision matrix. This is the only fact that many existing algorithms rely on to do efficient synthesis. Therefore to use the autoregressive HMM with existing synthesis algorithms we only need to be able to compute the parameters of the Gaussian $P(c|\theta)$, and this can be done efficiently using (25) and (26). In this section we show explicitly how to adapt two current synthesis algorithms for the autoregressive HMM.

It is common to use synthesis methods that first choose the state sequence θ based on $P(\theta)$ and then choose an output sequence c given this state sequence based on $P(c|\theta)$. However we may also choose the state sequence and output sequence jointly based on $P(c,\theta)$. For example *EM parameter generation* (case 2 in [17]) iteratively maximizes $P(c)$ using expectation maximization (or more precisely it can be interpreted in this way for an appropriate model $P(c,\theta)$). Although we don't consider this method below the autoregressive HMM is well suited to this approach and it is easy to adapt EM generation for the autoregressive HMM.

2.4.1 Synthesis using dynamic features

For synthesis using dynamic features ([2] and case 1 in [17]), we first choose a state sequence θ and then choose the feature sequence c which maximizes $P(c|\theta)$.

For both the autoregressive HMM with linear summarizers (§2.3) and the standard HMM synthesis framework (§3.4) $P(c|\theta)$ is a multidimensional Gaussian so the maximum value is at its mean $\overline{\mu} \triangleq \mathbb{E}[c|\theta]$. The natural parameters of this Gaussian can be computed efficiently using (25) and (26) for the autoregressive HMM or (40) and (41) for the standard HMM synthesis framework, and from these we can compute the mean $\overline{\mu}$ by solving $\overline{P}\overline{\mu} = \overline{b}$, which can be done using Cholesky decomposition $\overline{P} = LL^\mathsf{T}$ followed by forward substitution $Ly = \overline{b}$ and backward substitution $L^\mathsf{T}\overline{\mu} = y$. This is efficient since $\overline{P}$ is band diagonal.

In fact there is an even more straightforward way to compute $\overline{\mu}$ for the autoregressive HMM with linear summarizers. We have:

$$\begin{aligned}\mathbb{E}[c_t|\theta] &= \mathbb{E}\left[\mathbb{E}[c_t|c_{1:t-1}, \theta]\right] \\ &= \mathbb{E}[\mu_{\theta_t}(c_{1:t-1})] \\ &= \mu_{\theta_t}(\mathbb{E}c_{1:t-1})\end{aligned}$$

where we have used the general fact $\mathbb{E}X = \mathbb{E}[\mathbb{E}[X|Y]]$ and the fact that for linear summarizers the mean functions $\mu_q(c_{1:t-1})$ in (3) are affine-linear. Therefore the mean vector sequence $\overline{\mu}$ can be computed efficiently by a one-pass forward recursion over time:

$$\overline{\mu}_t = \mu_{\theta_t}(\overline{\mu}_{1:t-1}) \tag{28}$$

These two methods are in fact closely related. In §2.3 we mention that for the autoregressive HMM we are given a Cholesky-like decomposition $\overline{P} = L^\mathsf{T}L$ essentially for free. And above we could equally have solved $\overline{P}\overline{\mu} = \overline{b}$ using the Cholesky-like decomposition $\overline{P} = L^\mathsf{T}L$ instead of the standard Cholesky decomposition $\overline{P} = LL^\mathsf{T}$. If we use the Cholesky-like decomposition to solve for $\overline{\mu}$ by doing a backward then a forward substitution, then (28) is precisely the last forward substitution $L\overline{\mu} = b$, and we have effectively solved the backward substitution $L^\mathsf{T}b = \overline{b}$ analytically beforehand.

2.4.2 Synthesis considering global variance

Standard techniques in HMM synthesis, such as synthesis using dynamic features, are found to produce utterances that sound "flat" or "dull" [18]. In particular, it is found that synthesized utterances tend to have far less *global variance (GV)* than natural ones, where the global variance $v(c_i)$ of the i^{th} component of the feature vector sequence is defined as [18]:

$$v(c_i) \triangleq \frac{1}{T}\sum_t c_{ti}^2 - \left(\frac{1}{T}\sum_t c_{ti}\right)^2$$

Toda [18] introduced parameter generation *considering global variance* as a way to alleviate this lack of global variance, while using existing models. The distribution of global variances observed in training utterances is modelled by a Gaussian, typically treating each component of the feature vector as independent. The HMM and GV parameters are trained

independently of each other. During synthesis, we use some form of gradient descent to optimize a cost function that is a weighted sum of the HMM log probability of the output sequence and the GV log probability of the output sequence (keeping the state sequence fixed). This procedure is found to dramatically improve the naturalness of synthetic speech [18].

It is trivial to adapt this for use with the autoregressive HMM with linear summarizers. Since we keep the state sequence fixed during gradient descent, the HMM log probability is in both cases just a multidimensional Gaussian. Therefore we can do parameter generation considering global variance for the autoregressive HMM simply by passing the appropriate multidimensional Gaussian to the GV generation algorithm.

2.5 Other considerations

2.5.1 End effects

For the autoregressive HMM with linear summarizers given by (6) the value of the summarizer f^d at time t depends on the output at previous times $c_{t-K:t-1}$. For the initial frames where $t \leq K$ this means we have a dependence on the output at times $t \leq 0$. However these outputs are not specified in our training corpus and we do not want to have to specify them for every utterance we synthesize.

Here we take the simple approach of arbitrarily assuming $c_t = 0$ for $t \leq 0$ when computing the summarizers at time $t \leq K$. This is expected to have a small overall effect as long as our training corpus has few very short utterances and we are not trying to synthesize very short utterances, where 'very short' means tens of frames for the standard windows. In practice these conditions are usually satisfied.

Similar *end effect* issues occur in the standard HMM synthesis framework when computing delta coefficients for the initial and final frames of an utterance. The delta coefficient at time T depends on the static coefficient at time $T + 1$, for example.

For the autoregressive HMM we do not have to worry about times after the end of the utterance, since it is trivial to analytically marginalize over all possible future state and output sequences $(\theta_{T+1:\infty}, c_{T+1:\infty})$.

2.5.2 Equivalent summarizer sets

There is a systematic redundancy in the specification of sets of summarizers for the autoregressive HMM. We define a notion of *equivalence* for summarizer sets below. Two systems with equivalent summarizer sets trained on the same data produce an identical model $P(c, \theta)$ and so identical synthesized utterances. For example the summarizers defined by Table 1(a) and Table 1(b) are equivalent but neither is equivalent to Table 1(c). This redundancy is worth taking into account when choosing summarizer sets.

The basic idea is as follows. The model $P(c, \theta)$ depends only on the summarizers (f_i^d) and parameters $(a_{qi}^d, \mu_{qi}^d, \mu_{qi}^0)$ through (5). If we change the set of summarizers in a particular way and change the parameters accordingly we can end up with the same mean functions $(\mu_{qi}(c_{1:t-1}))$ and so the same model $P(c, \theta)$.

More precisely, consider the vector space $\mathcal{F}$ of functions from past output $c_{1:t-1}$ to $\mathbb{R}$. Each summarizer f_i^d is in $\mathcal{F}$. Let $F \subset \mathcal{F}$ be the set of summarizers (f_i^d) and define $F_i \triangleq \left\{f_i^d : d\right\}$. From (5) we can see that the set of possible mean functions for feature vector component i is just the linear span $\left\langle F_i, \overline{1}\right\rangle$ where $\overline{1}$ is the constant summarizer $\overline{1}(c_{1:t-1}) \triangleq 1$. We call two sets of summarizers F and $\widetilde{F}$ *equivalent* if they have the same span $\left\langle F_i\right\rangle = \left\langle \widetilde{F}_i\right\rangle$ for each i. In this case the two sets of summarizers have the same set of possible mean functions $\mu_{qi}(c_{1:t-1})$ for each i, and so the same set of possible output distributions $P(c_t|c_{1:t-1}, \theta_t)$, and so the same set of possible generative models $P(c, \theta)$. Therefore given any set of model parameters $\lambda \triangleq (a_{qi}^d, \mu_{qi}^d, \mu_{qi}^0, \sigma_{qi}^2)$ for F we can find a set of model parameters $\widetilde{\lambda}$ for $\widetilde{F}$ such that:

$$P(c, \theta|\lambda, F) = P(c, \theta|\widetilde{\lambda}, \widetilde{F}) \quad \forall c, \theta \tag{29}$$

We call the parameter sets λ and $\widetilde{\lambda}$ *equivalent* if (29) is satisfied.

We will assume that for a given summarizer set F the output distributions uniquely determine the parameter set, i.e. no two parameter sets for F are equivalent. This is typically the case unless we have redundant summarizers that are a linear combination of other summarizers, and even in this case the following analysis holds with equivalence classes of parameter sets instead of parameter sets. By the above we have a bijection ϕ from parameter sets for F to parameter sets for $\widetilde{F}$:

$$\phi : \lambda \stackrel{\cong}{\longmapsto} \widetilde{\lambda} \tag{30}$$

Synthesis procedures such as synthesis using dynamic features and synthesis considering global variance generate identical output for equivalent parameter sets. This is because they only depend on the overall generative model $P(c, \theta)$ and by (29) this is identical for equivalent parameter sets.

Parameter estimation procedures such as expectation maximization and decision tree clustering respect this bijection, in the sense that applying ϕ then doing training gives the same result as doing training then applying ϕ. The root of this property is the fact that these are both likelihood-based methods, and a reparameterization of parameter space transforms the likelihood simply as a function – in particular the maximum after reparameterization is just the transform of the maximum before reparameterization. This equivalence could therefore be expected to hold for any likelihood-based training procedure. Here we show it explicitly for EM and decision tree clustering.

The full EM auxiliary function is:

$$A(\lambda) \triangleq \sum_{\theta} Q(\theta) \log P(c, \theta|\lambda, F) \tag{31}$$

where $Q(\theta) \triangleq P(\theta|c, \lambda_{\text{prev}}, F)$ where λ_{prev} is the parameter set from the previous iteration. Suppose F and $\widetilde{F}$ are equivalent sets of summarizers and consider applying an EM iteration to both λ_{prev} for F and $\widetilde{\lambda}_{\text{prev}} \triangleq \phi(\lambda_{\text{prev}})$ for $\widetilde{F}$. Let $\widetilde{Q}$ and $\widetilde{A}$ be the quantities for $\widetilde{F}$ corresponding to Q and A for F. By (29) $Q(\theta) = \widetilde{Q}(\theta)$ since λ_{prev} and $\phi(\lambda_{\text{prev}})$ are equivalent. Furthermore $A(\lambda) = \widetilde{A}(\phi(\lambda))$ since λ and $\phi(\lambda)$ are equivalent. Therefore the range of possible values of A and $\widetilde{A}$ is the same. If there is a unique parameter set $\widehat{\lambda}$ that attains the maximum of A then $\phi(\widehat{\lambda})$ is the unique parameter set that attains the maximum of $\widetilde{A}$, that is EM respects ϕ.

There is a slight subtlety in the case where there are multiple values of λ which attain the maximum of A, and the above claim does not hold precisely. For example in §2.2.3 the

pseudo-inverse was suggested as a way to choose between the multiple values. However in general EM-with-pseudo-inverse does not respect the isomorphism ϕ, since ϕ does not respect the Euclidean norm used to define the pseudo-inverse.

For decision tree clustering the situation is similar. Suppose F and $\widetilde{F}$ are equivalent sets of summarizers and consider applying decision tree clustering to both λ_{prev} for F and $\widetilde{\lambda}_{\text{prev}} \triangleq \phi(\lambda_{\text{prev}})$ for $\widetilde{F}$. By (29) the computed occupancies $(\gamma_q(t))$ are identical. Moreover given a clustering $\mathcal{C}$ the optimal variance $\widehat{\sigma}^2_{Ci}$ given by (12) is the same in both cases, since this is just the result of applying EM for the clustering $\mathcal{C}$. Now suppose during decision tree construction that the clustering so far is the same in both cases. Then the occupancies and optimal variance before and after a potential split will be the same in both cases, so by (21) the change in likelihood for the split will be the same, and so the same split will be selected. Therefore identical decision trees will be constructed. The parameter sets for the final clustering will also be equivalent, subject to the same proviso as above regarding uniqueness of the EM maximum, since each is the result of applying EM for this clustering. Thus decision tree clustering respects ϕ.

What about initialization? If two systems with different summarizer sets are initialized differently then they may remain different after re-estimation and decision tree clustering. This obviously depends on the details of initialization. One initialization scheme is to use *two model re-estimation* [15] where a monophone standard HMM is used to compute occupancies which are then used to re-estimate autoregressive output distributions. In this case $Q(\theta)$ will be identical, since it comes from the same standard HMM, and as we saw above this means the re-estimated parameter sets will be equivalent.

For linear summarizers each summarizer f^d_i is a linear combination of past output in the i^{th} feature vector component $c_{(1:t-1)i}$. Given a linear summarizer set F, let $V \triangleq \mathbb{R}^K$ where K is greater than or equal to the maximum window depth. Then each summarizer is a linear map from V to $\mathbb{R}$, that is $f^d_i \in V^*$. Since $V \cong V^*$ we can represent each summarizer as an element of V. In fact the components of the representation of f^d_i in terms of the canonical basis for $V = \mathbb{R}^K$ are precisely the window coefficients w^d_{ik} (where we normally assume the same windows are used for all i so this is just w^d_k). Therefore two window sets (w^d_k) and $(\widetilde{w}^d_k)$ define equivalent summarizers if and only if $\left\langle (w^d) \right\rangle = \left\langle (\widetilde{w}^d) \right\rangle$ where w^d is a vector with components w^d_k.

An example of equivalent window sets are Table 1(a) and Table 1(b). Neither is equivalent to Table 1(c). Viewing w^d_k as a matrix as laid out in these tables, window sets are equivalent if and only if their matrices have the same *row span*.

In light of this equivalence we can view our choice of summarizers not as choosing a *set of elements* of $\mathcal{F}$ so much as choosing a *linear subspace* of $\mathcal{F}$. For linear summarizers we are not choosing sets of window coefficients so much as a linear subspace of $\mathbb{R}^K$ for some depth K.

3 Standard HMM synthesis framework

In this section we review enough of the standard HMM synthesis framework and the trajectory HMM to allow us to compare them to the autoregressive HMM.

3.1 Model for parameter estimation

In the standard HMM framework the hidden state sequence $\theta = \theta_{1:T}$ and observed feature vector sequence $c = c_{1:T}$ are the same as for the autoregressive HMM. However in addition to the *static* feature vector sequence c, a set of D *dynamic* feature vector sequences (o^d) is computed from c. Each dynamic feature vector sequence o^d is computed using:

$$o_t^d \triangleq \sum_{k=-K_{\mathrm{L}}}^{K_{\mathrm{R}}} w_k^d c_{t+k} \tag{32}$$

Here we call (w_k^d) the *window coefficients* and we call K_{L}, K_{R} and $K \triangleq K_{\mathrm{L}} + K_{\mathrm{R}}$ respectively the *left-depth*, *right-depth* and *depth* of the set of windows. For notational convenience we define o_t^0 to be the static sequence c_t, i.e. $w_k^0 = \delta_k^0$. Standard HMM synthesis window coefficients with $D = 2$ dynamic windows of depth $K_{\mathrm{L}} = K_{\mathrm{R}} = 1$ and $K = 2$ are shown in Table 1(d). These window coefficients are chosen to approximately compute first and second derivatives. Together the dynamic feature vectors at time t form an *observation* $o_t \triangleq o_t^{0:D}$.

Note the similarity of these windows to those used for the autoregressive HMM with linear summarizers. However for the standard HMM there is no restriction to have $w_k^d = 0$ for $k \geq 0$.

During parameter estimation we ignore the relationship (32) between static and dynamic feature vector sequences. In fact we assume the observations o_t at successive times are independent given the state sequence. The joint probability distribution of the state sequence θ and observation sequence $o = o_{1:T}$ is of the form:

$$P(o, \theta) = \prod_t P(\theta_t|\theta_{t-1}) P(o_t|\theta_t) \tag{33}$$

and we model o_t as conditionally Gaussian, with state-dependent mean and covariance:

$$P(o_t|\theta_t) = \mathcal{N}\left(o_t|\mu_{\theta_t}, \Sigma_{\theta_t}\right) \tag{34}$$

We typically further assume diagonal covariance matrices $\Sigma_{qij}^{de} = 1/\tau_{qi}^d \delta_{ij} \delta^{de}$ and so:

$$P(o_t|\theta_t) = \prod_{d,i} \mathcal{N}\left(o_{ti}^d|\mu_{\theta_t i}^d, 1/\tau_{\theta_t i}^d\right) \tag{35}$$

The set of parameters specifying an HMM in the standard framework is therefore $(\mu_{qi}^d, \tau_{qi}^d)$, where μ_{qi}^d is the mean and τ_{qi}^d is the precision for state q, feature vector component i and window d.

This model allows efficient parameter estimation using expectation maximization [15].

3.2 Model for synthesis

Because it ignores the constraints (32) between static and dynamic features, most of the probability mass for the model in §3.1 is on observation sequences o that are *incoherent*, meaning that they don't respect these constraints. Synthesizing using that model will therefore typically output an incoherent observation sequence. If we throw away the dynamics

and just use the static feature vector sequence the resulting synthesized utterance sounds awful, since each frame was generated independently given the state sequence and so lacks realistic time dynamics. Clearly for synthesis we need adjust to the model in §3.1.

From another point of view, for synthesis we want a distribution over static feature vector sequences $P(c|\theta)$, but the parameter estimation model gives us a distribution over observation sequences $P_{\text{PE}}(o|\theta)$, and worse one that assigns probability zero to the set of coherent observation sequences, which are in fact the only ones possible!

One solution is to restrict the parameter estimation distribution $P_{\text{PE}}(o|\theta)$ to coherent observation sequences. We set:

$$P(c|\theta) \triangleq \frac{1}{Z_\theta} P_{\text{PE}}(W(c)|\theta) \tag{36}$$

where $W(c)$ is the observation sequence corresponding to static feature vector sequence c as computed by (32). The normalization constant Z_θ is required because not all observation sequences are coherent. We use joint distribution $P(c, \theta) = P(c|\theta)P(\theta)$ with $P(\theta) = P_{\text{PE}}(\theta)$ unchanged.

In §2.4.1 we reviewed how to do synthesis using dynamic features for this model – that is, how to find the output sequence c which maximizes $P(c|\theta)$. Historically the procedure to compute $\arg\max_c P_{\text{PE}}(c|\theta)$ [2] preceded its interpretation as maximizing $P(c|\theta)$ for a particular model [3].

Note that in the case of diagonal covariance matrices the output sequence components (c_i) are independent given the state sequence. Indeed from (33), (35) and (36) we have $P(c|\theta) = \prod_i P(c_i|\theta)$ where:

$$P(c_i|\theta) = \frac{1}{Z_\theta^i} \prod_{t,d} \mathcal{N}\left(o_{ti}^d | \mu_{\theta_t i}^d, 1/\tau_{\theta_t i}^d\right) \tag{37}$$

3.3 Trajectory HMM

Using different models for parameter estimation and synthesis is inconsistent. The *trajectory HMM* [3, 19] uses the standard synthesis model from §3.2 for both synthesis and parameter estimation. However parameter estimation for the trajectory HMM is more complicated than for the standard HMM, requiring alignment with a delayed-decision Viterbi algorithm and gradient-based parameter re-estimation procedures [3]. The root of these complications is the fact that the normalization constant Z_θ in (36) depends on the entire state sequence θ meaning that $P(c, \theta)$ no longer factorizes nicely with respect to the state sequence.

3.4 $P(c|\theta)$ is Gaussian

The standard synthesis model from §3.2 gives a Gaussian distribution $P(c|\theta)$ over output sequences [3]. Here we re-derive this result using notation that highlights the similarities and differences to the autoregressive HMM.

We assume diagonal covariance matrices, so as we saw in §3.2 the feature vector sequence components are independent given the state sequence. Thus we want to show $P(c_i|\theta)$ is a

multidimensional Gaussian for each static sequence component i. For clarity, in this section we will leave the index i implicit, so c is a sequence of scalars, μ_q^d is a scalar, etc.

From (37):

$$\begin{aligned}
\log P(c_i|\theta) &\stackrel{c}{=} \sum_{u,d} \log \mathcal{N}\left(o_u^d | \mu_{\theta_u}^d, 1/\tau_{\theta_u}^d\right) \\
&= -\tfrac{T(D+1)}{2} \log 2\pi + \tfrac{1}{2} \sum_{u,d} \log \tau_{\theta_u}^d - \tfrac{1}{2} \sum_{u,d} \tau_{\theta_u}^d \left(o_u^d - \mu_{\theta_u}^d\right)^2 \\
&\stackrel{c}{=} -\tfrac{1}{2} \sum_{u,d} \tau_{\theta_u}^d \left((o_u^d)^2 - 2\mu_{\theta_u}^d o_u^d\right) \\
&= -\tfrac{1}{2} \sum_{u,d} \sum_{s,t} \tau_{\theta_u}^d w_{s-u}^d w_{t-u}^d c_s c_t + \sum_{u,d} \sum_{s} \tau_{\theta_u}^d \mu_{\theta_u}^d w_{s-u}^d c_s \\
&= -\tfrac{1}{2} \sum_{s,t} \overline{P}^{st} c_s c_t + \sum_s \overline{b}^s c_s
\end{aligned}$$

where

$$\overline{P}_q^{st} \triangleq \sum_{d=0}^{D} \tau_q^d w_s^d w_t^d \tag{38}$$

$$\overline{b}_q^s \triangleq \sum_{d=0}^{D} \tau_q^d \mu_q^d w_s^d \tag{39}$$

and

$$\overline{P}^{st} \triangleq \sum_u \overline{P}_{\theta_u}^{(s-u)(t-u)} \tag{40}$$

$$\overline{b}^s \triangleq \sum_u \overline{b}_{\theta_u}^{(s-u)} \tag{41}$$

We can see that $P(c_i|\theta)$ is Gaussian with precision matrix $\overline{P}$ and b-value $\overline{b}$.

As for the autoregressive HMM with linear summarizers we have a nice decomposition of the overall precision matrix $\overline{P}$ and b-value $\overline{b}$ as the sum over time of *local contributions*. Specifically (40) shows that the overall precision matrix $\overline{P}$ is the sum of overlapping local contributions $\overline{P}_{\theta_u}$ for each time u, where each local contribution is a $(K+1) \times (K+1)$ matrix which depends only on the state θ_u at that time. Similarly (41) shows that the overall b-value $\overline{b}$ is the sum of overlapping local contributions $\overline{b}_{\theta_u}$ for each time u, where each local contribution is a $(K+1)$-dimensional vector which depends only on the state θ_u at that time. As for the autoregressive HMM (27) provides a schematic view of these decompositions into local contributions.

We can see from (38) that each local contribution to the precision matrix is a state-dependent linear combination of a fixed set of $(K+1) \times (K+1)$ matrices, where each of these fixed matrices $w^d (w^d)^\mathsf{T}$ is the outer product of a vector of window coefficients with

	standard	trajectory	AR
consistent	×	✓	✓
easy and efficient parameter estimation	✓	×	✓
output distribution	joint Gaussian	joint Gaussian	linear Gaussian
$P(c\|\theta)$ is Gaussian built from local contribs	✓	✓	✓
form of local contribs	linear comb of self products	linear comb of self products	self product of linear comb
typical free params	6	6	5
synthesis using dynamic features	✓	✓	✓
synthesis considering global variance	✓	✓	✓

Table 2: Summary of similarities and differences between the standard HMM synthesis framework, the trajectory HMM and the autoregressive HMM

itself.

4 Comparison

In this section we compare the standard HMM synthesis framework, the trajectory HMM and the autoregressive HMM with linear summarizers. A summary of the similarities and differences between these three models is shown in Table 2. We also discuss to what extent the standard model used for synthesis can be emulated using an autoregressive HMM.

The autoregressive HMM and trajectory HMM both use the same probabilistic model during parameter estimation as during synthesis, and are therefore consistent. As we saw in §3.1 and §3.2 the standard HMM synthesis framework does not – the constraints between static and dynamic features are incorporated during synthesis but ignored during parameter estimation.

The joint distribution $P(c, \theta)$ factorizes nicely over time for the autoregressive HMM (1) and similarly $P(o, \theta)$ factorizes nicely over time for the standard parameter estimation model (33). The trajectory HMM does not shared this property due to the normalization constant in (36). This allows the autoregressive HMM and standard HMM synthesis framework to support easy and efficient parameter estimation using expectation maximization, whereas the trajectory HMM requires more complicated parameter estimation procedures [3].

The linear Gaussian state output distributions (2), (3) and (6) used for the autoregressive HMM with linear summarizers are different to the joint Gaussian state output distributions (34) used for the standard HMM framework and the trajectory HMM.

The different forms of state output distribution still lead to similar distributions over output *sequences*. As we showed in §2.3 and §3.4, in all three cases $P(c|\theta)$ is Gaussian with band-diagonal precision matrix and the precision matrix $\overline{P}$ decomposes into a sum over time of local contributions that depend only on the state at that time.

However the form of these local contributions is different. For the autoregressive HMM each local contribution (23) takes the form of *the self product of a linear combination* of fixed basis vectors whereas for the standard HMM synthesis framework each local contribution (38) takes the form of a *linear combination of self products* of fixed basis vectors.[2]

4.1 Emulating the standard model for synthesis

In this section we discuss *emulating* a standard HMM synthesis model (§3.2) with an autoregressive HMM. That is, we are given a standard HMM and we want to find parameters for an autoregressive HMM with linear summarizers that give the same model in some sense. Clearly this cannot be done for the full model $P(c, \theta)$ since the two models are genuinely different, but it is interesting to investigate how far we can get.

Given a state sequence it turns out we can emulate any standard HMM with an autoregressive one, meaning that we can choose parameters for the autoregressive one such that the two distributions $P(c|\theta)$ are identical. To see this, firstly note that the precision matrix in (38) is $(2K+1)$-diagonal, where K is the depth of the window set being used. In general if a set of random variables $(x_{1:T})$ is jointly Gaussian then the conditionals $P(x_t|x_{1:t-1})$ are *linear Gaussian*, meaning that the mean is an affine combination of $x_{1:t-1}$ and the variance is fixed. Furthermore it is a general result that if the joint distribution has a precision matrix that is $(2K+1)$-diagonal then the process is K-Markov, meaning that $P(x_t|x_{1:t-1}) = P(x_t|x_{t-K:t-1})$, so the conditional distribution of x_t only depends on the previous K values.[3] Putting these results together, we see that for a given state sequence we can exactly emulate any standard HMM with an autoregressive HMM (with K windows).

This raises the question of whether we can find a *single* autoregressive HMM which emulates the standard HMM's $P(c|\theta)$ for *any* state sequence θ. In general we can't emulate a given standard HMM local contribution with an autoregressive local contribution, no matter how many windows we use, since a linear combination of, say, 3 self products isn't in general expressible as the self product of *any* single vector. Now if we wanted to emulate a standard HMM with an autoregressive one, so that the precision matrices were the same for every state sequence, it's hard to imagine how we'd be able to achieve this without making all the state-dependent local contributions equal, since the state sequence is arbitrary. Therefore in general it is presumably impossible to emulate $P(c|\theta)$ for all θ.

4.2 Model complexity

For autoregressive HMM, for each state q and feature vector component i we have $D+1$ mean values $\mu_{qi}^{0:D}$ of which D values are redundant, D autoregressive coefficients $a_{qi}^{1:D}$, and a

[2] by the *self product* of a vector a we mean the outer product of a with itself, i.e. the matrix aa^{T}.

[3] we can verify both of these facts at the same time by considering the general formulae for conditioning and for marginalizing a jointly Gaussian distribution with respect to a subset of its components. For our purposes, for each time t, we condition the present and future $x_{t:T}$ on the past $x_{1:t-1}$, then marginalize over the future $x_{t+1:T}$.

single variance value σ_{qi}^2. Here D is the number of summarizers. The total model complexity is therefore $D+2$ free parameters per state per feature vector component. For example if $D=3$ there are 5 free parameters.

For the standard HMM framework and trajectory HMM, for each state q and feature vector component i we have $D+1$ mean values and $D+1$ variance values. Here D is the number of dynamic features – for example $D=2$ for a system with static, delta and delta-delta coefficients. The total model complexity is therefore $2D+2$ free parameters per state per feature vector component. Typically $D=2$ so there are 6 free parameters.

Note that for the autoregressive HMM the number of free parameters is different from the number of accumulated values. We accumulate $D+1$ mean values for $\left\langle f_i^d \right\rangle_q$ and $\frac{1}{2}(D+1)(D+2)$ values for $\left\langle f_i^d f_i^e \right\rangle_q$. Thus for the autoregressive HMM we accumulate a total of $\frac{1}{2}(D+1)(D+4)$ values per state per feature vector component. For example if $D=3$ we accumulate 14 values.

5 Conclusion

We have presented a formulation of the autoregressive HMM for speech synthesis and compared it to the standard HMM synthesis framework and the trajectory HMM. We have given details of how to do efficient parameter estimation and synthesis with the autoregressive HMM and discussed consequences of the autoregressive HMM model. There are substantial similarities between the three models – in particular we have shown that in all three cases the output distribution given a state sequence is a multidimensional Gaussian built up from local contributions. We have also highlighted some important differences between the three models. In light of the similarities to and key advantages over current models we believe the autoregressive HMM represents an attractive candidate for an *efficient* and *consistent* model of speech for speech synthesis.

Acknowledgements

This research was funded by the European Community's Seventh Framework Programme (FP7/2007-2013), grant agreement 213845 (EMIME).

References

[1] A. Black, H. Zen, and K. Tokuda, "Statistical parametric speech synthesis," in *Proc. ICASSP 2007*, pp. 1229–1232, 2007.

[2] K. Tokuda, T. Kobayashi, and S. Imai, "Speech parameter generation from HMM using dynamic features," in *Proc. ICASSP 1995*, vol. 1, 1995.

[3] H. Zen, K. Tokuda, and T. Kitamura, "An Introduction of Trajectory Model into HMM-Based Speech Synthesis," in *Proc. Fifth ISCA Workshop on Speech Synthesis*, 2004.

[4] C. Wellekens, "Explicit time correlation in hidden Markov models for speech recognition," in *Proc. ICASSP 1987*, vol. 12, 1987.

[5] P. Kenny, M. Lennig, and P. Mermelstein, "A linear predictive HMM for vector-valued observations with applications to speech recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, no. 2, pp. 220–225, 1990.

[6] P. Woodland, "Hidden Markov models using vector linear prediction and discriminative output distributions," in *Proc. ICASSP 1992*, vol. 1, pp. 509–512, 1992.

[7] J. Bilmes, "Graphical models and automatic speech recognition," in *Mathematical foundations of speech and language processing* (M. Johnson, S. Khudanpur, M. Ostendorf, and R. Rosenfeld, eds.), Springer-Verlag, 2004.

[8] K. Chin and P. Woodland, "Maximum mutual information training of hidden Markov models with vector linear predictors," in *Proc. Interspeech 2002*, 2002.

[9] A. Poritz, "Linear predictive hidden Markov models and the speech signal," in *Proc. ICASSP 1982*, vol. 7, 1982.

[10] B. Juang and L. Rabiner, "Mixture autoregressive hidden Markov models for speech signals," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 6, pp. 1404–1413, 1985.

[11] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B (Methodological)*, pp. 1–38, 1977.

[12] S. Yu and H. Kobayashi, "An efficient forward-backward algorithm for an explicit-duration hidden Markov model," *IEEE Signal Processing Letters*, vol. 10, no. 1, pp. 11–14, 2003.

[13] H. Zen, "Implementing an HSMM-based speech synthesis system using an efficient forward-backward algorithm," Technical Report TR-SP-0001, Nagoya Institute of Technology, 2007.

[14] HTS working group, "HMM-based speech synthesis system (HTS)." `http://hts.sp.nitech.ac.jp/`. accessed 17 April 2009.

[15] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK book version 3.4*. Cambridge University Engineering Department, 2006. `http://htk.eng.cam.ac.uk/docs/docs.shtml`.

[16] S. Young, J. Odell, and P. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proc. ARPA Human Language Technology Workshop*, pp. 307–312, 1994.

[17] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *Proc. ICASSP 2000*, vol. 3, 2000.

[18] T. Toda and K. Tokuda, "Speech Parameter Generation Algorithm Considering Global Variance for HMM-Based Speech Synthesis," in *Proc. Interspeech 2005*, 2005.

[19] H. Zen, *Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic features*. PhD thesis, Nagoya Institute of Technology, 2006.