Modelling trajectories in statistical speech synthesis

Cambridge statistical speech synthesis (SSS) seminar series

Matt Shannon[1] Heiga Zen[2]

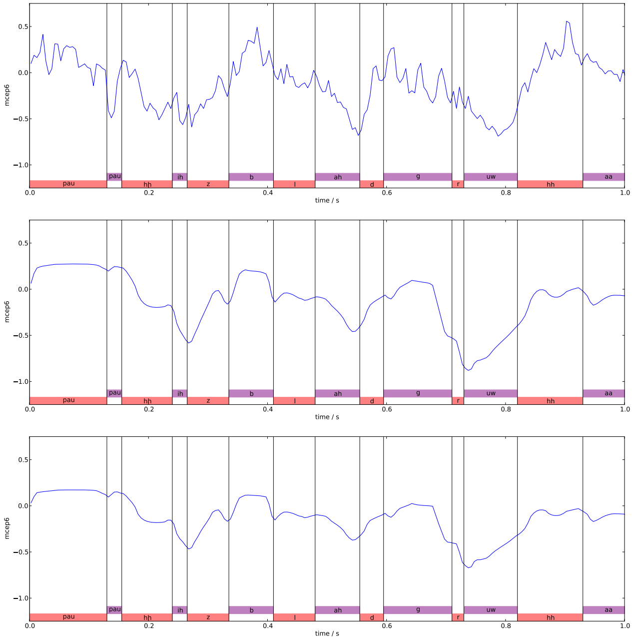
[1]University of Cambridge

[2]Toshiba Research Europe Ltd

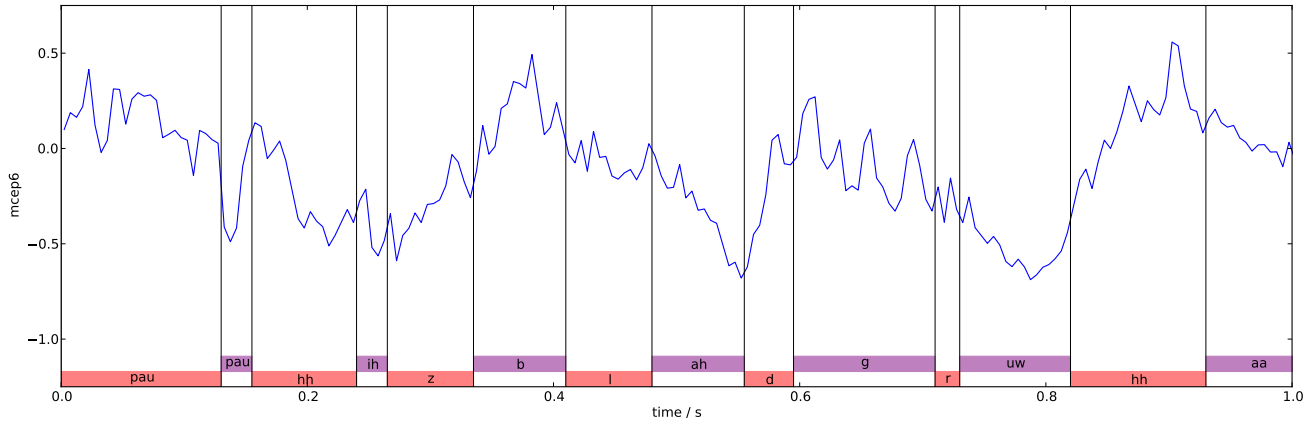
26 January 2011

Outline

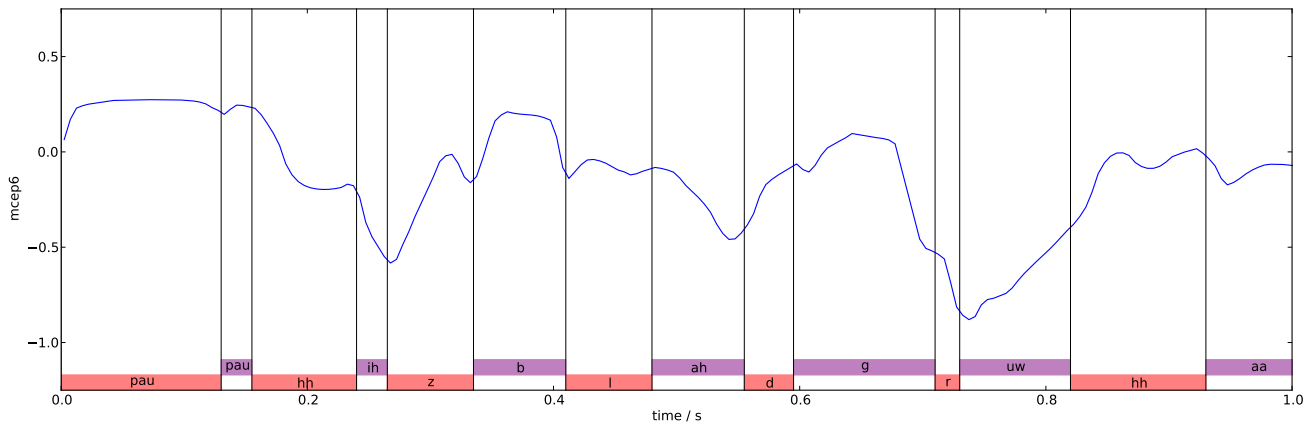
Warm-up – guess the fake



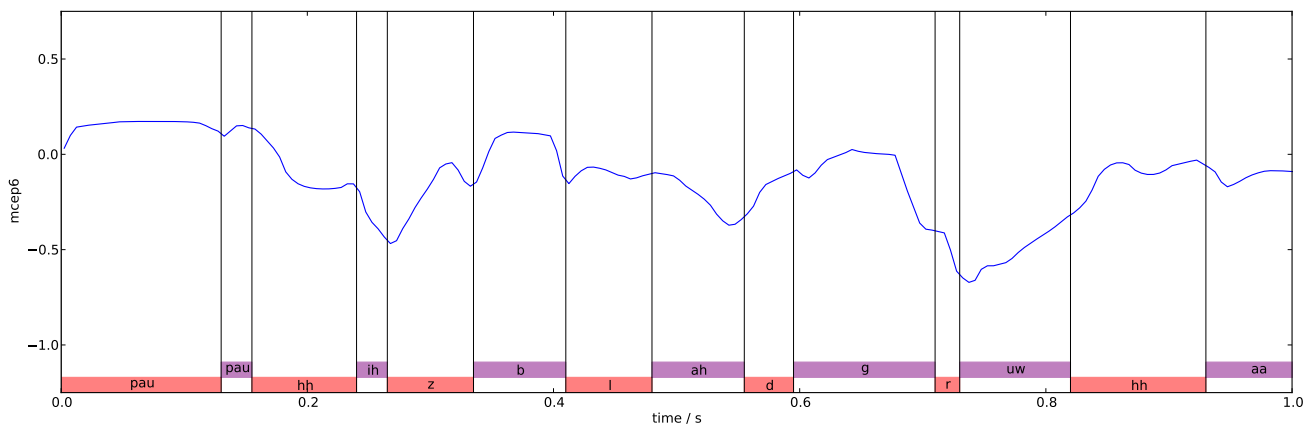
Warm-up – guess the fake



natural

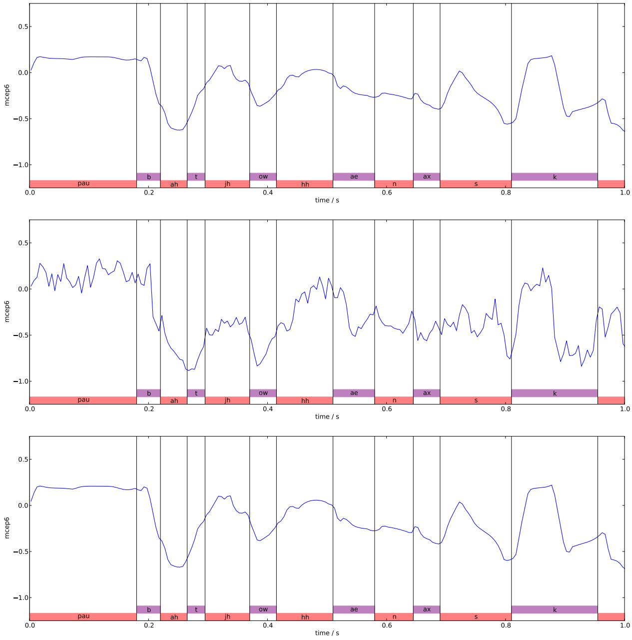


std HMM with GV

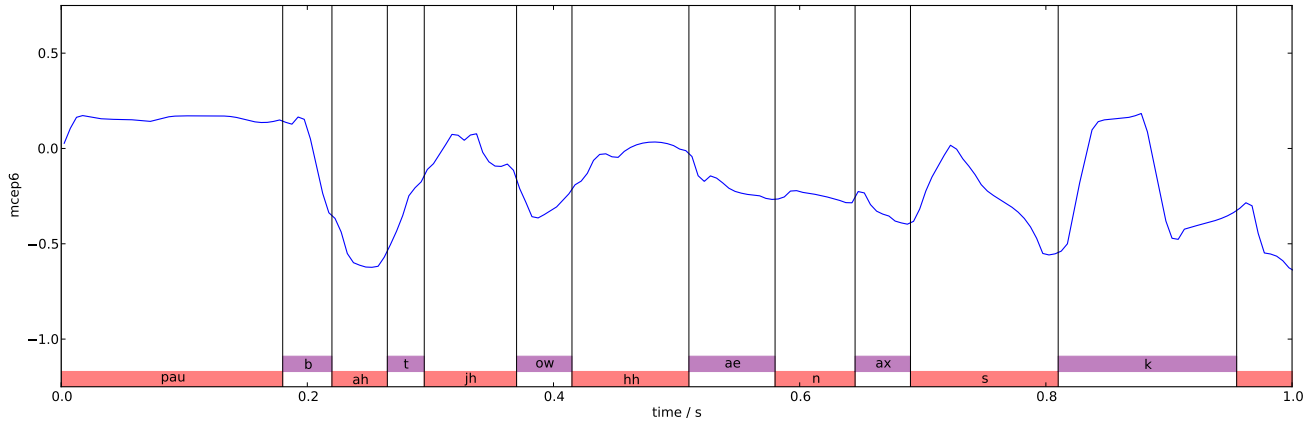


std HMM mean

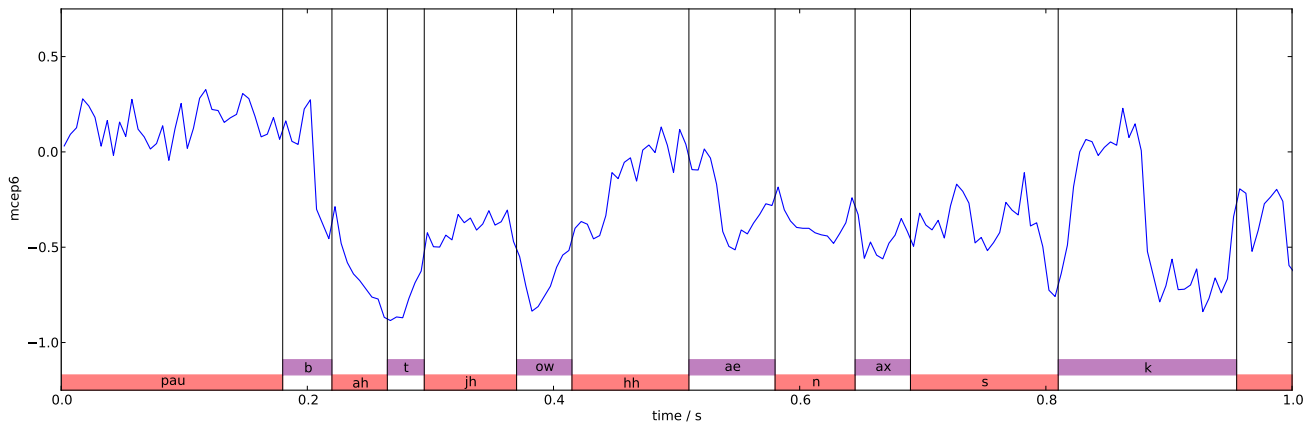
Warm-up – guess the fake



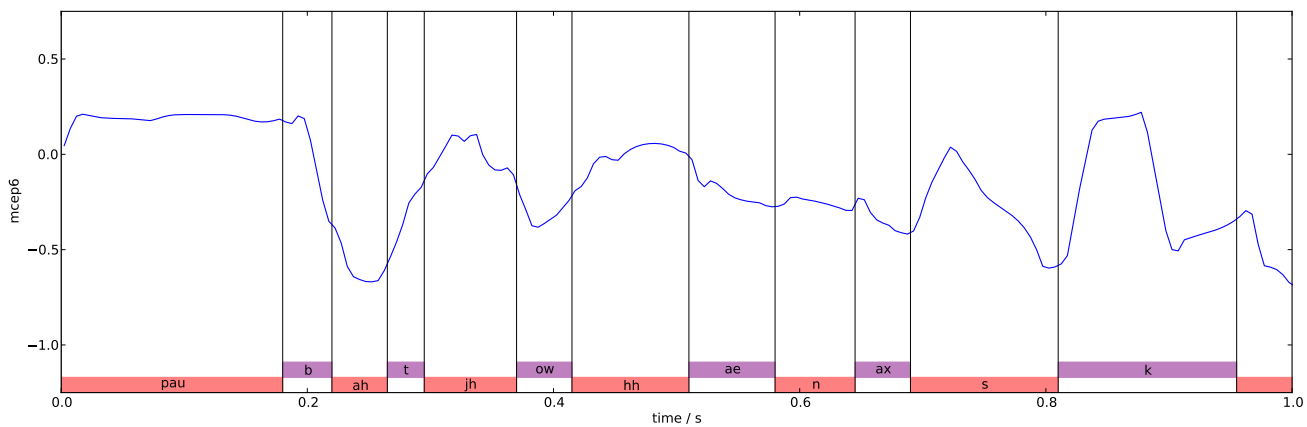
Warm-up – guess the fake



traj HMM mean

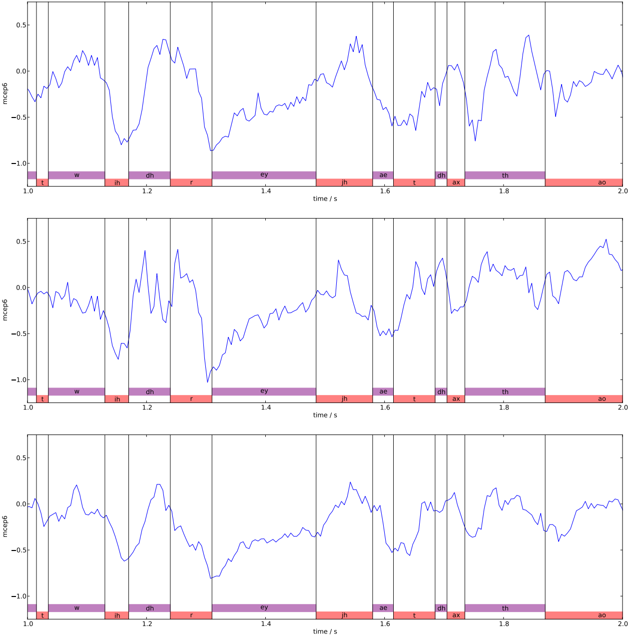


natural

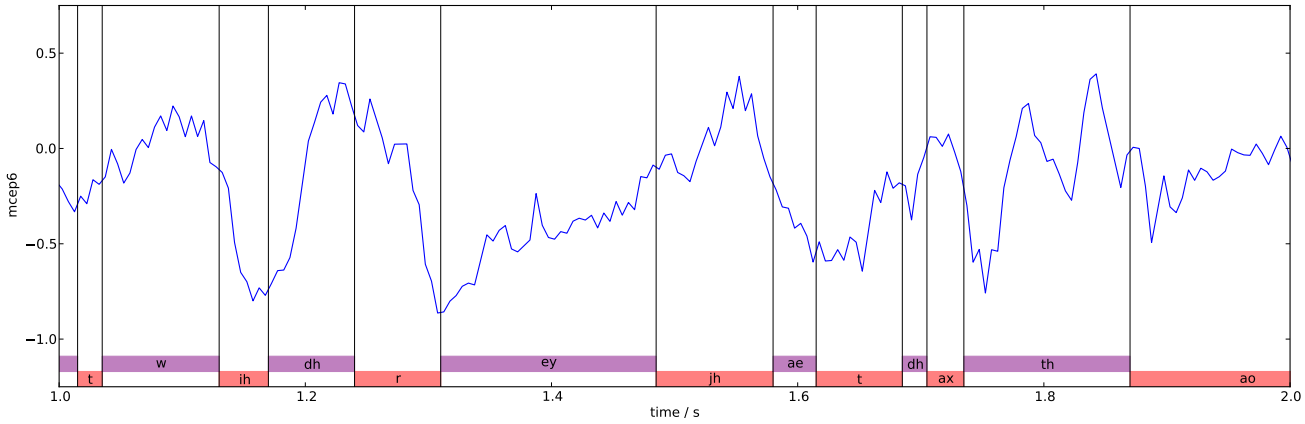


traj HMM with GV

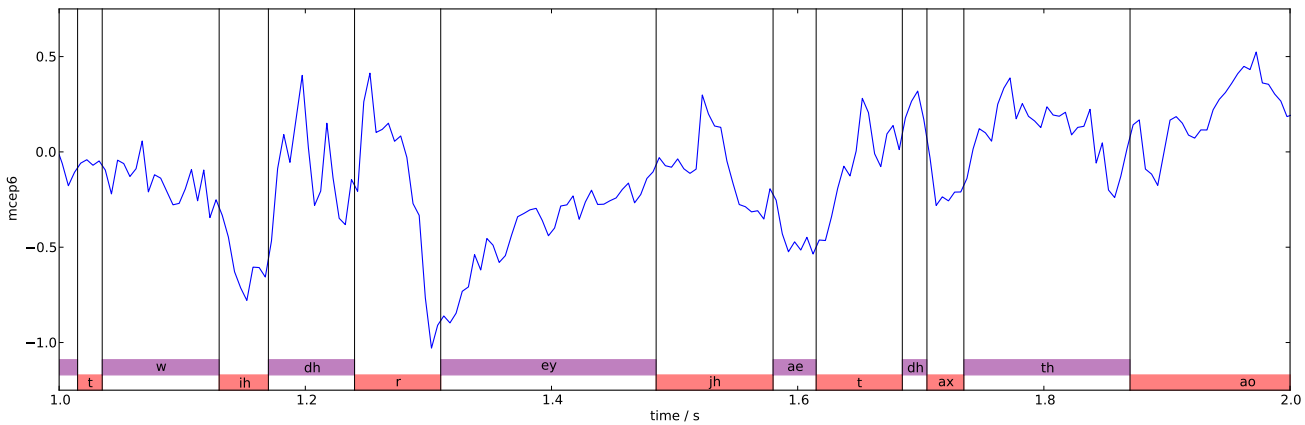
Warm-up – guess the fake



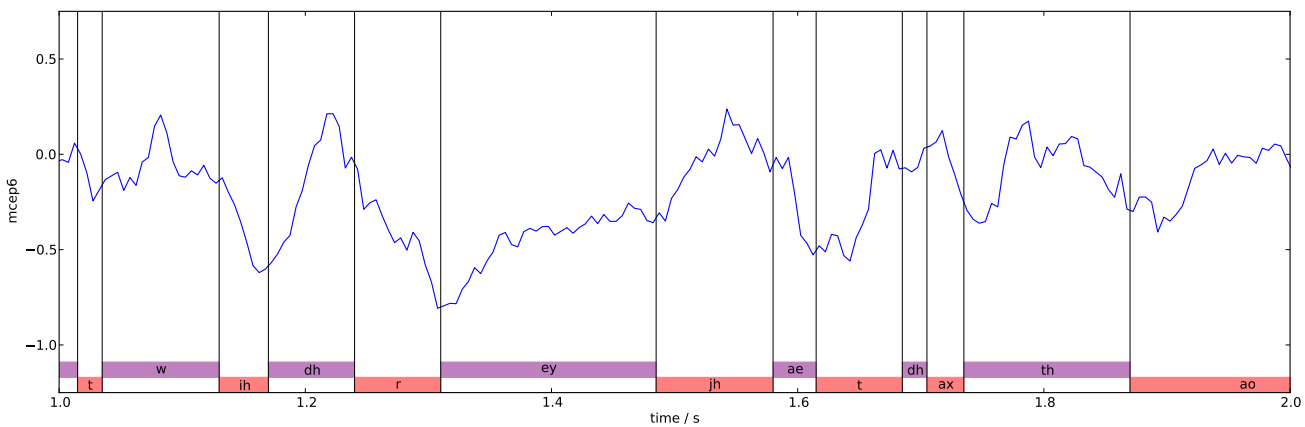
Warm-up – guess the fake



traj HMM sampled

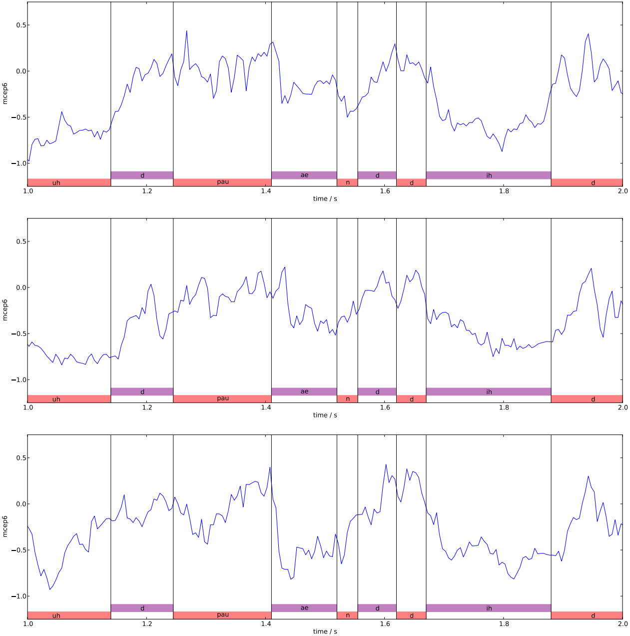


natural

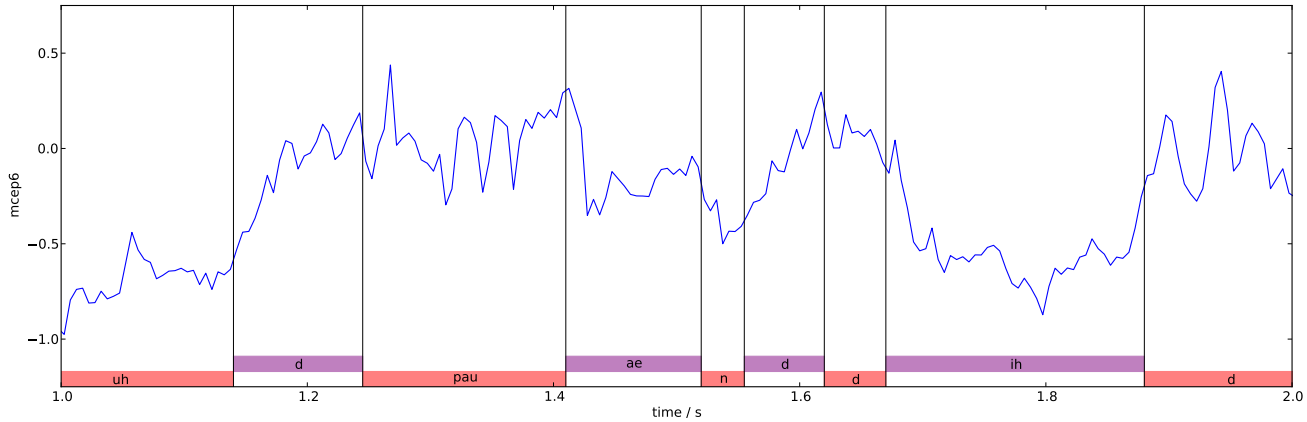


std HMM sampled

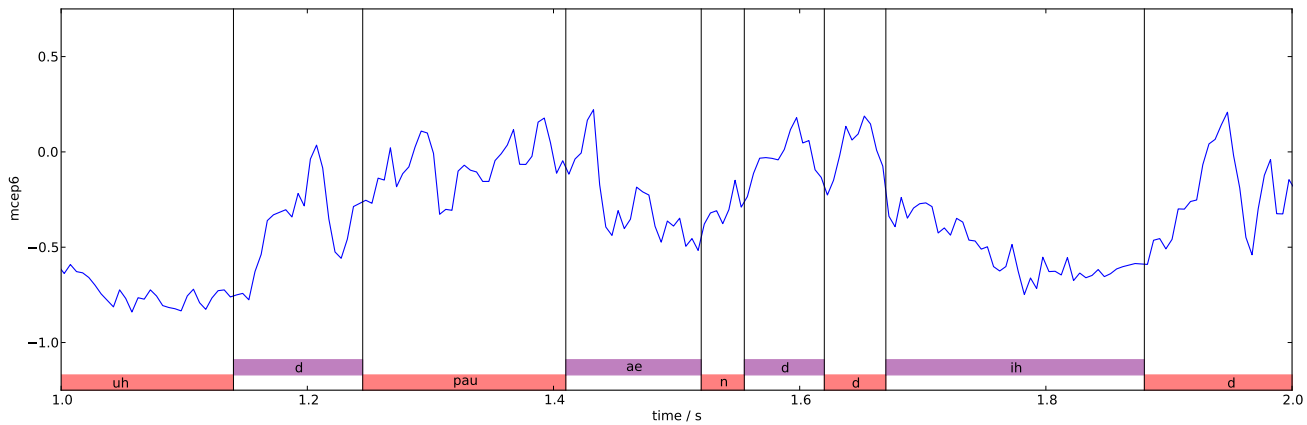
Warm-up – guess the fake



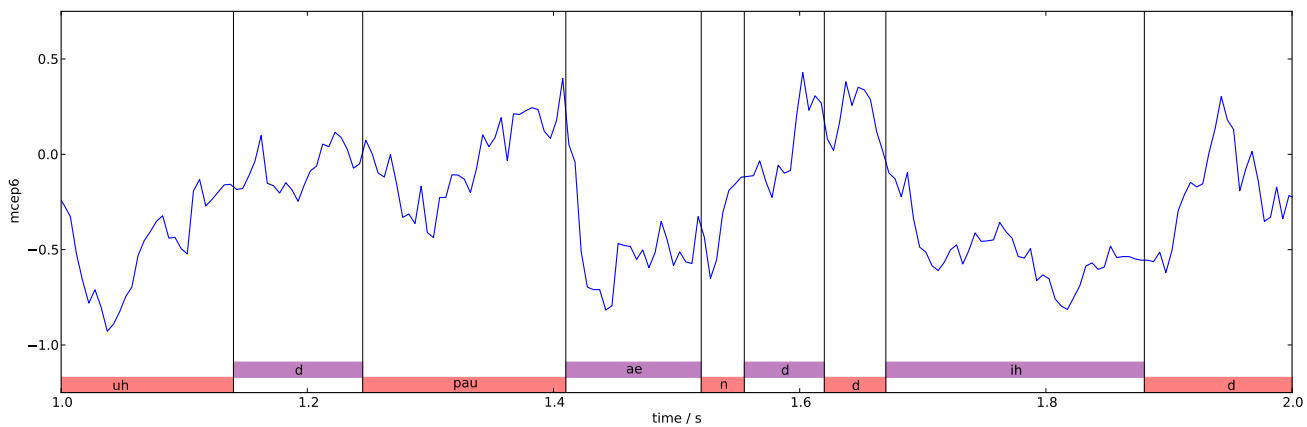
Warm-up – guess the fake



traj HMM sampled



natural



AR HMM sampled

Outline

Overview

Extremely quick overview of statistical speech synthesis

- overall goal is to turn text into speech
- we break this down as

  word seq $\rightarrow$ label seq $\rightarrow$ state seq $\rightarrow$ feature vector seq $\leftrightarrow$ waveform

- label sequence $l = l_{1:J}$
  - e.g. each l_j is a full-context label (quinphone, POS, etc)
- state sequence $q = q_{1:T}$
  - e.g. each q_t is a full-context label together with an integer state index
- feature vector sequence $c = c_{1:T}$
  - e.g. each c_t is a 40-dim static Mel-cepstrum together with a 0/1-dim log F0 value

Overview

Extremely quick overview of statistical speech synthesis

- overall goal is to turn text into speech
- we break this down as

  word seq $\rightarrow$ label seq $\rightarrow$ state seq $\rightarrow$ feature vector seq $\leftrightarrow$ waveform

- label sequence $l = l_{1:J}$
  - e.g. each l_j is a full-context label (quinphone, POS, etc)
- state sequence $q = q_{1:T}$
  - e.g. each q_t is a full-context label together with an integer state index
- feature vector sequence $c = c_{1:T}$
  - e.g. each c_t is a 40-dim static Mel-cepstrum together with a 0/1-dim log F0 value
- in statistical speech synthesis we build probabilistic models $P(q|l)$ (duration model) and $P(c|q)$ (acoustic model)

Overview

To build the models

- use a training corpus $((l^r, c^r))$ of examples from a single speaker
- we assume (generative model) speaker generates c^r for each l^r by
  - sampling q^r randomly from $P_{\text{true}}(q|l = l^r)$
  - sampling c^r randomly from $P_{\text{true}}(c|q = q^r)$ given this q^r
- want to estimate the P_{true} for the speaker as closely as possible, so that we can synthesize c for new unseen label sequences l

Overview

To build the models

- use a training corpus $((l^r, c^r))$ of examples from a single speaker
- we assume (generative model) speaker generates c^r for each l^r by
  - sampling q^r randomly from $P_{\text{true}}(q|l = l^r)$
  - sampling c^r randomly from $P_{\text{true}}(c|q = q^r)$ given this q^r
- want to estimate the P_{true} for the speaker as closely as possible, so that we can synthesize c for new unseen label sequences l

Typically

- assume $P_{\text{true}}(c|q)$ lies in some parametrized family of distributions $\{P(c|q, \lambda) : \lambda\}$ and similarly for $P_{\text{true}}(q|l)$
- use training corpus to estimate parameters λ

Overview

To build the models

- use a training corpus $((l^r, c^r))$ of examples from a single speaker
- we assume (generative model) speaker generates c^r for each l^r by
  - sampling q^r randomly from $P_{\text{true}}(q|l = l^r)$
  - sampling c^r randomly from $P_{\text{true}}(c|q = q^r)$ given this q^r
- want to estimate the P_{true} for the speaker as closely as possible, so that we can synthesize c for new unseen label sequences l

Typically

- assume $P_{\text{true}}(c|q)$ lies in some parametrized family of distributions $\{P(c|q, \lambda) : \lambda\}$ and similarly for $P_{\text{true}}(q|l)$
- use training corpus to estimate parameters λ

One possible goal

- to imitate the given speaker
  - i.e. the perfect synthesis system is one where you can't tell utterances from the training corpus and new synthesized utterances apart
  - note that the original speaker certainly satisfies this criterion

Overview

The duration model $P(q|l,\lambda)$

- usually assumed to have Markov transition structure $P(q|l,\lambda) = \prod_t P(q_t|q_{t-1},l,\lambda)$

Today we'll be focusing on the acoustic model $P(c|q,\lambda)$

- the sequence over time of a single component i of the feature vector (e.g. mcep6) forms a trajectory c^i
- usually assume the trajectories (c^i) for the various feature vector components i are independent given the state sequence

$\Rightarrow$ focus on modelling $c^i|q$

- for clarity of notation
  - drop i index and write c instead of c^i from now on
  - $c_t \in \mathbb{R}$
  - c is a sequence of real numbers over time (a trajectory)
  - $P(c|q,\lambda)$ is a distribution over trajectories
    - slightly tricky concept

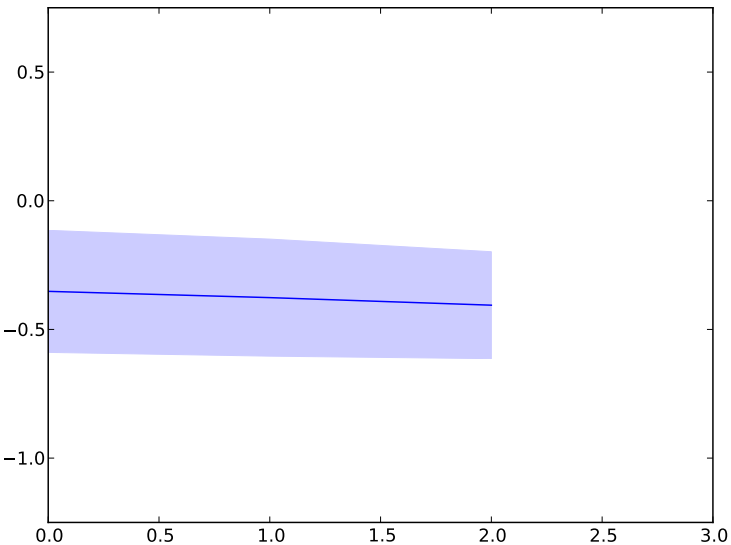
Distributions over trajectories

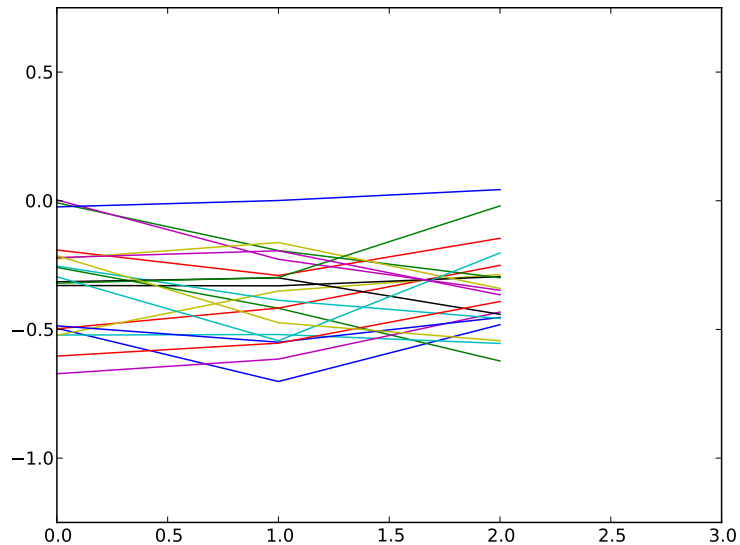
To help explain the concept of a distribution over trajectories, consider a 3-dimensional Gaussian distribution $c \sim \mathcal{N}(\overline{\mu}, \overline{\Sigma})$

$$\overline{\mu} = \begin{pmatrix} -0.352 & -0.376 & -0.405 \end{pmatrix}$$

$$\overline{\Sigma} = \begin{pmatrix} 0.024 & 0.016 & 0.009 \\ 0.016 & 0.022 & 0.013 \\ 0.009 & 0.013 & 0.019 \end{pmatrix}$$

- $P(c)$ assigns a real number to each 3-dimensional vector c
- sampling from $P(c)$ gives a random 3-dimensional vector





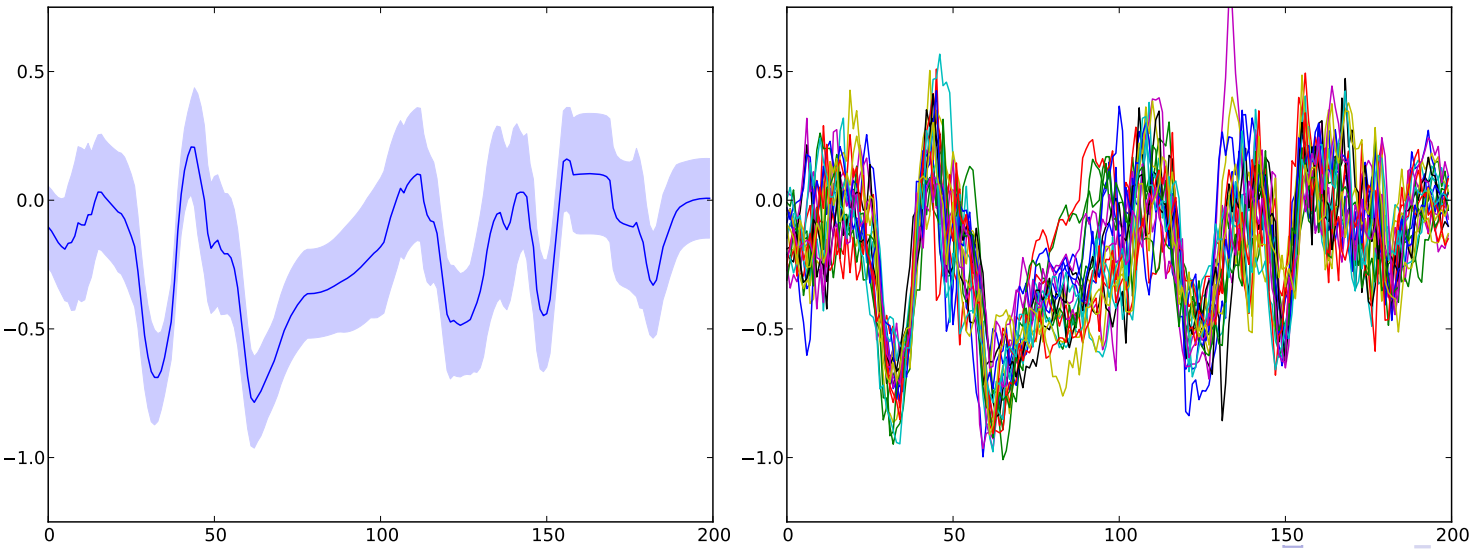
Distributions over trajectories

Now consider a 200-dimensional Gaussian distribution $c \sim \mathcal{N}(\overline{\mu}, \overline{\Sigma})$

$$\overline{\mu} = \begin{pmatrix} -0.103 & -0.119 & -0.143 & -0.164 & -0.179 & \dots \end{pmatrix}$$

$$\overline{\Sigma} = \begin{pmatrix} 0.011 & 0.009 & 0.008 & 0.006 & 0.005 & \dots \\ 0.009 & 0.011 & 0.009 & 0.007 & 0.006 & \dots \\ 0.008 & 0.009 & 0.012 & 0.009 & 0.008 & \dots \\ 0.006 & 0.007 & 0.009 & 0.013 & 0.011 & \dots \\ 0.005 & 0.006 & 0.008 & 0.011 & 0.015 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

- $P(c)$ assigns a real number to each 200-dimensional trajectory c
- sampling from $P(c)$ gives a random 200-dimensional trajectory



Distributions over trajectories

For any Gaussian distribution $c \sim \mathcal{N}(\overline{\mu}, \overline{\Sigma})$

- $\overline{\mu}_t = \mathbb{E} c_t$
- $\overline{\Sigma}_{st} = \mathrm{Cov}(c_s, c_t)$
- in particular $\overline{\Sigma}_{tt} = \mathrm{Var}(c_t)$

Can therefore think of

- $\overline{\mu}$ as the mean trajectory
- $\overline{\Sigma}$ as expressing correlations over time
- in particular $\overline{\Sigma}_{tt}$ gives the marginal or pointwise variance of c_t
  - can vary with time

And each sample from the distribution $P(c)$ is a trajectory

Standard HMM

In the standard HMM synthesis framework

- we augment the trajectory c with delta parameters $c_t^{\Delta} \triangleq \frac{1}{2}c_{t+1} - \frac{1}{2}c_{t-1}$ and delta-delta parameters $c_t^{\Delta\Delta}$ to get an observation sequence $o = (c, c^{\Delta}, c^{\Delta\Delta})$
- we then model $o|q$ instead of $c|q$, with some distribution $\widetilde{P}(o|q, \lambda)$
- o is a deterministic linear transform $o = w(c)$ of c
- the problem is, since $o \in \mathbb{R}^{3T}$ and $c \in \mathbb{R}^{T}$, most random o are incoherent – there is no c such that $o = w(c)$
- therefore a model of $o|q$ doesn't define a model of $c|q$

Why not just restrict to the set of coherent sequences?

- want to set $P(c|q, \lambda) \propto \widetilde{P}(w(c)|q, \lambda)$
- however we need a normalization constant if we want this to define a valid probability distribution over c

$$P(c|q, \lambda) \triangleq \frac{1}{Z(q,\lambda)}\widetilde{P}(w(c)|q, \lambda)$$

Standard HMM

- during synthesis we do this already – we restrict to the subspace of coherent sequences and effectively use

$$P(c|q,\lambda) \triangleq \frac{1}{Z(q,\lambda)} \widetilde{P}(w(c)|q,\lambda)$$

- however during training we effectively use the unnormalized distribution

$$\text{“}P\text{”}(c|q,\lambda) \triangleq \widetilde{P}(w(c)|q,\lambda)$$

$\Rightarrow$ inconsistent

$\Rightarrow$ no guarantee training is doing anything sensible from the point of view of using the trained model for synthesis

- hope to convince you that the standard training really is getting something a bit wrong, and that using the same normalized model for training and synthesis does make a difference

Normalized models

We have seen

- standard model used during training is either unnormalized or defined over the wrong quantity, depending on how you look at it
- standard model effectively used during synthesis is a valid normalized model

So why not use the same normalized model we effectively use during synthesis during training as well?

- this is precisely the trajectory HMM

Normalized models

In fact there are several ways to obtain a normalized model – we could

1. use a model $P(c|q,\lambda)$ which is explicitly built up of local conditional distributions, all of which are individually normalized
   - e.g. autoregressive HMM $P(c|q,\lambda) = \prod_t P(c_t|q_t, c_{t-K:t-1}, \lambda)$
   - locally normalized

Normalized models

In fact there are several ways to obtain a normalized model – we could

1. use a model $P(c|q,\lambda)$ which is explicitly built up of local conditional distributions, all of which are individually normalized
   - e.g. autoregressive HMM $P(c|q,\lambda) = \prod_t P(c_t|q_t, c_{t-K:t-1}, \lambda)$
   - locally normalized
2. choose any positive cost function $U(c;q,\lambda)$ and then normalize the conditional distribution $P(c|q,\lambda) = \frac{1}{Z(q,\lambda)} U(c;q,\lambda)$
   - e.g. trajectory HMM $U(c;q,\lambda) = \widetilde{P}(w(c)|q,\lambda)$
   - globally normalized at the level of c

Normalized models

In fact there are several ways to obtain a normalized model – we could

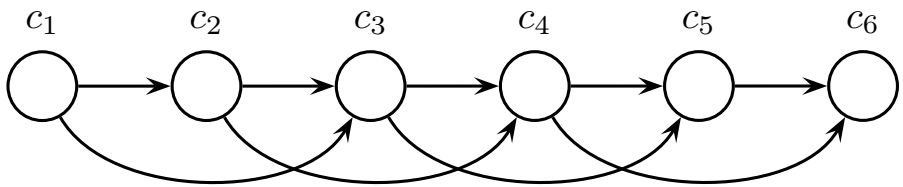
1. use a model $P(c|q,\lambda)$ which is explicitly built up of local conditional distributions, all of which are individually normalized
   - e.g. autoregressive HMM $P(c|q,\lambda) = \prod_t P(c_t|q_t, c_{t-K:t-1}, \lambda)$
   - locally normalized
2. choose any positive cost function $U(c;q,\lambda)$ and then normalize the conditional distribution $P(c|q,\lambda) = \frac{1}{Z(q,\lambda)} U(c;q,\lambda)$
   - e.g. trajectory HMM $U(c;q,\lambda) = \widetilde{P}(w(c)|q,\lambda)$
   - globally normalized at the level of c
3. choose any positive cost function $U(c,q;\lambda)$ and then normalize the joint distribution $P(c,q|\lambda) = \frac{1}{Z(\lambda)} U(c,q;\lambda)$
   - e.g. $U(c,q;\lambda) = \widetilde{P}(w(c),q|\lambda)$
   - as far as I know no-one has tried this
   - globally normalized at the level of (c,q)

Will explore the exact parametrizations of both the autoregressive HMM and trajectory HMM later in the talk.

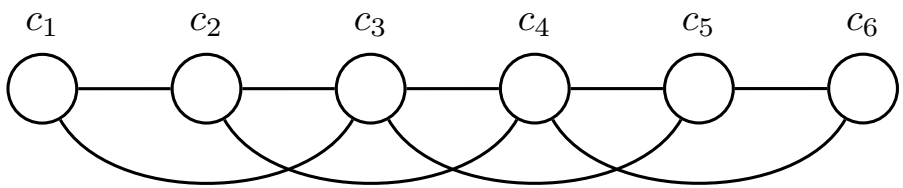
Normalized models (aside)

For those interested all the normalized models we've discussed are examples of probability distributions defined using graphical models

- autoregressive HMM is an example of a directed graphical model (locally normalized)



- trajectory HMM $P(c|q, \lambda)$ distribution is an example of an undirected graphical model, also known as a Markov random field (globally normalized)



Provide a very clean conceptual framework for reasoning about probabilistic models, but we won't discuss further today.

Normalized models

Summary of the different models of $c|q, \lambda$

	training	synthesis
std	"P" $(c|q,\lambda) = \widetilde{P}(w(c)|q,\lambda)$	$P(c|q,\lambda) = \frac{1}{Z(q,\lambda)}\widetilde{P}(w(c)|q,\lambda)$
traj	$P(c|q,\lambda) = \frac{1}{Z(q,\lambda)}\widetilde{P}(w(c)|q,\lambda)$	
AR	$P(c|q,\lambda) = \prod_t P(c_t|q_t, c_{t-K:t-1}, \lambda)$	

Also helpful to know that

- for all of these models $P(c|q,\lambda)$ is Gaussian
- i.e. $c|q,\lambda \sim \mathcal{N}(\overline{\mu}_q, \overline{\Sigma}_q)$ for some mean trajectory $\overline{\mu}_q$ and covariance $\overline{\Sigma}_q$ (which also depend on the parameters λ)
- so we can think of the distribution over trajectories $P(c|q,\lambda)$ in the same way as we did above

Effect of normalization

How does normalization affect the trained models?

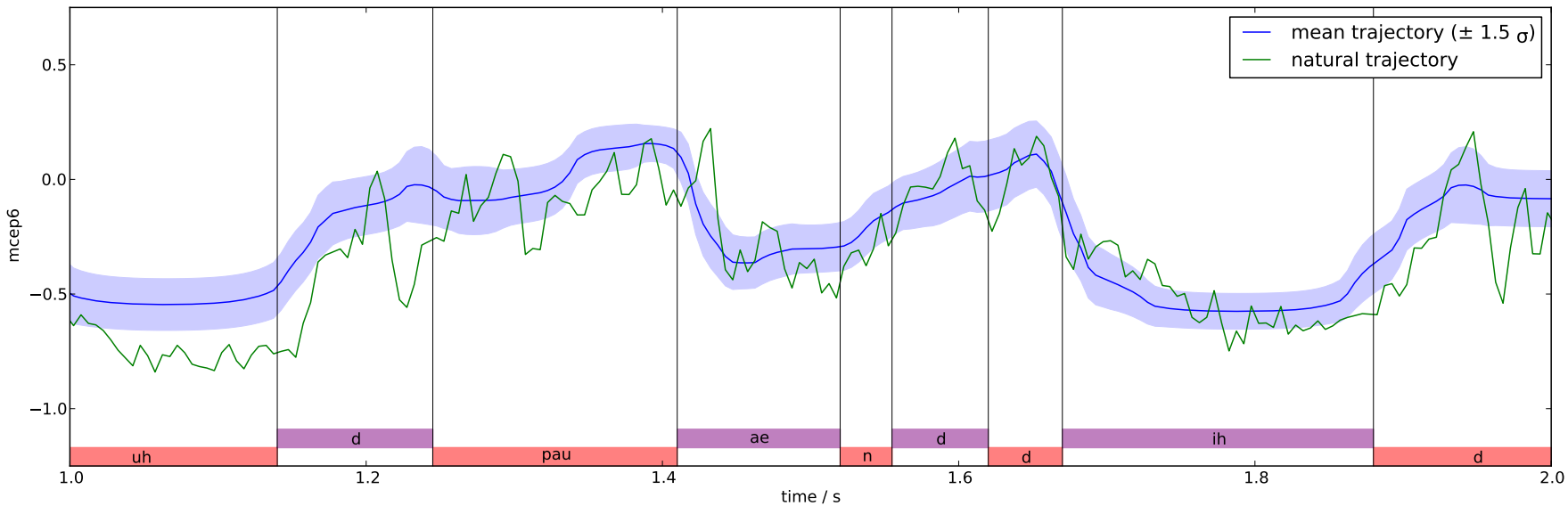
- plot the distribution over trajectories $P(c|q, \lambda)$ for some real utterances
- compare to natural trajectory

Technical details

- mcep6 (7th Mel-cepstral component)
- 1 second of speech
- synthesis given standard CMU ARCTIC phone-level transcription
- plot mean trajectory ± 1.5 standard deviation, and natural trajectory
- (N.B. correlations over time not represented in this picture)

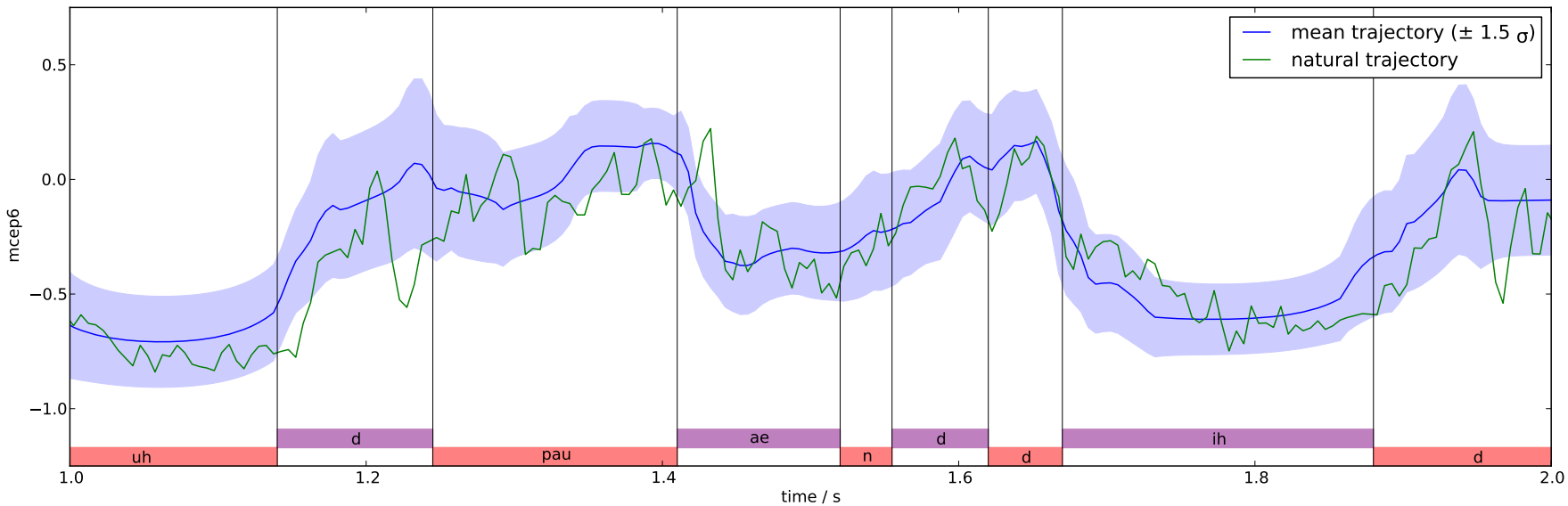
Effect of normalization

Unnormalized (standard HTS training)



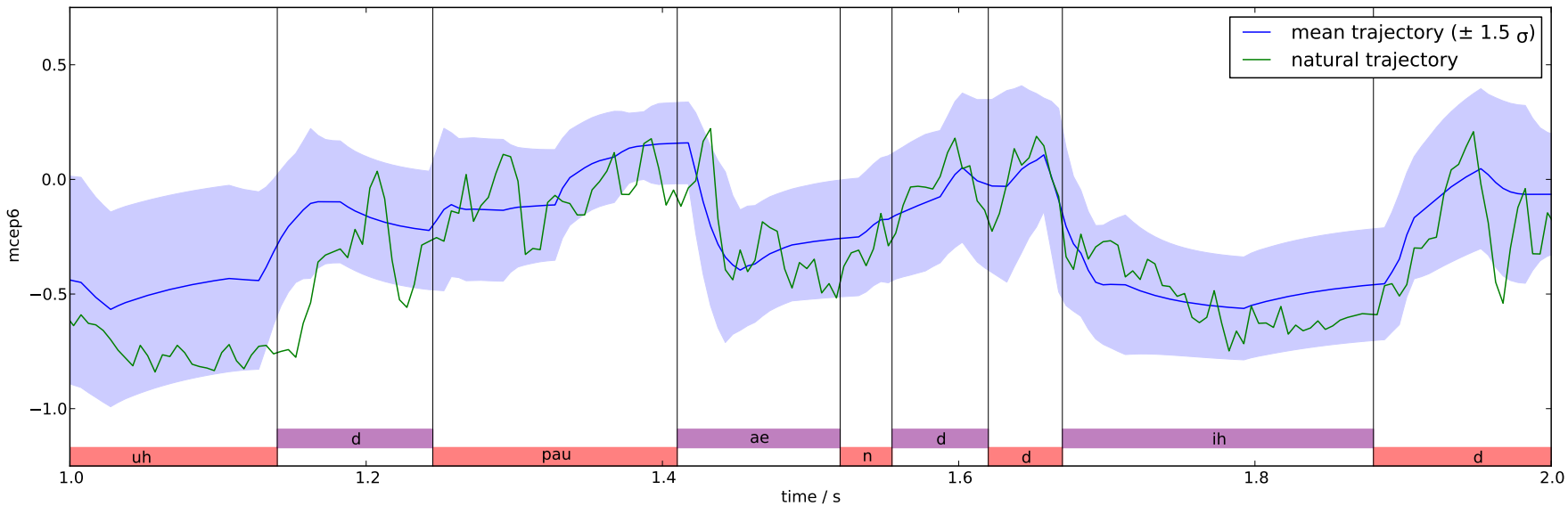
Effect of normalization

Normalized (trajectory HMM)



Effect of normalization

Normalized (autoregressive HMM)



Effect of normalization

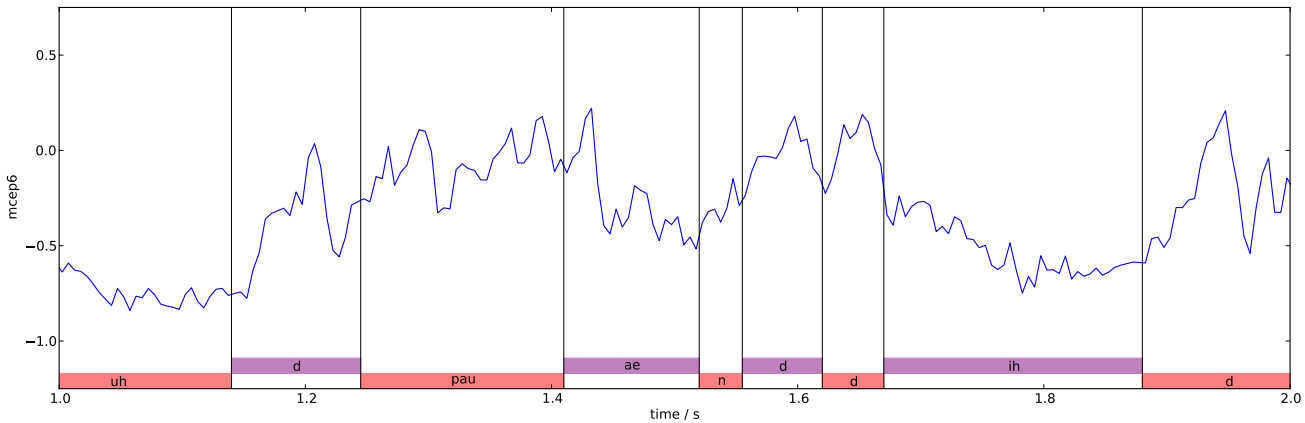
We can see

- the variance of the distribution over trajectories for the unnormalized model is too small (over-confident)
- the variance for the normalized models is larger, and looks more reasonable
- this is reflected in probabilities – log prob per frame of the natural trajectory is
  - 0.3 (unnormalized HMM)
  - 0.9 (trajectory HMM)
  - 0.9 (autoregressive HMM)
- normalization also changes the mean trajectory
  - at least for the trajectory HMM, improves naturalness of synthesized mean trajectories[1]

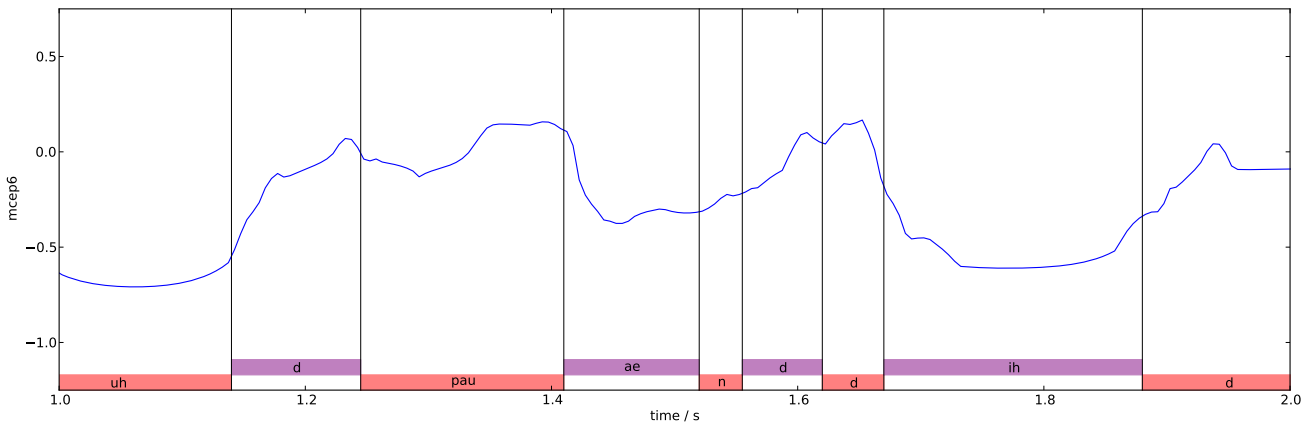
[1] H. Zen, K. Tokuda, and T. Kitamura. An Introduction of Trajectory Model into HMM-Based Speech Synthesis. In *Proc. Fifth ISCA Workshop on Speech Synthesis*, 2004

Sampling trajectories

- so far we have good reasons (subjective and objective) to believe normalized models are better models of speech
- but mean trajectories still look very unrealistic – much too smooth



natural



traj HMM mean

Sampling trajectories

The smoothness is caused in part by taking the mean trajectory

- our stated goal was to imitate the original speaker exactly (to extend the training corpus without anyone realizing)
- our assumption during training is that the training corpus was generated by the speaker sampling (independently) from $P(c|q, \lambda)$ for each utterance

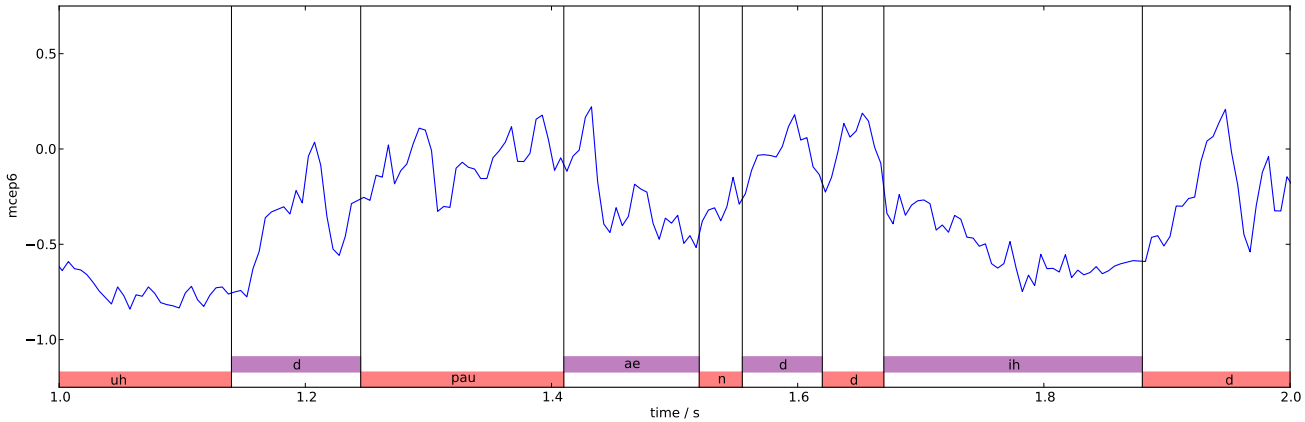
$\Rightarrow$ should really do synthesis by sampling trajectory

In this view

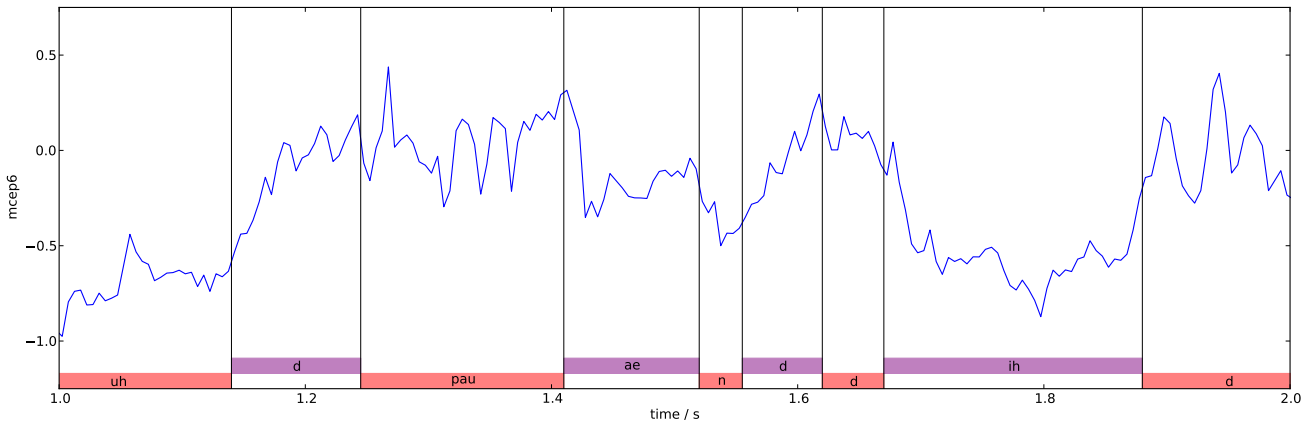
- the fact the mean trajectory sounds over-smoothed is not a sign of anything going wrong – we would probably expect the mean trajectory to be smoother than any given random trajectory
- the random part of the probability distribution over trajectories is aiming to capture the speaker's natural variability – the speaker says the same label sequence slightly differently each time they say it

Sampling trajectories

Sampled trajectories certainly capture the characteristic roughness of natural trajectories



natural



traj HMM sampled

Sampling trajectories

Sampled trajectories from the normalized models we have currently

- look more like natural speech than mean trajectories
- have some nice properties
  - e.g. sampled trajectories from these normalized models have almost completely natural global variance distributions, without using any additional global variance modelling
- sound terrible (!)
  - traj HMM with GV
  - traj HMM mean
  - traj HMM sampled

Sampling trajectories

Sampled trajectories from the normalized models we have currently

- look more like natural speech than mean trajectories
- have some nice properties
  - e.g. sampled trajectories from these normalized models have almost completely natural global variance distributions, without using any additional global variance modelling
- sound terrible (!)
  - traj HMM with GV
  - traj HMM mean
  - traj HMM sampled

$\Rightarrow$ existing models are not modelling something they should be modelling

- (this conclusion still holds even if you prefer doing the synthesis itself using the mean trajectory)

Sampling trajectories

Sampled trajectories from the normalized models we have currently

- look more like natural speech than mean trajectories
- have some nice properties
  - e.g. sampled trajectories from these normalized models have almost completely natural global variance distributions, without using any additional global variance modelling
- sound terrible (!)
  - traj HMM with GV
  - traj HMM mean
  - traj HMM sampled

$\Rightarrow$ existing models are not modelling something they should be modelling

  - (this conclusion still holds even if you prefer doing the synthesis itself using the mean trajectory)
- and it seems to be something low-level – instantly noticeable and uniform over the utterance, not some complicated contextual effect

A unified view of current normalized models (optional)

Can distinguish two inter-related aspects to modelling $c|q$ well

1. model the random variation present for fixed q
   - imagine we fix the state sequence q once and for all
   - just try to model the variability in the way speaker says the utterance
   - not necessarily easy!
2. model the way the overall distribution $P(c|q, \lambda)$ over c depends on the individual states q_t at each time t
   - expect state at time t to have a local effect on trajectory – i.e. affect mainly $c_{t-L:t+L}$ for some L
   - the overlapping local effects of states near each other in the state sequence should interact in such a way that even unseen state sequences result in a sensible overall distribution $P(c|q, \lambda)$

How do current normalized models approach these two aspects?

A unified view of current normalized models (optional)

1. model the random variation present for fixed q

2. model the way the overall distribution over c depends on the individual states q_t at each time t

A unified view of current normalized models (optional)

1. model the random variation present for fixed q
   - assume $c|q$ is a Gaussian, i.e. $c|q \sim \mathcal{N}(\overline{\mu}_q, \overline{\Sigma}_q)$
     - Gaussian is over time (c is a T-dimensional vector)
     - $\overline{\mu}_q$ is mean trajectory
2. model the way the overall distribution over c depends on the individual states q_t at each time t

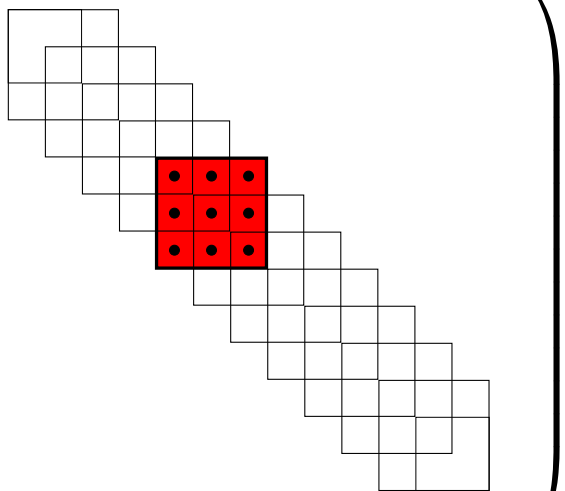
A unified view of current normalized models (optional)

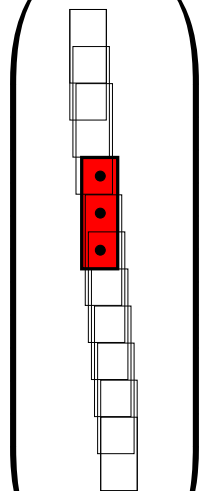
1. model the random variation present for fixed q
   - assume $c|q$ is a Gaussian, i.e. $c|q \sim \mathcal{N}(\overline{\mu}_q, \overline{\Sigma}_q)$
     - Gaussian is over time (c is a T-dimensional vector)
     - $\overline{\mu}_q$ is mean trajectory
2. model the way the overall distribution over c depends on the individual states q_t at each time t
   - define $\overline{P}_q = \overline{\Sigma}_q^{-1}$ (precision matrix) and $\overline{b}_q = \overline{P}_q \overline{\mu}_q$ (b-value)
   - assume the effect of the state q_t at time t is local in terms of the precision matrix and b-value
     - q_t affects $(\overline{b}_q)_{t-K_L:t+K_R}$
     - q_t affects $(\overline{P}_q)_{(t-K_L:t+K_R)(t-K_L:t+K_R)}$
     - N.B. effect of q_t on $\overline{\Sigma}_q$ and $\overline{\mu}_q$ typically lasts much longer than K frames
   - $\overline{P}_q$ and $\overline{b}_q$ are the natural parameters of the Gaussian

A unified view of current normalized models (optional)

In other words, $\overline{P}_q$ and $\overline{b}_q$ are built up from overlapping local contributions

$$\overline{P}_q = \left(\quad \right) \qquad \overline{b}_q = \left(\quad \right)$$





- the difference between the autoregressive HMM and trajectory HMM is in the form of the local contributions[2]

[2] M. Shannon and W. Byrne. A formulation of the autoregressive HMM for speech synthesis. Technical Report CUED/F-INFENG/TR.629, Department of Engineering, University of Cambridge, UK, 2009b. http://mi.eng.cam.ac.uk/~sms46/papers/shannon2009fah.pdf

Summary

To summarize so far

- standard model used during training is unnormalized
- normalization (trajectory HMM, autoregressive HMM) results in a better distribution over trajectories
  - theoretically more consistent
    - uses the same normalized model for training and synthesis
  - subjectively better
    - sampled trajectories from normalized models have many large rises and falls, just like natural trajectories, whereas sampled trajectories from the standard model are slightly too tame
    - the natural trajectory is massively outside the expected range less often with normalized models
  - objectively better
    - greatly increases test set log probability

Summary

- need to sample trajectories to take full advantage of the better covariance present in normalized models
  - theoretically the right thing to do
  - generates much more natural looking trajectories
  - sounds terrible (!)
  - existing models (standard HMM, trajectory HMM, autoregressive HMM) are all failing to capture some important low-level aspect of speech
- (optionally) have also seen that the standard HMM used during synthesis, trajectory HMM and autoregressive HMM have substantial similarities
  - $P(c|q, \lambda)$ is a Gaussian
    - with precision matrix and b-value built up from local contributions
    - difference between the different models is in the form of these local contributions

Outline

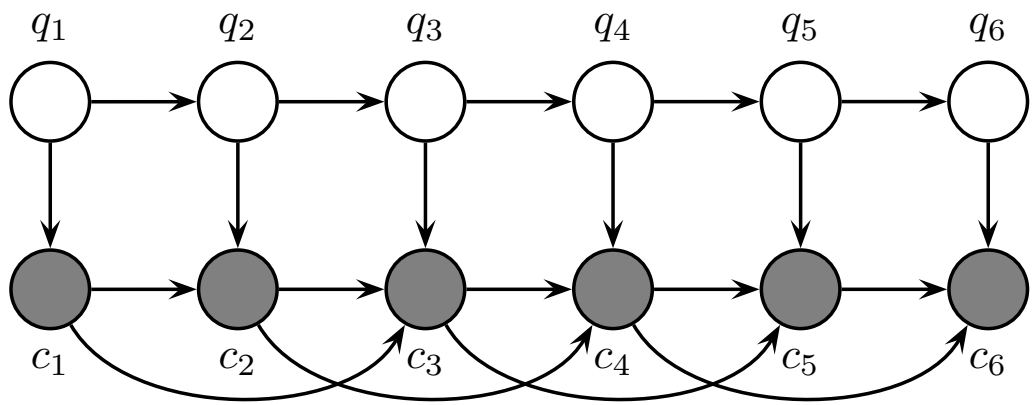
Model

The autoregressive HMM uses

- the standard duration model $P(q|l,\lambda) = \prod_t P(q_t|q_{t-1}, l, \lambda)$
- an autoregressive acoustic model of depth K

$$P(c|q,\lambda) = \prod_t P(c_t|c_{t-K:t-1}, q_t, \lambda)$$

For example, for depth $K = 2$



Locally normalized – conditional distributions $P(c_t|c_{t-K:t-1}, q_t, \lambda)$ are all individually normalized

Model

- turns problem of learning a model over trajectories $P(c|q)$ from data

t	...	20	21	22	23	24	...
q_t	...	ae-3	ae-4	ae-4	ae-4	t-0	...
c_t	...	1.0	1.3	1.6	2.0	1.8	...

- into learning a function $(c_{t-K:t-1}, q_t) \mapsto c_t$ from data

(c_{t-2}, c_{t-1}, q_t)	$\mapsto$	c_t
$(0.6, 0.7, \text{ae-3})$	$\mapsto$	1.0
$(0.7, 1.0, \text{ae-4})$	$\mapsto$	1.3
$(1.0, 1.3, \text{ae-4})$	$\mapsto$	1.6
$(1.3, 1.6, \text{ae-4})$	$\mapsto$	2.0
$(1.6, 2.0, \text{t-0})$	$\mapsto$	1.8

- a standard regression problem

$\Rightarrow$ can plug in any regression model

Model

- often we assume a linear-Gaussian form for the regression model

$$c_t | c_{t-K:t-1}, (q_t = m), \lambda \sim \mathcal{N}\left(\sum_{k=1}^{K} \underbrace{a_m^k}_{coeffs} c_{t-K} + \underbrace{a_m^{K+1}}_{\text{bias}}, \underbrace{(\sigma^2)_m}_{\text{variance}} \right)$$

- the collection of model parameters λ contains the model parameters $(a_m, (\sigma^2)_m)$ for each state m
- but we can in principle use any regression model, including complicated non-Gaussian non-linear regression models

$\Rightarrow$ flexible

Training and synthesis

A nice aspect of the autoregressive HMM is that in spite of this flexibility its factorization properties ensure we can do efficient training and synthesis

- $P(c, q|\lambda)$ factorizes over time for the state sequence q

$\Rightarrow$ we can do efficient Viterbi, Forward-Backward, etc as for the standard HMM

- parameter re-estimation procedure depends on the form of regression model used
- for the autoregressive HMM with a linear-Gaussian regression model the parameter re-estimation procedure is as follows[3]
  - accumulate a $(K+1) \times (K+1)$ matrix R_m (typically 4×4)
  - and a $(K+1)$-dimensional vector r_m
  - re-estimate a_m by solving $R_m a_m = r_m$

  $\Rightarrow$ efficient training using expectation-maximization

[3] M. Shannon and W. Byrne. Autoregressive HMMs for speech synthesis. In *Proc. Interspeech 2009*, 2009a. `http://mi.eng.cam.ac.uk/~sms46/papers/shannon2009ahs.pdf`

Training and synthesis

- autoregressive decision tree clustering is conceptually the same as for the standard HMM but uses autoregressive accumulators and output distributions instead of standard ones
  - autoregressive clustering improves naturalness slightly compared to re-using standard HMM trees with the autoregressive HMM[4]
- synthesis is just as for the standard HMM framework, since $P(c|q,\lambda)$ is still just a Gaussian

  $\Rightarrow$ can do synthesis considering global variance as normal
  - synthesis considering global variance for the autoregressive HMM has roughly the same naturalness as for the standard HMM[5]

[4] M. Shannon and W. Byrne. Autoregressive clustering for HMM speech synthesis. In *Proc. Interspeech 2010*, 2010. `http://mi.eng.cam.ac.uk/~sms46/papers/shannon2010autoregressive.pdf`

[5] M. Shannon and W. Byrne. Autoregressive HMMs for speech synthesis. In *Proc. Interspeech 2009*, 2009a. `http://mi.eng.cam.ac.uk/~sms46/papers/shannon2009ahs.pdf`

Summary

The autoregressive HMM provides

- a consistent normalized model that can be used during both training and synthesis
- efficient training using expectation-maximization
- a flexible framework
  - e.g. non-linear regression models
- high-quality synthesis

Comparison (if time)

The autoregressive HMM has some strong similarities and some important differences to the trajectory HMM

- if we fix the state sequence q the models are equivalent – both model $c|q$ as a Gaussian with band-diagonal precision matrix
- more generally for the autoregressive HMM

$$P(c_t|c_{1:t-1}, q, \lambda) = P(c_t|c_{t-K:t-1}, q_t, \lambda)$$

whereas viewing the the trajectory HMM from this autoregressive point of view

$$P(c_t|c_{1:t-1}, q, \lambda) = P(c_t|c_{t-K:t-1}, q_{t-1:T}, \lambda)$$

Thus

- in principle the trajectory HMM can depend on how long the current state lasts, and what future states are
- whereas the autoregressive HMM doesn't know when the state q_t is about to change

$\Rightarrow$ the autoregressive HMM may use states differently to the trajectory HMM

Outline

Trajectory HMM

[see Heiga's slides]

Acknowledgements

Matt Shannon

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

References I

M. Shannon and W. Byrne. Autoregressive HMMs for speech synthesis. In *Proc. Interspeech 2009*, 2009a. `http://mi.eng.cam.ac.uk/~sms46/papers/shannon2009ahs.pdf`.

M. Shannon and W. Byrne. A formulation of the autoregressive HMM for speech synthesis. Technical Report CUED/F-INFENG/TR.629, Department of Engineering, University of Cambridge, UK, 2009b. `http://mi.eng.cam.ac.uk/~sms46/papers/shannon2009fah.pdf`.

M. Shannon and W. Byrne. Autoregressive clustering for HMM speech synthesis. In *Proc. Interspeech 2010*, 2010. `http://mi.eng.cam.ac.uk/~sms46/papers/shannon2010autoregressive.pdf`.

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.