Globally Normalized Model for Statistical Speech Synthesis

Heiga ZEN

Toshiba Research Europa Ltd.
Cambridge Research Lab.

Speech Synthesis Seminar Series @ CUED, Cambridge, UK
January 26th, 2011
Outline

Trajectory HMM
- Speech parameter generation
- Derivation of trajectory HMM
- Relationship between parameter generation & trajectory HMM
- Trajectory HMM as globally normalized model
- Parameter estimation

Min generation error (MGE) training & trajectory HMM
- Relationship
- Properties of MGE

(If time remains) Product of Experts (PoE)
- Combination of multiple AMs as PoE
- PoE & trajectory HMM
HMM-based speech synthesis system

**Training part**
- Speech signal
- Spectral parameter extraction
- Spectral parameters
- Context-dependent HMMs & state duration models
- Training HMMs
  - Excitation parameter extraction
  - Excitation parameters
- Labels

**Synthesis part**
- TEXT
- Text analysis
- Labels
- Excitation parameters
- Excitation generation
- Spectral parameters
- Synthesis Filter
  - Excitation
- SYNTHESIZED SPEECH
  - Parameter generation from HMMs
  - Spectral parameters
Speech parameter generation algorithm

Determine a speech parameter vector sequence that maximizes its output probability given label $l$ & HMM $\lambda$

\[ \hat{o} = \arg \max_o p(o \mid l, \lambda) \]
\[ = \arg \max_o \sum_{q} p(o \mid q, \lambda)p(q \mid l, \lambda) \]
\[ = \arg \max_{o,q} p(o \mid q, \lambda)p(q \mid l, \lambda) \]
\[ \hat{q} = \arg \max_q p(q \mid l, \lambda) \]
\[ \hat{o} = \arg \max_o p(o \mid \hat{q}, \lambda) \]
Output prob of $o$ given $l$ & HMM $\lambda$

$$p(o \mid l, \lambda) = \sum_{q} p(o \mid q, \lambda) P(q \mid l, \lambda)$$

**state-output**

$$p(o \mid q, \lambda) = \prod_{t=1}^{T} \mathcal{N}(o_t ; \mu_{qt}, \Sigma_{qt}) \quad \leftarrow \text{single Gaussian}$$

$$= \mathcal{N} \left( \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_T \end{bmatrix} ; \begin{bmatrix} \mu_{q_1} \\ \mu_{q_2} \\ \vdots \\ \mu_{q_T} \end{bmatrix}, \begin{bmatrix} \Sigma_{q_1} & \Sigma_{q_2} & \cdots & 0 \\ \Sigma_{q_2} & \Sigma_{q_T} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{q_T} \end{bmatrix} \right)$$

$$= \mathcal{N}(o ; \mu_q, \Sigma_q) \quad \text{diagonal}$$
Generated trajectory

\[ \hat{o} = \arg\max_o p(o \mid \hat{q}, \hat{\lambda}) \]

\[ = \arg\max_o \mathcal{N}(o ; \mu_{\hat{q}}, \Sigma_{\hat{q}}) \]

\[ = \mu_{\hat{q}} \iff \text{mean vector sequence} \]
Relationship between $o$ and $c$

$$O_t = \begin{bmatrix} C_t \\ \Delta C_t \end{bmatrix} \iff \text{static}$$

$$\Delta C_t = C_t - C_{t-1}$$

\[
\begin{array}{ccccccc}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
C_{t-1} & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\Delta C_{t-1} & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
C_t & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\Delta C_t & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
C_{t+1} & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\Delta C_{t+1} & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

\[
\begin{array}{ccccccc}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & -I & I & 0 & 0 & \cdots & \vdots \\
\vdots & I & -I & 0 & 0 & \cdots & \vdots \\
\vdots & -I & I & 0 & 0 & \cdots & \vdots \\
\vdots & I & -I & I & 0 & \cdots & \vdots \\
\vdots & -I & I & I & 0 & \cdots & \vdots \\
\vdots & I & -I & I & I & \cdots & \vdots \\
\vdots & -I & I & I & I & \cdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

\[
\begin{array}{ccccccc}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

\[
\begin{array}{ccccccc}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

Static vs. Dynamic
Speech parameter generation algorithm

\[ \hat{o} = \arg \max_{o} p(o \mid \hat{q}, \hat{\lambda}) \bigg|_{o=Wc} \]

\[ = \arg \max_{o} \mathcal{N}(o \mid \mu_{\hat{q}}, \Sigma_{\hat{q}}) \bigg|_{o=Wc} \]

\[ \Downarrow \]

\[ \hat{c} = \arg \max_{c} \mathcal{N}(Wc \mid \mu_{\hat{q}}, \Sigma_{\hat{q}}) \]

\[ \Downarrow \]

\[ W^{\top} \Sigma_{q}^{-1} W \hat{c} = W^{\top} \Sigma_{q}^{-1} \mu_{q} \]
Generated trajectory

Static

Dynamic

Mean

Variance

\(c\)
Inconsistency between training & synthesis

Training & synthesis parts are inconsistent

- Training part
  * Baum-Welch training
  * Labels are often given manually
  * Model training model w/o dynamic feature constraints

- Synthesis part
  * Viterbi (single-path) approximation
  * Labels are often given automatically (by text analysis)
  * Parameter generation w/ dynamic feature constraints

How about introducing dyn feature constraints to training?
Output prob of $o$ given $l$ & HMM $\lambda$

$$p(o \mid l, \lambda) = \sum_{\forall q} p(o \mid q, \lambda) P(q \mid l, \lambda)$$

(state-output) (state-transition)

$$p(o \mid q, \lambda) = \prod_{t=1}^{T} \mathcal{N}(o_t \mid \mu_{q_t}, \Sigma_{q_t}) \quad \Leftarrow \text{single Gaussian}$$

$$= \mathcal{N} \left( \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_T \end{bmatrix} ; \begin{bmatrix} \mu_{q_1} \\ \mu_{q_2} \\ \vdots \\ \mu_{q_T} \end{bmatrix} , \begin{bmatrix} \Sigma_{q_1} & \Sigma_{q_2} & \cdots & 0 \\ 0 & \Sigma_{q_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{q_T} \end{bmatrix} \right)$$

$$= \mathcal{N}(o \mid \mu_q, \Sigma_q)$$

TOSHIBA
Leading Innovation
Inconsistency in HMM w/ dynamic features

Under \( o = Wc \)

\[
p(o \mid q, \lambda) = \mathcal{N}(Wc ; \mu_q, \Sigma_q) \Rightarrow \text{incorrect!}
\]

Why?

\[
\int_c \mathcal{N}(Wc ; \mu_q, \Sigma_q) dc \neq 1 \Rightarrow \text{integral over } c \text{ must be 1 to be a valid PDF}
\]

Why does this happen?

Static features \( \Rightarrow \) random variables

Dynamic features \( \Rightarrow \) Not random variables!!
Normalization

Normalized to achieve valid PDF

\[ Z_q = \int_c \mathcal{N}(Wc; \mu_q, \Sigma_q) \, dc \]
\[ = \frac{\sqrt{(2\pi)^{MT} |P_q|}}{\sqrt{(2\pi)^{2MT} |\Sigma_q|}} \exp \left\{ -\frac{1}{2} \left( \mu_q^\top \Sigma_q^{-1} \mu_q - r_q^\top P_q r_q \right) \right\} \]

\[ \mathcal{N}(Wc; \mu_q, \Sigma_q) \Rightarrow \text{invalid PDF!} \]
\[ \frac{1}{Z_q} \mathcal{N}(Wc; \mu_q, \Sigma_q) \Rightarrow \text{valid PDF!!} \]
Definition of trajectory HMM

Use normalized Gaussian \(\Rightarrow\) trajectory HMM is defined

\[
p(c \mid l, \lambda) = \sum_{q} p(c \mid q, \lambda) P(q \mid l, \lambda)
\]

\[
p(c \mid q, \lambda) = \frac{1}{Z_q} \mathcal{N}(Wc ; \mu_q, \Sigma_q) \leftarrow \text{normalized Gaussian}
\]

\[
= \mathcal{N}(c ; \bar{c}_q, P_q) \leftarrow \text{Gaussian over } c
\]

\[
R_q \bar{c}_q = r_q
\]

\[
R_q = W^\top \Sigma_q^{-1} W = P_q^{-1}
\]

\[
r_q = W^\top \Sigma_q^{-1} \mu_q
\]
Mean & variance

⇒ varies in a state

Frame correlation

⇒ captured by $P_q$

Covariance matrix $P_q$
Trajectory HMM & speech parameter generation

Mean vector of trajectory HMM

\[ W^\top \Sigma_q^{-1} W \bar{c}_q = W^\top \Sigma_q^{-1} \mu_q \]

Trajectory by speech parameter generation algorithm

\[ W^\top \Sigma_q^{-1} W \hat{c} = W^\top \Sigma_q^{-1} \mu_q \]

⇒ they are identical
Trajectory HMM as globally normalized model

**HMM ⇒ locally (frame-level) normalized model**

\[
p(o \mid q, \lambda) = \prod_{t=1}^{T} p(o_t \mid q_t, \lambda)
\]

\[
= \prod_{t=1}^{T} \mathcal{N}(o_t ; \mu_{q_t}, \Sigma_{q_t})
\]

**Trajectory HMM ⇒ globally (utt-level) normalized model**

\[
p(c \mid q, \lambda) = \frac{1}{Z_q} \mathcal{N}(Wc ; \mu_q, \Sigma_q)
\]

\[
= \frac{1}{Z_q} \prod_{t=1}^{T} \mathcal{N} \left( \begin{bmatrix} c_t^\top & \Delta c_t^\top \end{bmatrix}^\top ; \mu_{q_t}, \Sigma_{q_t} \right)
\]
Estimating trajectory HMM parameters

**ML estimation of trajectory HMM**

\[ \hat{\lambda} = \arg \max_\lambda p(c \mid l, \lambda) \]

**Locally normalized model**

Parameter estimation for each state can be done separately

**Globally normalized model**

Parameter estimation of all states have to be done jointly

\[ \mu = [\mu_1^\top, \mu_2^\top, \ldots, \mu_N^\top]^\top : \text{all mean vectors} \]

\[ \phi = [\Sigma_1^{-1}, \Sigma_2^{-1}, \ldots, \Sigma_N^{-1}] : \text{all precision matrices} \]
Parameter update formulae

\[ \sum_{q} p(q \mid c, \lambda') S_q^{\top} \Sigma_q^{-1} WP_q W^{\top} \Sigma_q^{-1} S_q \mu \]

\[ = \sum_{q} p(q \mid c, \lambda') S_q^{\top} \Sigma_q^{-1} W c \]

mean vectors \Rightarrow closed form

\[ \frac{\partial Q(\lambda, \lambda')}{\partial \phi} = \sum_{q} p(q \mid c, \lambda') \left\{ \frac{1}{2} S_q^{\top} \text{diag}^{-1} \left( WP_q W^{\top} - W cc^{\top} W^{\top} \right) + W \bar{c}_q \bar{c}_q^{\top} W^{\top} + \mu_q c^{\top} W^{\top} + W c \mu_q^{\top} - \mu_q \bar{c}_q^{\top} W^{\top} - W \bar{c}_q \mu_q^{\top} \right\} \]

covariance matrices \Rightarrow numerical optimization
Drawback of trajectory HMM training

**Exact EM is intractable**
- Computing posterior prob of $q$ is intractable
- Single-path (Viterbi) or Monte Carlo approximation

**Exact tree-based clustering is also intractable**
- Splitting one nodes affects the other nodes
- Trees built for HMMs are often used

**Computationally & memory intensive**
- High dimensional matrix operations
- Numerical optimization
Effect of parameter reestimation

Training data
- Mean sequence of the HMM
- Mean sequence of the trajectory HMM (w/o update)
- Mean sequence of the trajectory HMM (with update)
Outline

Trajectory HMM
- Speech parameter generation
- Derivation of trajectory HMM
- Relationship between parameter generation & trajectory HMM
- Trajectory HMM as globally normalized model
- Parameter estimation

Min generation error (MGE) training & trajectory HMM
- Relationship
- Properties of MGE

(If time remains) Product of Experts (PoE)
- Combination of multiple AMs as PoE
- PoE & trajectory HMM
ML training & MGE training w/ Euclidean dist \([Wu;'06]\)

\[\hat{\lambda}_{ML} = \arg \max_{\lambda} p(c \mid q, \lambda)\]

\[= \arg \max_{\lambda} \mathcal{N}(c ; \bar{c}_q, P_q)\]

\[\hat{\lambda}_{MGE} = \arg \min_{\lambda} \mathcal{E}(c ; q, \lambda)\]

\[= \arg \min_{\lambda} \|c - \bar{c}_q\|_2 \quad \Leftarrow \text{MMSE estimation}\]

\[= \arg \max_{\lambda} \mathcal{N}(c ; \bar{c}_q, I) \quad \Leftarrow \text{Identity covariance matrix}\]
Performance of ML & MGE w/ Euc is similar, why?

⇒ Due to speech parameter generation algorithm

\[
\hat{c}_{\text{ML}} = \arg\max_c p \left( c \mid \hat{q}, \lambda_{\text{ML}} \right) \\
= \arg\max_c \mathcal{N} \left( c \mid \bar{c}_q, P_{\hat{q}} \right) \\
= \bar{c}_q
\]

\[
\hat{c}_{\text{MGE}} = \arg\max_c p \left( c \mid \hat{q}, \lambda_{\text{MGE}} \right) \\
= \arg\max_c \mathcal{N} \left( c \mid \bar{c}_q, I \right) \\
= \bar{c}_q
\]
Random sampling from ML & MGE w/ Euc distance

**ML**

\[ \tilde{c}_{\text{ML}} \sim \mathcal{N} (\bar{c}_q, P_q) \]

⇒ Temporal correlations will be kept

**MGE**

\[ \tilde{c}_{\text{MGE}} \sim \mathcal{N} (\bar{c}_q, I) \]

⇒ Temporal correlations will be discarded
MGE training & trajectory HMM (5)

Which is better, ML or MGE?

- w/ parameter generation, MGE is more reasonable
  * MGE $\Rightarrow \mu$ & $\Sigma$ to represent mean trajectory
  * ML $\Rightarrow \mu$ for mean trajectory, $\Sigma$ for mean trj & temporal cov
  $\Rightarrow$ MGE can focus on modeling mean trajectory

- w/ random sampling, ML is more reasonable
  * MGE ignores temporal correlations
  * ML models temporal correlations
Summary

**Trajectory HMM**
- Derived from HMM w/ dynamic feature constraints
- Can be viewed as a globally normalized model
- All states need to be estimated jointly
- Generated params = mean vector of trajectory HMM

**MGE training**
- MGE w/ Euclid distance = MMSE estimation of trajectory HMM
- w/ speech parameter generation algorithm (ML parm gen)
  ⇒ ML & MGE work similarly
- w/ random sampling
  ⇒ MGE won't work well
Outline

Trajectory HMM
- Speech parameter generation
- Derivation of trajectory HMM
- Relationship between parameter generation & trajectory HMM
- Trajectory HMM as globally normalized model
- Parameter estimation

Min generation error (MGE) training & trajectory HMM
- Relationship
- Properties of MGE

(If time remains) Product of Experts (PoE)
- Combination of multiple AMs as PoE
- PoE & trajectory HMM
Combination of multiple acoustic models

Combine multiple AMs to reduce over-smoothing

* Training; estimate multiple-level AMs \textit{individually}

\[ \hat{\lambda}_i = \operatorname{arg\ max}_{\lambda_i} p \left( f_i(c) \mid \lambda_i \right) \quad i = 1, \ldots, M \]

* Synthesis; generate \( c \) that \textit{jointly} maximize output probs from AMs

\[ \hat{c} = \operatorname{arg\ max}_c \sum_{i=1}^{M} \alpha_i \log p(f_i(c) \mid \hat{\lambda}_i) \]

* Feature function, \( f_i(c) \), extracts acoustic feats for \( i \)-th AM from \( c \)
  - e.g., dynamic feats, DCT, average, summation, global variance

* Parameters of AMs, \( \lambda_i \), are trained \textit{independently}

  \[ \rightarrow \text{Use weights to control balance among AMs} \]

* Weights, \( \alpha_i \), are determined by \textit{held-out data} (or tuned manually)
Mixture model vs Product model

**Mixture of experts**

\[ p(c | \lambda_1, \ldots, \lambda_M) = \frac{1}{Z} \sum_{i=1}^{M} \alpha_i p(f_i(c) | \lambda_i) \]

* Data is generated from *union* of experts
* Robust for modeling data with many variations
* GMM → Mixture of Gaussians

**Product of experts** [Hinton;'02]

\[ p(c | \lambda_1, \ldots, \lambda_M) = \frac{1}{Z} \prod_{i=1}^{M} \{p(f_i(c) | \lambda_i)\}^{\alpha_i} \]

* Data is generated from *intersection* of experts
* Efficient for modeling data with many constraints
* PoG → Product of Gaussians
Combination of multiple AMs can be viewed as PoE

\[ \hat{c} = \arg \max_c p(c \mid \lambda_1, \ldots, \lambda_M) = \arg \max_c \frac{1}{Z} \prod_{i=1}^{M} \{p(f_i(c) \mid \lambda_i)\}^{\alpha_i} \]

\[ = \arg \max_c \prod_{i=1}^{M} \{p(f_i(c) \mid \lambda_i)\}^{\alpha_i} = \arg \max_c \sum_{i=1}^{M} \alpha_i \log p(f_i(c) \mid \lambda_i) \]

* Generating \( c \) from combination of multiple AMs
  → Equivalent to generating \( c \) from PoE consisting of AMs

* Regarding combination of multiple AMs as PoE
  → \textit{Jointly} estimate multiple AMs

\[ \{\hat{\lambda}_1, \ldots, \hat{\lambda}_M\} = \arg \max_{\lambda_1, \ldots, \lambda_M} \frac{1}{Z} \prod_{i=1}^{M} \{p(f_i(c) \mid \lambda_i)\}^{\alpha_i} \]
Product of Gaussians

Product of Gaussians (PoG)

\[ p(c | \lambda_1, \ldots, \lambda_M) = \frac{1}{Z} \prod_{i=1}^{M} \mathcal{N}(f_i(c) ; \mu_i, \Sigma_i) \]

* Special case of PoE; All experts are Gaussian
* If all feature functions are linear
  - PoG also becomes Gaussian
  - Normalization constant

\[ Z = \int \prod_{i=1}^{M} \mathcal{N}(f_i(c) ; \mu_i, \Sigma_i) \, dc \]

can be computed in **closed form**
Trajectory HMM as product of Gaussians

**Trajectory HMM can be viewed as PoG** [Williams;'05, Zen;'07]

\[
p(c \mid \lambda) = \sum_{q} p(c \mid q, \lambda) p(q \mid \lambda)
\]

\[
p(c \mid q, \lambda) = \mathcal{N}(c; \bar{c}_q, P_q) = \frac{1}{Z_q} \mathcal{N}(Wc; \mu_q, \Sigma_q)
\]

\[
\begin{align*}
\mathcal{N} & (\mu_q, \Sigma_q) & \quad Wc & \quad W & \quad c \\
\vdots & & \vdots & \vdots & \vdots \\
\mu_{q_{t-1}}^{(0)} & \Sigma_{q_{t-1}}^{(0)} & \rightarrow & c_{t-1} & 0 & I & 0 & 0 & \ldots \\
\mu_{q_t}^{(1)} & \Sigma_{q_t}^{(1)} & \rightarrow & \Delta c_{t-1} & -I & I & 0 & 0 & \ldots \\
\mu_{q_{t+1}}^{(0)} & \Sigma_{q_{t+1}}^{(0)} & \rightarrow & c_t & 0 & 0 & I & 0 & \ldots \\
\mu_{q_{t+1}}^{(1)} & \Sigma_{q_{t+1}}^{(1)} & \rightarrow & \Delta c_t & 0 & -I & I & 0 & \ldots \\
\mu_{q_{t+2}}^{(0)} & \Sigma_{q_{t+2}}^{(0)} & \rightarrow & c_{t+1} & 0 & 0 & 0 & I & \ldots \\
\mu_{q_{t+2}}^{(1)} & \Sigma_{q_{t+2}}^{(1)} & \rightarrow & \Delta c_{t+1} & 0 & 0 & 0 & -I & I & \ldots \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{align*}
\]
Trajectory HMM as product of Gaussians

Trajectory HMM can be viewed as PoG [Williams;'05, Zen;'07]

\[ p(c \mid \lambda) = \sum_{q} p(c \mid q, \lambda) p(q \mid \lambda) \]

\[ p(c \mid q, \lambda) = \mathcal{N}(c; \bar{c}_q, P_q) = \frac{1}{Z_q} \mathcal{N}(Wc; \mu_q, \Sigma_q) \]

\[ = \frac{1}{Z_q} \prod_{t=1}^{T} \prod_{d=0}^{2} \mathcal{N}\left(f_t^{(d)}(c); \mu_{qt}^{(d)}, \Sigma_{qt}^{(d)}\right) \]

\[ f_t^{(d)}(c) : d\text{-th dyn feat at frame } t \]

\[ Z_q = \int \prod_{t=1}^{T} \prod_{d=0}^{2} \mathcal{N}\left(f_t^{(d)}(c); \mu_{qt}^{(d)}, \Sigma_{qt}^{(d)}\right) dc \]

* Experts are Gaussians, feature functions are dynamic features
* Gaussian experts are multiplied over time
Linear feature function with Gaussian experts

Combining multiple AMs as PoE

* Multiple-level AMs often use linear feature functions w/ Gaussians
  - DCT [Latorre;'08, Qian;'09], average [Wang;'08], sum [Ling;'06, Gao;'08]
* PoEs become the same form as trajectory HMM
  → Training algorithm for trajectory HMM are applicable

Example: state & phoneme duration models [Ling;'06]

\[
p(d | \lambda) = \frac{1}{Z} \mathcal{N}(Wd; \mu, \Sigma)
= \frac{1}{Z} \prod_{i=1}^{P} \prod_{j=1}^{N_i} \mathcal{N}(d_{ij}; \xi_{ij}, \sigma_{ij}) \times \prod_{k=1}^{P} \mathcal{N}(p_k; \nu_k, \omega_k)
\]

\[
d_{ij} : \text{duration of state } j \text{ in phoneme } i
\]
General PoE (non-linear feat or non-Gaussian)

General form of PoE

\[ p (c \mid \lambda_1, \ldots, \lambda_M) = \frac{1}{Z} \prod_{i=1}^{M} \{p (f_i(c) \mid \lambda_i)\}^{\alpha_i} \]

* Feature functions can be non-linear, experts can be non-Gaussian
* Normalization term has no closed form
* Training is complex, usually normalization term is approximated

Example: **global variance (GV)** [Toda;'07]

\[ p (c \mid q, \lambda, \lambda_{GV}) = \frac{1}{Z_q} \mathcal{N} (c ; \bar{c}_q, P_q)^{\alpha} \mathcal{N} (f_v(c) ; \mu_v, \Sigma_v) \]

\[ f_v(c) = \frac{1}{T} \sum_{t=1}^{T} \text{diag} \left[ (c_t - \bar{c})(c_t - \bar{c})^\top \right] : \text{intra-utt variance, quadratic} \]
Contrastive divergence learning [Hinton;'02]

* Training algorithm for general PoE
* Combination of sampling & gradient methods

1. Draw $J$ samples from PoE

$$c^{(j)} \sim p(c \mid \lambda) \quad j = 1, \ldots, J \quad \lambda = \{\lambda_1, \ldots, \lambda_M\}: \text{PoE model params}$$

2. Compute approximated derivative of log likelihood w.r.t. $\lambda$

$$\frac{\partial \log p(c \mid \lambda)}{\partial \lambda} \approx \left\langle \frac{\partial \log p(c \mid \lambda)}{\partial \lambda} \right\rangle_p^0 - \left\langle \frac{\partial \log p(c \mid \lambda)}{\partial \lambda} \right\rangle_p^J$$

expectation over data expectation over samples

3. Update model params using gradient method

$$\lambda' = \lambda - \eta \cdot \left( \left\langle \frac{\partial \log p(c \mid \lambda)}{\partial \lambda} \right\rangle_p^0 - \left\langle \frac{\partial \log p(c \mid \lambda)}{\partial \lambda} \right\rangle_p^J \right)$$

$$\lambda = \lambda'$$

4. Iterate 1-3 until converge
Experimental conditions

* Training data; 2,469 utterances
* Development data; 137 utterances
  - Used to optimize weights in conventional method
  - Weights were optimized to minimize RMSE by grid search
  - Baseline & proposed method did not use development data
* Almost the same training setup as Nitech-HTS 2005 [Zen;'06]
* Test data; 137 utterances
* State, phone, & syllable-level models were clustered individually
  - # of leaf nodes
    * state; 607, phoneme; 1,364, syllable; 281
## Experimental Results

### Duration prediction results (RMSE in frame (rel imp))

<table>
<thead>
<tr>
<th>Model</th>
<th>Phoneme</th>
<th>Syllable</th>
<th>Pause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (st)</td>
<td>5.08 (ref)</td>
<td>8.98 (ref)</td>
<td>35.0 (ref)</td>
</tr>
<tr>
<td>uPoE (st*ph)</td>
<td>4.62 (9.1%)</td>
<td>8.13 (9.5%)</td>
<td>31.8 (9.1%)</td>
</tr>
<tr>
<td>uPoE (st<em>ph</em>syl)</td>
<td>4.62 (9.1%)</td>
<td>8.11 (9.7%)</td>
<td>31.8 (9.1%)</td>
</tr>
<tr>
<td>PoE (st*ph)</td>
<td>4.60 (9.4%)</td>
<td>8.04 (10.5%)</td>
<td>31.9 (8.9%)</td>
</tr>
<tr>
<td>PoE (st<em>ph</em>syl)</td>
<td>4.57 (10.0%)</td>
<td>8.02 (10.7%)</td>
<td>31.9 (8.9%)</td>
</tr>
</tbody>
</table>

st; state only,  st*ph; state & phoneme,  st*ph*syl; state, phoneme, & syllable
uPoE; individually trained multiple-level duration models with optimized weights
PoE; jointly estimated multiple-level duration models
Experiment - Global Variance as PoE

Experimental conditions

* Training data; 2,469 utterances
  - Training data was split into mini-batch (250 utterances)
  - 10 MCMC sampling at each contrastive divergence learning
    * Hybrid Monte Carlo with 20 leap-frog steps
    * Leap-frog size was adjusted adaptively
  - Learning rate was annealed at every 2,000 iterations
  - Momentum method was used to accelerate learning
  - Context-dependent logarithmic GV w/o silence was used

* Test sentences; 70 sentences
  - Paired comparison test, # of subjects 7 (native English speaker)
  - 30 sentences per subject
Experimental Results

Paired comparison test result

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>PoE</th>
<th>No preference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>17.1</td>
<td>32.4</td>
<td>50.5</td>
</tr>
</tbody>
</table>

Baseline; conventional (not jointly estimated) GV
PoE; proposed (jointly estimated) GV

Difference was statistically significant at \( p < 0.05 \) level
Summary

Statistical parametric synthesis based on PoE
- Combination of multiple-level AMs is formulated as PoE
- Jointly estimate multiple-level AMs as PoE
  * Linear feature function with Gaussian experts
    → Can be estimated in the same way as trajectory HMM
  * Non-linear feature function and/or non-Gaussian experts
    → Contrastive divergence learning
- Experiments
  * Jointly estimating multiple AMs as PoE improved performance
References

[Williams;'05] C. Williams, "How to pretend that correlated variables are independent by using difference observations," Neural Computation, vol. 17, no. 1, pp. 1--6, 2005.