Statistical Parametric Speech Synthesis Based on Product of Experts

Heiga Zen & Mark Gales

TOSHIBA
Leading Innovation

Yoshihiko Nankaku & Keiichi Tokuda

Background

Statistical parametric synthesis based on HMMs

- **Advantage**
  - * Flexibility to change voice characteristics
  - * Small footprint
  - * Robustness
- **Limitation**
  - * Quality
- **Major factors of quality degradation**
  - * Vocoding
  - * Accuracy of acoustic models (AMs)
  - * Over-smoothing

This work addresses accuracy of AMs & over-smoothing

Combination of Multiple-Level Acoustic Models

Combine multiple AMs to reduce over-smoothing

* Training; estimate multiple-level AMs ***individually***

$$\hat{\lambda}_i = \arg\max_{\lambda_i} p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right) \quad i = 1, \ldots, M$$

* Synthesis; generate $\boldsymbol{c}$ that ***jointly*** maximize output probs from AMs

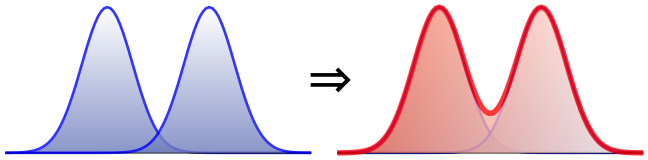
$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}} \sum_{i=1}^{M} \alpha_i \log p(f_i(\boldsymbol{c}) \mid \hat{\lambda}_i)$$

* Feature function, $f_i(\boldsymbol{c})$, extracts acoustic feats for i-th AM from $\boldsymbol{c}$
  - e.g., dynamic feats, DCT, average, summation, global variance
* Parameters of AMs, λ_i , are trained ***independently***
  → Use weights to control balance among AMs
* Weights, α_i , are determined by ***held-out data*** (or tuned manually)

Mixture Model vs Product Model

Mixture of experts

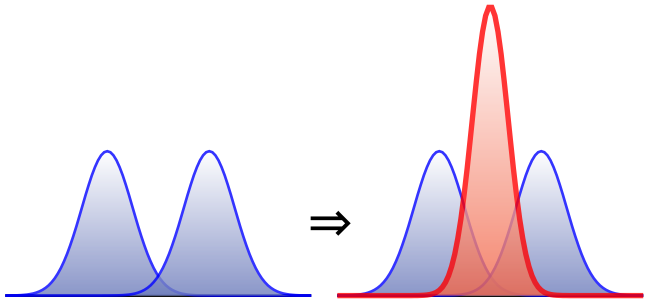
$$p(\boldsymbol{c} \mid \lambda_1, \ldots, \lambda_M) = \frac{1}{Z} \sum_{i=1}^{M} \alpha_i p(f_i(\boldsymbol{c}) \mid \lambda_i)$$



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

Product of experts [Hinton;'02]

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



* Data is generated from ***intersection*** of experts
* Efficient for modeling data with many constraints
* PoG → Product of Gaussians

Combination of Multiple-Level AMs as PoE

Comibination of multiple AMs can be viewed as PoE

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}} p\left(\boldsymbol{c} \mid \lambda_1, \ldots, \lambda_M\right) = \arg\max_{\boldsymbol{c}} \frac{1}{Z} \prod_{i=1}^{M} \left\{p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right)\right\}^{\alpha_i}$$

$$= \arg\max_{\boldsymbol{c}} \prod_{i=1}^{M} \left\{p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right)\right\}^{\alpha_i} = \arg\max_{\boldsymbol{c}} \sum_{i=1}^{M} \alpha_i \log p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right)$$

* Generating $\boldsymbol{c}$ from combination of multiple AMs

  → Equivalent to generating $\boldsymbol{c}$ from PoE consisting of AMs

* Regarding combination of multiple AMs as PoE

  → ***Jointly*** estimate multiple AMs

$$\{\hat{\lambda}_1, \ldots, \hat{\lambda}_M\} = \arg \max_{\lambda_1, \ldots, \lambda_M} \frac{1}{Z} \prod_{i=1}^{M} \left\{p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right)\right\}^{\alpha_i}$$

Combination of Multiple-Level AMs as PoE

Comibination of multiple AMs can be viewed as PoE

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}} p\left(\boldsymbol{c} \mid \lambda_1, \ldots, \lambda_M\right) = \arg\max_{\boldsymbol{c}} \frac{1}{Z} \prod_{i=1}^{M} \left\{p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right)\right\}^{\alpha_i}$$

$$= \arg\max_{\boldsymbol{c}} \prod_{i=1}^{M} \left\{p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right)\right\}^{\alpha_i} = \arg\max_{\boldsymbol{c}} \sum_{i=1}^{M} \alpha_i \log p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right)$$

* Generating $\boldsymbol{c}$ from combination of multiple AMs

  → Equivalent to generating $\boldsymbol{c}$ from PoE consisting of AMs

* Regarding combination of multiple AMs as PoE

  → ***Jointly*** estimate multiple AMs

  - AMs become ***complementary***

  - ***held-out data*** to estimate weights is no longer required

Product of Gaussians

Product of Gaussians (PoG)

$$p(\boldsymbol{c} \mid \lambda_1, \ldots, \lambda_M) = \frac{1}{Z} \prod_{i=1}^{M} \mathcal{N}\left(f_i(\boldsymbol{c}) \; ; \; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)$$

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

$$Z = \int \prod_{i=1}^{M} \mathcal{N}\left(f_i(\boldsymbol{c}) \; ; \; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) d\boldsymbol{c}$$

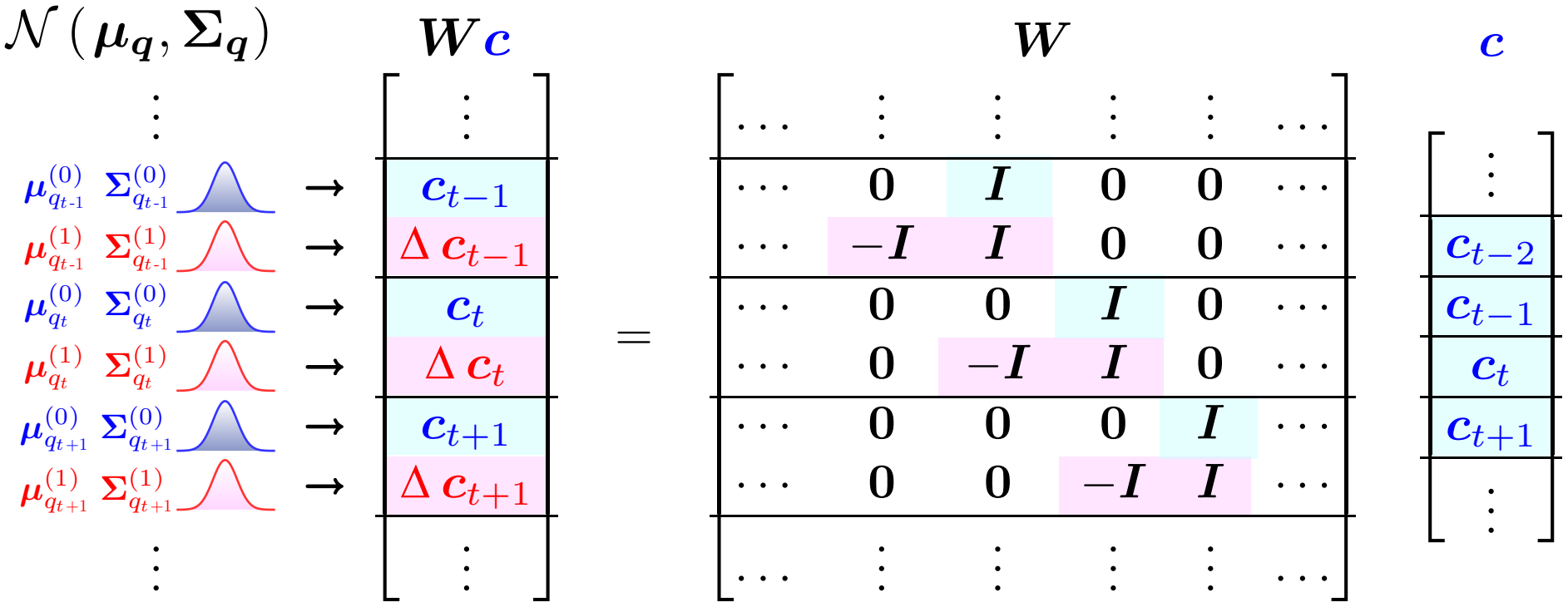
can be computed in ***closed form***

Trajectory HMM as Product of Gaussians

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

$$p(\boldsymbol{c} \mid \lambda) = \sum_{\forall \boldsymbol{q}} p(\boldsymbol{c} \mid \boldsymbol{q}, \lambda)\, p(\boldsymbol{q} \mid \lambda)$$

$$p(\boldsymbol{c} \mid \boldsymbol{q}, \lambda) = \mathcal{N}(\boldsymbol{c};\ \bar{\boldsymbol{c}}_{\boldsymbol{q}}, \boldsymbol{P}_{\boldsymbol{q}}) = \frac{1}{Z_{\boldsymbol{q}}} \mathcal{N}(\boldsymbol{W}\boldsymbol{c};\ \boldsymbol{\mu}_{\boldsymbol{q}}, \boldsymbol{\Sigma}_{\boldsymbol{q}})$$



Trajectory HMM as Product of Gaussians

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

$$p(\boldsymbol{c} \mid \lambda) = \sum_{\forall \boldsymbol{q}} p(\boldsymbol{c} \mid \boldsymbol{q}, \lambda)\, p(\boldsymbol{q} \mid \lambda)$$

$$p(\boldsymbol{c} \mid \boldsymbol{q}, \lambda) = \mathcal{N}(\boldsymbol{c};\ \bar{\boldsymbol{c}}_{\boldsymbol{q}}, \boldsymbol{P}_{\boldsymbol{q}}) = \frac{1}{Z_{\boldsymbol{q}}} \mathcal{N}(\boldsymbol{W}\boldsymbol{c};\ \boldsymbol{\mu}_{\boldsymbol{q}}, \boldsymbol{\Sigma}_{\boldsymbol{q}})$$

$$= \frac{1}{Z_{\boldsymbol{q}}} \prod_{t=1}^{T} \prod_{d=0}^{2} \mathcal{N}\left(f_t^{(d)}(\boldsymbol{c});\ \boldsymbol{\mu}_{q_t}^{(d)}, \boldsymbol{\Sigma}_{q_t}^{(d)}\right)$$

$$Z_q = \int \prod_{t=1}^{T} \prod_{d=0}^{2} \mathcal{N}\left(f_t^{(d)}(\boldsymbol{c});\ \boldsymbol{\mu}_{q_t}^{(d)}, \boldsymbol{\Sigma}_{q_t}^{(d)}\right) d\boldsymbol{c}$$

$f_t^{(d)}(\boldsymbol{c})$: d-th dyn feat at frame t

* Experts are Gaussians, feature functions are dynamic features
* Gaussian experts are multiplied over time
* Training/adaptation/search algorithms have been derived

Linear Feature Function with Gaussian Experts

Combining multiple-level 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]

$$p(\boldsymbol{d} \mid \lambda) = \frac{1}{Z}\mathcal{N}(\boldsymbol{W}\boldsymbol{d}\ ;\ \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$= \frac{1}{Z}\prod_{i=1}^{P}\prod_{j=1}^{N_i}\mathcal{N}(d_{ij}\ ;\ \xi_{ij}, \sigma_{ij})$$

$$\cdot \prod_{k=1}^{P}\mathcal{N}(p_k\ ;\ \nu_k, \omega_k)$$

d_{ij} : duration of state j in phoneme i

$$\begin{bmatrix} d_{11} \\ d_{12} \\ d_{13} \\ p_1 \\ \hline d_{21} \\ d_{22} \\ d_{23} \\ p_2 \\ \hline \vdots \end{bmatrix} = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 0 & & & & \dots \\ 0 & 1 & 0 & & \mathbf{0} & & \dots \\ 0 & 0 & 1 & & & & \dots \\ 1 & 1 & 1 & & & & \dots \\ \hline & & & 1 & 0 & 0 & \dots \\ & \mathbf{0} & & 0 & 1 & 0 & \dots \\ & & & 0 & 0 & 1 & \dots \\ & & & 1 & 1 & 1 & \dots \\ \hline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots \end{array}\right] \begin{bmatrix} d_{11} \\ d_{12} \\ d_{13} \\ \hline d_{21} \\ d_{22} \\ d_{23} \\ \hline \vdots \end{bmatrix}$$

$\boldsymbol{W}$ $\boldsymbol{d}$

General PoE (Non-Linear/Non-Gaussian)

General form of PoE

$$p\left(\boldsymbol{c} \mid \lambda_1, \ldots, \lambda_M\right) = \frac{1}{Z} \prod_{i=1}^{M} \left\{p\left(f_i(\boldsymbol{c}) \mid \lambda_i\right)\right\}^{\alpha_i}$$

* 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]

$$p\left(\boldsymbol{c} \mid \boldsymbol{q}, \lambda, \lambda_{\mathrm{GV}}\right) = \frac{1}{Z_{\boldsymbol{q}}} \mathcal{N}\left(\boldsymbol{c}\,;\, \bar{\boldsymbol{c}}_{\boldsymbol{q}}, \boldsymbol{P}_{\boldsymbol{q}}\right)^{\alpha} \mathcal{N}\left(f_v(\boldsymbol{c})\,;\, \boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v\right)$$

$$f_v(\boldsymbol{c}) = \frac{1}{T} \sum_{t=1}^{T} \mathrm{diag}\left[\left(\boldsymbol{c}_t - \bar{\boldsymbol{c}}\right)\left(\boldsymbol{c}_t - \bar{\boldsymbol{c}}\right)^{\top}\right]$$: intra-utt variance, ***quadratic***

Contrastive Divergence Learining

Contrastive divergence learning [Hinton;'02]

* Training algorithm for general PoE

* Combination of sampling & gradient methods

1. Draw J samples from PoE

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

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

$$\frac{\partial \log p(\boldsymbol{c} \mid \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \approx \left\langle \frac{\partial \log p(\boldsymbol{c} \mid \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \right\rangle_{p_0} - \left\langle \frac{\partial \log p(\boldsymbol{c} \mid \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \right\rangle_{p_J}$$

expectation over data expectation over samples

3. Update model params using gradient method

$$\boldsymbol{\lambda}' = \boldsymbol{\lambda} - \eta \cdot \left(\left\langle \frac{\partial \log p(\boldsymbol{c} \mid \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \right\rangle_{p_0} - \left\langle \frac{\partial \log p(\boldsymbol{c} \mid \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} \right\rangle_{p_j} \right)$$

η : learning rate

$\boldsymbol{\lambda} = \boldsymbol{\lambda}'$

4. Iterate 1-3 until converge

Experiment - Multiple-Level Dur Models as PoE

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))

Model	Phoneme	Syllable	Pause
Baseline (st)	5.08 (ref)	8.98 (ref)	35.0 (ref)
uPoE (st*ph)	4.62 (9.1%)	8.13 (9.5%)	**31.8 (9.1%)**
uPoE (st*ph*syl)	4.62 (9.1%)	8.11 (9.7%)	**31.8 (9.1%)**
PoE (st*ph)	4.60 (9.4%)	8.04 (10.5%)	31.9 (8.9%)
PoE (st*ph*syl)	**4.57 (10.0%)**	**8.02 (10.7%)**	31.9 (8.9%)

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

Baseline	PoE	No preference
17.1	32.4	50.5

Baseline; conventional (not jointly estimated) GV

PoE; proposed (jointly estimated) GV

Difference was statistically significant at $p<0.05$ level

Conclusions

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

Future plans

- **Investigate other feature functions/experts**
- **Make contrastive divergence learning faster**

References

[Hinton;'02] G. Hinton, "Training product of experts by minimizing contrastive divergence," Neural Computation, vol. 14, no. 8, pp. 1771-1800, 2002.

[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.

[Zen;'07] H. Zen, et al., "Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences," Computer Speech & Language, vol. 21, no. 1, pp.153-173, 2007.

[Latorre;'08] J. Latorre & M. Akamine, "Multilevel parametric-base F0 model for speech synthesis," Proc. Interspeech, pp. 2274--2277, 2008.

[Qian;'09] Y. Qian, et al., "Improved prosody generation by maximizing joint likelihood of state and longer units," Proc. ICASSP, pp. 1173--1176, 2009.

[Wang;'08] C.-C. Wang, et al., "Multi-layer F0 modeling for HMM-based speech synthesis," Proc. ISCSLP, pp. 129--132, 2008.

[Ling;'06] Z.-H. Ling, et al., "USTC system for Blizzard Challenge 2006 an improved HMM-based speech synthesis method," Proc. Blizzard Challenge Workshop, 2006.

[Gao;'08] B.-H. Gao, et al., "Duration refinement by jointly optimizing state and longer unit likelihood," Proc. Interspeech, 2266--2269, 2008.

[Toda;'07] T. Toda & K. Tokuda, "A speech parameter generation algorithm considering global variance for HMM-based speech synthesis," IEICE Trans. Inf. Syst., vol. E90-D, no. 5, pp. 816--824, 2007.