Model-Based Approaches to Robust Speech Recognition

Mark Gales with Federico Flego and Rogier van Dalen

(work partly funded by Toshiba Research Europe Ltd)

3rd October 2008

Cambridge University Engineering Department

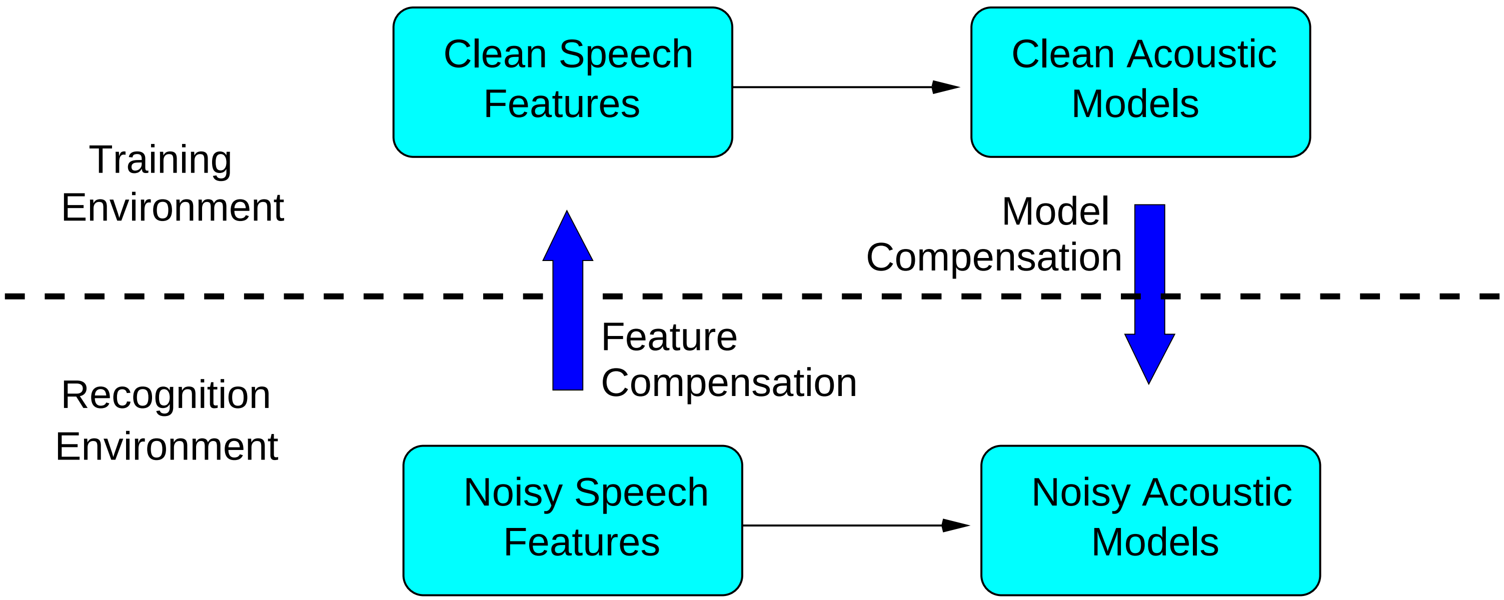
Edinburgh University Seminar

Overview

- Noise robust speech recognition
  - adaptive approaches
  - forms of the mismatch function
- Model based noise compensation
  - vector-Taylor series approximation (VTS)
  - joint uncertainty decoding (JUD)
- Predictive linear transforms
- Support Vector Machines (SVM)
  - adapting SVMs to handle noise
- Experimental results
  - Aurora2 continuous digit task
  - Toshiba in-car task

Example Application - In-Car Navigation

Noise Compensation Approaches



- Two main approaches:
  - feature compensation: "clean" the noisy features $\boldsymbol{y}_t \longrightarrow \boldsymbol{x}_t$
  - model compensation: "corrupt" the clean models $\boldsymbol{\lambda}_{\rm x} \longrightarrow \boldsymbol{\lambda}_{\rm y}$
- This work concentrates on model compensation approaches

"Adaptive" Linear Model Compensation

- A standard scheme for speaker/environment adaptation is linear transforms

  Various forms used:

  - MLLR Mean: $\boldsymbol{\mu}_{\text{y}}^{(m)} = \mathbf{A}\boldsymbol{\mu}_{\text{x}}^{(m)} + \mathbf{b}$
  - MLLR Variance: $\boldsymbol{\Sigma}_{\text{y}}^{(m)} = \mathbf{A}\boldsymbol{\Sigma}_{\text{x}}^{(m)}\mathbf{A}^{\mathsf{T}}$
  - CMLLR: $\boldsymbol{x}_t = \mathbf{A}\boldsymbol{y}_t + \mathbf{b}$ (MLLR mean/variance transforms same)

- General approach, but large numbers of model parameters

  - a single full-transform has about 1560 parameters to train
  - the impact of noise is non-linear, so many transforms useful

- Model compensation specifically aimed at noise robust speech recognition

  - only need to estimate a model of the acoustic environment
  - but need to model impact of the environment on the clean speech
  - examples PMC, VTS and JUD

Constrained MLLR

- CMLLR is a popular form of "adaptive" linear transforms
  - decoding implemented as

$$p(\boldsymbol{y}_t|\mathbf{s}_m) = |\mathbf{A}^{(r)}|\mathcal{N}(\mathbf{A}^{(r)}\boldsymbol{y}_t + \mathbf{b}^{(r)}; \boldsymbol{\mu}_{\mathtt{x}}^{(m)}, \boldsymbol{\Sigma}_{\mathtt{x}}^{(m)})$$

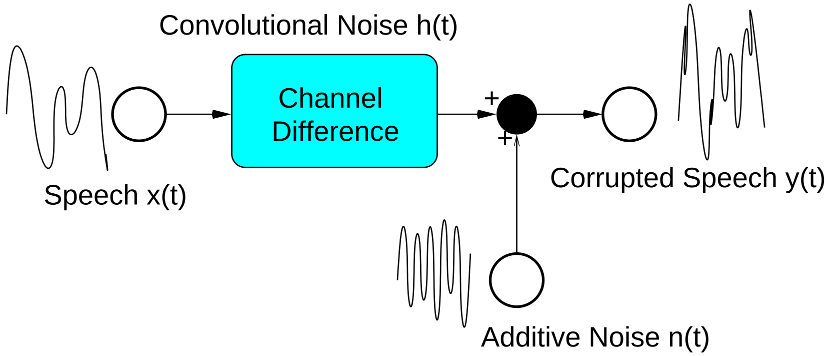
  - makes "exact" estimate of "clean" speech $\boldsymbol{x}_t$
  - multiple regression classes used to increase power of transform
- The recognition system (back-end) parameters not altered
  - important to efficiently handle changing noise environments
- Statistics required to use EM estimating transform

$$\mathbf{G}_i^{(r)} = \sum_{m \in \mathtt{r}_r} \sum_{t=1}^{T} \frac{\gamma_{\mathtt{y}t}^{(m)}}{\sigma_{\mathtt{x}i}^{(m)2}} \begin{bmatrix} 1 & \boldsymbol{y}_t^{\mathsf{T}} \\ \boldsymbol{y}_t & \boldsymbol{y}_t\boldsymbol{y}_t^{\mathsf{T}} \end{bmatrix}; \quad \mathbf{k}_i^{(r)} = \sum_{m \in \mathtt{r}_r} \sum_{t=1}^{T} \frac{\gamma_{\mathtt{y}t}^{(m)} \mu_{\mathtt{x}i}^{(m)}}{\sigma_{\mathtt{x}i}^{(m)2}} \begin{bmatrix} 1 \\ \boldsymbol{y}_t \end{bmatrix}$$

  - iterative scheme for full/block transforms, closed-form for diagonal

"Simplified" Acoustic Environment

- A simplified model of the effects of noise is often used



- Ignore effects of stress:
- Group noise sources

$$y(t) = x(t) * h(t) + n(t)$$

- Squared magnitude of the Fourier Transform of signal

$$Y(f)Y^*(f) = |H(f)X(f)|^2 + |N(f)|^2 + 2|N(f)||H(f)X(f)|\cos(\theta)$$

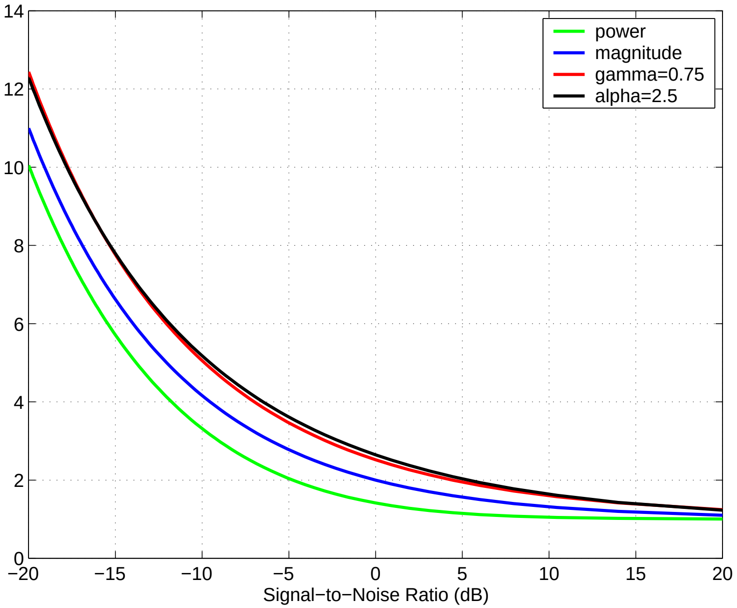
θ is the angle between the vectors $N(f)$ and $H(f)X(f)$.

- Considering Cepstral domain (**C** is the DCT), the mismatch function is obtained

$$\boldsymbol{y}_t = \boldsymbol{x}_t + \boldsymbol{h} + \frac{1}{2}\mathbf{C}\log\left(\mathbf{1} + \exp\left(2\mathbf{C}^{-1}(\boldsymbol{n}_t - \boldsymbol{x}_t - \boldsymbol{h})\right)\right)$$

Mismatch function optimisation

- The previous mismatch function only an approximation - variations possible
  - γ-optimised - tunable parameter γ, ignoring $\boldsymbol{h}$
$$\boldsymbol{y}_t = \boldsymbol{x}_t + \frac{1}{\gamma}\mathbf{C}\log\left(1+\exp\left(\gamma\mathbf{C}^{-1}(\boldsymbol{n}_t - \boldsymbol{x}_t)\right)\right)$$
  - Phase-sensitive - tunable parameter α, in theory $-1 \leq \alpha \leq 1$
$$\boldsymbol{y}_t = \boldsymbol{x}_t + \frac{1}{2}\mathbf{C}\log\left(1+\exp\left(2\mathbf{C}^{-1}(\boldsymbol{n}_t - \boldsymbol{x}_t)\right) + 2\alpha\exp\left(\mathbf{C}^{-1}(\boldsymbol{n}_t - \boldsymbol{x}_t)\right)\right)$$



Ratio of corrupted speech magnitude to clean speech magnitude

- magnitude ($\alpha = 1$, $\gamma = 1$)
- power ($\alpha = 0$, $\gamma = 2$)
- $\alpha = 2.5$ (MSR AURORA tuned)
- $\gamma = 0.75$ (AURORA tuned)

Vector Taylor Series Model Compensation

- Vector Taylor series (VTS) one popular approximation
  - Taylor series expansion about "current" parameter values
  - for these expression ignore impact of convolutional distortion
  - mismatch function approximated using first order series

$$\boldsymbol{y}_t \approx \boldsymbol{\mu}_{\rm x} + f(\boldsymbol{\mu}_{\rm x}, \boldsymbol{\mu}_{\rm n}) + \nabla_{\rm x} f(\boldsymbol{x}, \boldsymbol{n})|_{\mu_{\rm x}, \mu_{\rm n}}(\boldsymbol{x}_t - \boldsymbol{\mu}_{\rm x}) + \nabla_{\rm n} f(\boldsymbol{x}, \boldsymbol{n})|_{\mu_{\rm x}, \mu_{\rm n}}(\boldsymbol{n}_t - \boldsymbol{\mu}_{\rm n})$$

where $f(\boldsymbol{x}, \boldsymbol{n}) = \frac{1}{2}\mathbf{C}\log\left(\mathbf{1} + \exp\left(2\mathbf{C}^{-1}(\boldsymbol{n} - \boldsymbol{x})\right)\right)$ (ignoring $\boldsymbol{h}$)

- Model parameter compensation then becomes

$$\boldsymbol{\mu}_{\rm y}^{(m)} = \mathcal{E}\{\boldsymbol{y}_t|\mathrm{s}_m\} \approx \boldsymbol{\mu}_{\rm x}^{(m)} + f(\boldsymbol{\mu}_{\rm x}^{(m)}, \boldsymbol{\mu}_{\rm n})$$

$$\boldsymbol{\Sigma}_{\rm y}^{(m)} \approx \mathbf{A}\boldsymbol{\Sigma}_{\rm x}^{(m)}\mathbf{A}^{\mathsf{T}} + (\mathbf{I} - \mathbf{A})\boldsymbol{\Sigma}_{\rm n}^{(m)}(\mathbf{I} - \mathbf{A})^{\mathsf{T}}; \quad \mathbf{A} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$$

- "Current" noise parameter values have to be estimated

Noise Parameter Estimation

- In practice the noise model parameters, $\boldsymbol{\mu}_{\rm n}, \boldsymbol{\mu}_{\rm h}, \boldsymbol{\Sigma}_{\rm n}$, are not known
  - need to be estimated from test data
  - simplest approach - use VAD and start/end frames to estimate noise

- noise estimation based on ML estimation using observations $\mathbf{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$

$$\left\{\hat{\boldsymbol{\mu}}_{\rm n}, \hat{\boldsymbol{\mu}}_{\rm h}, \hat{\boldsymbol{\Sigma}}_{\rm n}\right\} = \operatorname*{argmax}_{\boldsymbol{\mu}_{\rm n}, \boldsymbol{\mu}_{\rm h}, \boldsymbol{\Sigma}_{\rm n}} \left\{p(\mathbf{Y}|\boldsymbol{\mu}_{\rm n}, \boldsymbol{\mu}_{\rm h}, \boldsymbol{\Sigma}_{\rm n}; \boldsymbol{\lambda}_{\rm x})\right\}$$

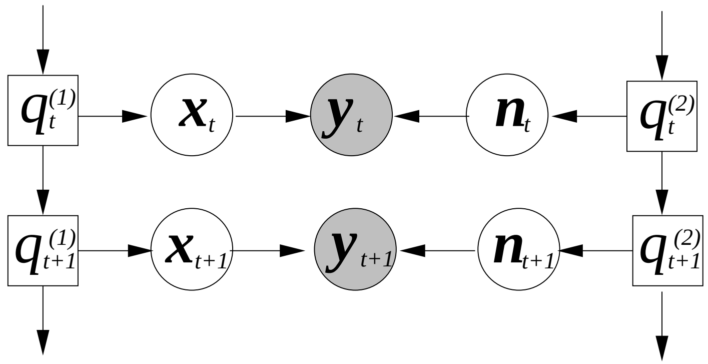
- Expectation maximisation can be used
  - simple statistics for the auxiliary function
  - yields simple approach to find $\boldsymbol{\mu}_{\rm n}, \boldsymbol{\mu}_{\rm h}$
  - first/second-order gradient based approaches to find $\boldsymbol{\Sigma}_{\rm n}$

- For details see technical report number TR552

Limitations of VTS

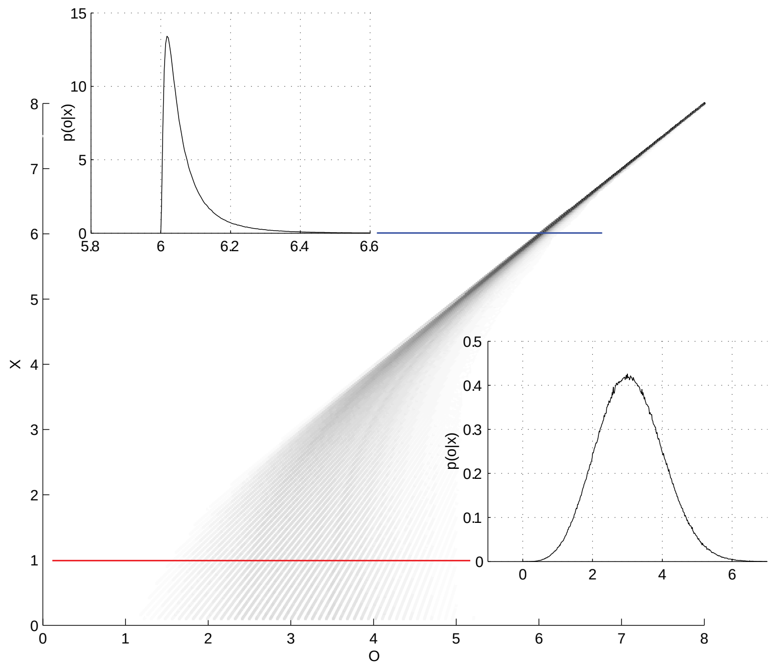
- VTS applied to a range of tasks, but
  1. computationally expensive, scales as number of components in recogniser
  2. mismatch function requires a number of approximations
  3. normally first order approximation used (not as good as DPMC)
  4. mismatch functions for dynamic parameters poor (continuous-time-approx)
  5. doesn't model changes in correlations of feature vector
- This work addresses (1) using Joint Uncertainty Decoding (JUD)
  - (2) will be examined by optimising γ
  - (3)-(5) have been examined see recent InterSpeech/ICASSP papers
- Would like to decorrelate compensation cost from complexity of recogniser
  - JUD partly does this using uncertainty decoding framework
  - though bias on recognition system variances still required

Uncertainty Decoding



$$p(\boldsymbol{y}_t) = \int p(\boldsymbol{y}_t|\boldsymbol{x}_t, \boldsymbol{n}_t)p(\boldsymbol{x}_t)p(\boldsymbol{n}_t)d\boldsymbol{x}_t d\boldsymbol{n}_t$$

$$p(\boldsymbol{y}_t) = \int p(\boldsymbol{y}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t)d\boldsymbol{x}_t$$



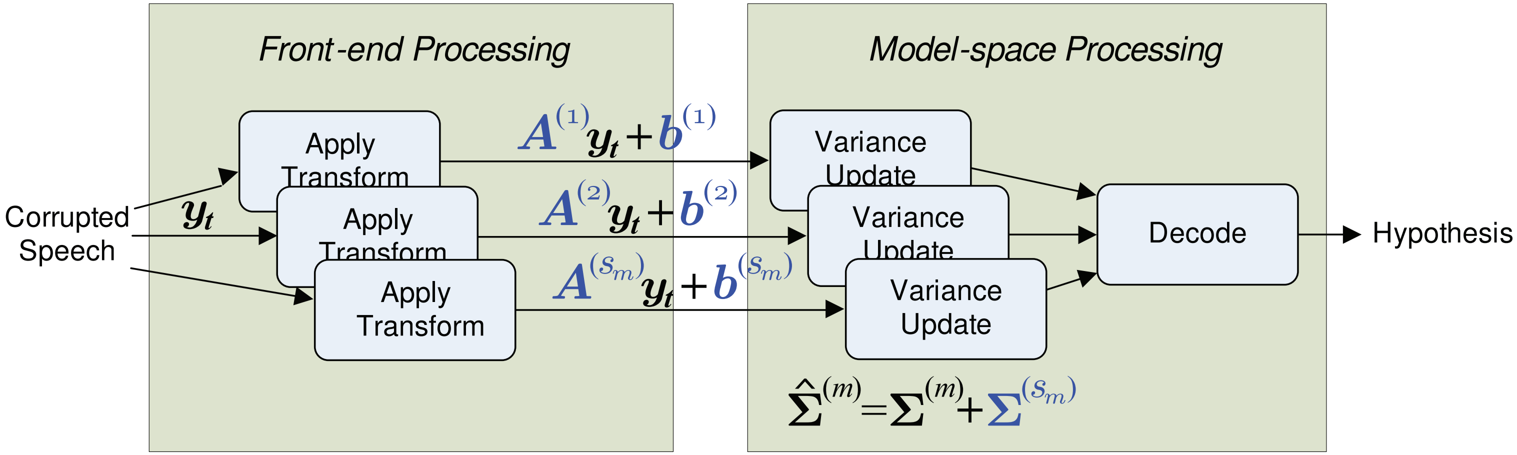
- Simplest approach is to assume $\boldsymbol{y}_t$ and $\boldsymbol{x}_t$ jointly Gaussian
  - $p(\boldsymbol{y}_t|\boldsymbol{x}_t)$ from joint distribution $p(\boldsymbol{y}_t, \boldsymbol{x}_t)$ obtained by means of VTS
- Considering a unique joint distribution for the whole acoustic space
  - $p(\boldsymbol{y}_t) = |\mathbf{A}|\ \mathcal{N}(\mathbf{A}\boldsymbol{y}_t + \mathbf{b}; \boldsymbol{\mu}_{\mathrm{x}}, \boldsymbol{\Sigma}_{\mathrm{x}} + \boldsymbol{\Sigma}_{\mathrm{b}})$ for a single model component

Joint Uncertainty Decoding Implementation

- JUD implementation closely related to CMLLR

$$p(\boldsymbol{y}_t|\mathbf{s}_m) = |\mathbf{A}^{(r)}|\mathcal{N}(\mathbf{A}^{(r)}\boldsymbol{y}_t + \mathbf{b}^{(r)}; \boldsymbol{\mu}_{\mathrm{x}}^{(m)}, \boldsymbol{\Sigma}_{\mathrm{x}}^{(m)} + \boldsymbol{\Sigma}_{\mathrm{b}}^{(r)})$$

  – bias variance on recognition components, but important for robustness



- For JUD compensation, VTS only required at regression class level
  – $\mathbf{A}^{(r)}, \mathbf{b}^{(r)}$ and $\boldsymbol{\Sigma}_{\mathrm{b}}^{(r)}$ functions of noise parameters $\boldsymbol{\mu}_{\mathrm{n}}, \boldsymbol{\mu}_{\mathrm{h}}, \boldsymbol{\Sigma}_{\mathrm{n}}$

"Predictive" Linear Model Compensation

- Though JUD improves efficiency there are still some problems
  1. if $\mathbf{A}^{(r)}$ is full, $\boldsymbol{\Sigma}_{\mathsf{b}}^{(r)}$ is full $\rightarrow$ full decoding cost
  2. $\boldsymbol{\Sigma}_{\mathsf{b}}^{(r)}$ applied to every back-end component
- Predictive linear transforms can be used to address both of these
- If $\boldsymbol{\Sigma}_{\mathsf{b}}^{(r)}$ is full can use predictive semi-tied transforms

$$p(\boldsymbol{y}_t|\mathbf{s}_m) = |\mathbf{A}^{(r)}||\mathbf{A}_{\mathsf{jnt}}^{(r)}|\mathcal{N}\left(\mathbf{A}^{(r)}(\mathbf{A}_{\mathsf{jnt}}^{(r)}\boldsymbol{y}_t + \mathbf{b}_{\mathsf{jnt}}^{(r)}); \mathbf{A}^{(r)}\boldsymbol{\mu}_{\mathsf{x}}^{(m)}, \tilde{\boldsymbol{\Sigma}}_{\mathsf{diag}}^{(m)}\right)$$

  - use semi-tied approximation: $(\boldsymbol{\Sigma}_{\mathsf{x}}^{(m)} + \boldsymbol{\Sigma}_{\mathsf{b}}^{(r)})^{\text{-1}} \approx \mathbf{A}^{(r)\mathsf{T}}\tilde{\boldsymbol{\Sigma}}_{\mathsf{diag}}^{(m)\text{-1}}\mathbf{A}^{(r)}$
  - various forms of approximation possible (various levels of efficiency)
  - but requires altering back-end recogniser parameters
- Predictive CMLLR can be used - only CMLLR-like transforms required

Predictive CMLLR

- For schemes like CMLLR required EM statistics can be expressed as:

$$\mathbf{G}_i^{(r)} = \sum_{m \in \mathrm{r}_r} \frac{\gamma^{(m)}}{\sigma_{\mathrm{x}i}^{(m)2}} \begin{bmatrix} 1 & \mathcal{E}\{\boldsymbol{y}^{\mathsf{T}}|\mathrm{s}_m\} \\ \mathcal{E}\{\boldsymbol{y}|\mathrm{s}_m\} & \mathcal{E}\{\boldsymbol{y}\boldsymbol{y}^{\mathsf{T}}|\mathrm{s}_m\} \end{bmatrix}; \quad \mathbf{k}_i^{(r)} = \sum_{m \in \mathrm{r}_r} \frac{\gamma^{(m)}\mu_{\mathrm{x}i}^{(m)}}{\sigma_{\mathrm{x}i}^{(m)2}} \begin{bmatrix} 1 \\ \mathcal{E}\{\boldsymbol{y}|\mathrm{s}_m\} \end{bmatrix}$$

  - normally the expectations obtained from observed data (adaptive)
  - could also use model-compensation schemes to obtain values (predictive)
  - $\gamma^{(m)}$ either based on observations $\gamma_{\mathrm{y}t}^{(m)}$ or training data counts $\gamma_{\mathrm{x}}^{(m)}$

$$\mathcal{E}\{\boldsymbol{y}|\mathrm{s}_m\} = \frac{\sum_t \gamma_{\mathrm{y}t}^{(m)} \boldsymbol{y}_t}{\sum_t \gamma_{\mathrm{y}t}^{(m)}} \quad \textbf{or} \quad \mathcal{E}\{\boldsymbol{y}|\mathrm{s}_m\} = \boldsymbol{\mu}_{\mathrm{y}}^{(m)}$$

- Schemes such as JUD can be used to efficiently obtain "pseudo" statistics

$$\sum_{m \in \mathrm{r}_r} \frac{\gamma_{\mathrm{x}}^{(m)}\mu_{\mathrm{x}i}^{(m)}}{\sigma_{\mathrm{x}i}^{(m)2}} \mathcal{E}\{\boldsymbol{y}|\mathrm{s}_m\} = \mathbf{A}^{(r)\text{-}1}\left(\sum_{m \in \mathrm{r}_r} \frac{\gamma_{\mathrm{x}}^{(m)}\boldsymbol{\mu}_{\mathrm{x}}^{(m)}}{\sigma_{\mathrm{x}i}^{(m)2}}\right) - \mathbf{A}^{(r)\text{-}1}\mathbf{b}^{(r)}\left(\sum_{m \in \mathrm{r}_r} \frac{\gamma_{\mathrm{x}}^{(m)}}{\sigma_{\mathrm{x}i}^{(m)2}}\right)$$

"Adaptive" vs "Predictive" Schemes

- Adaptive and predictive schemes complementary to one another

Adaptive	Predictive
general approach	applicable to noise
linear assumption - use many linear transforms	mismatch function required - may be inaccurate
transform parameters estimated - large numbers of parameters	noise model estimated - small number of parameters

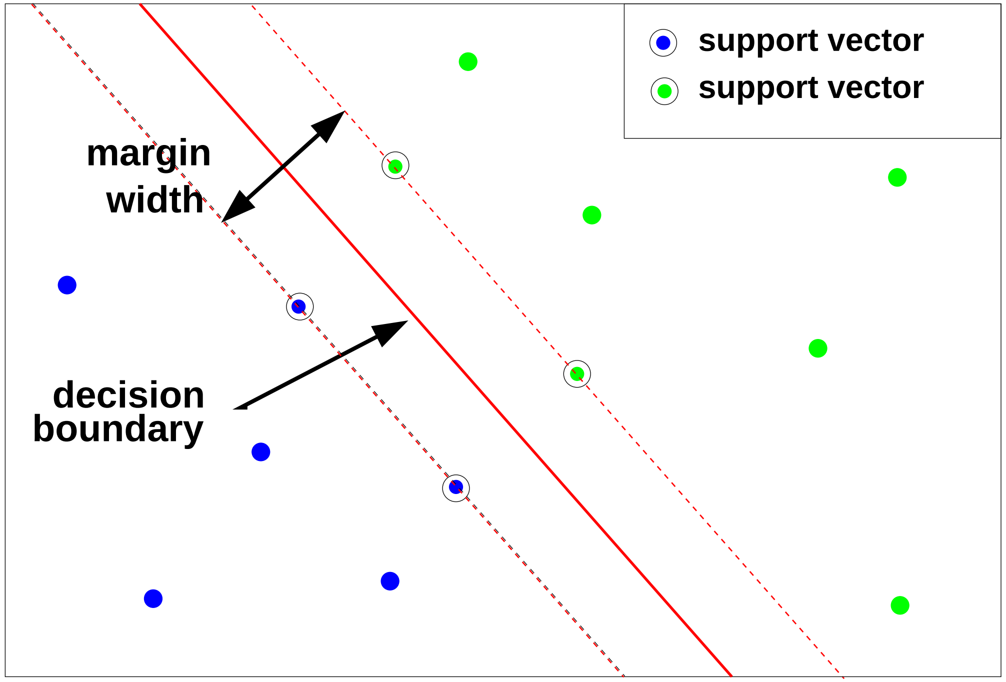
- Obvious approach is to combine the two in a fashion similar to MAP:
  - limited data predictive approaches used
  - increased data adaptive approaches used
- Count smoothing simple approach to use (parent transforms also possible)

$$\mathbf{G}^{(r)}_{\mathrm{pa}i} = \frac{\mathbf{G}^{(r)}_{\mathrm{pc}i}}{\sum_{m\in\mathbf{r}_r}\gamma_{\mathrm{x}}^{(m)}} + \tau_{\mathsf{sm}}\mathbf{G}^{(r)}_i; \quad \mathbf{k}^{(r)}_{\mathrm{pa}i} = \frac{\mathbf{k}^{(r)}_{\mathrm{pc}i}}{\sum_{m\in\mathbf{r}_r}\gamma_{\mathrm{x}}^{(m)}} + \tau_{\mathsf{sm}}\mathbf{k}^{(r)}_i$$

Improved Robustness

- Even if the "perfect" compensation scheme was available
  - unlikely to achieve required level of performance at low SNRs
  - use of alternative, non-HMM, classifiers may be needed
- Discriminative models/functions alternative to generative models (HMMs)
  - Support Vector Machines (SVMs) successful discriminative function
- To apply SVMs to noise robust speech recognition must be able to
  - handle variable-length sequence data from speech
  - adapt/compensate model to changing noise conditions
  - handle possibly large number of classes in speech recognition
- For details see technical report number TR605

Support Vector Machines



- SVMs are a maximum margin, binary, classifier:
  - related to minimising generalisation error;
  - unique solution (compare to neural networks);
  - may be kernelised - training/classification a function of dot-product $(\mathbf{x}_i.\mathbf{x}_j)$.
- Can be applied to speech - use a kernel to map variable data to a fixed length.

Generative Score-Space

- Generative models, e.g. HMMs and GMMs, handle variable length data
  - view as "mapping" sequence to a single dimension (log-likelihood)

$$\phi\left(\mathbf{Y};\boldsymbol{\lambda}\right)=\frac{1}{T}\log\left(p(\mathbf{Y};\boldsymbol{\lambda})\right)$$

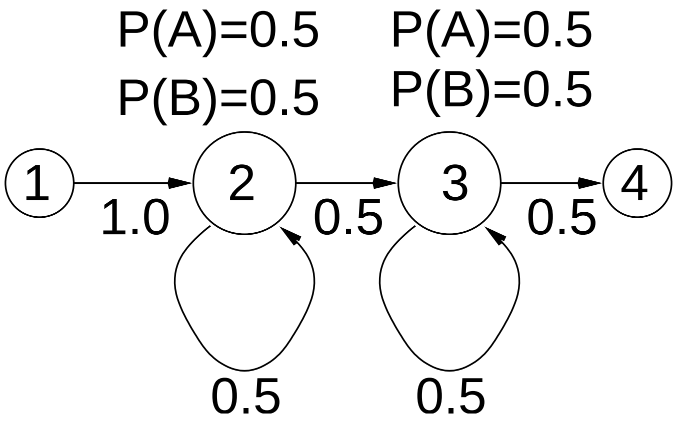
- Extend feature-space to a high dimension:
  - add derivatives with respect to the model parameters
  - example is a log-likelihood ratio plus first derivative score-space:

$$\boldsymbol{\phi}(\mathbf{Y};\boldsymbol{\lambda})=\frac{1}{T}\left[\begin{array}{c}\log\left(p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_l)})\right)-\log\left(p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_j)})\right)\\ \boldsymbol{\nabla}_{\lambda^{(\omega_l)}}\log\left(p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_l)})\right)\\ -\boldsymbol{\nabla}_{\lambda^{(\omega_j)}}\log\left(p(\mathbf{Y};\boldsymbol{\lambda}^{(\omega_j)})\right)\end{array}\right]$$

  - score-space is still a fixed dimension

Discrete Score-Space Example

- Consider a simple 2-class, 2-symbol $\{\texttt{A}, \texttt{B}\}$ problem:
  - Class ω_1: `AAAA`, `BBBB`
  - Class ω_2: `AABB`, `BBAA`



Feature	Class ω_1		Class ω_2	
	`AAAA`	`BBBB`	`AABB`	`BBAA`
Log-Lik	-1.11	-1.11	-1.11	-1.11
∇_{2A}	0.50	-0.50	0.33	-0.33
$\nabla_{2A}\nabla_{2A}^{\mathsf{T}}$	-3.83	0.17	-3.28	-0.61
$\nabla_{2A}\nabla_{3A}^{\mathsf{T}}$	-0.17	-0.17	-0.06	-0.06

- ML-trained HMMs are the same for both classes
- First derivative classes separable, but not linearly separable
  - also true of second derivative within a state
- Second derivative across state linearly separable

Generative Kernels

- Form of kernel need a metric to be specified

$$K(\mathbf{Y}_i, \mathbf{Y}_j; \boldsymbol{\lambda}) = \boldsymbol{\phi}(\mathbf{Y}_i; \boldsymbol{\lambda})^{\mathsf{T}} \mathbf{G}^{-1} \boldsymbol{\phi}(\mathbf{Y}_j; \boldsymbol{\lambda})$$

  - $\mathbf{G}$ is a maximally non-committal metric - all dimensions treated equally
  - Related to the Fisher kernel

- Log-likelihood ratio (LLR) known to be highly discriminatory

  - it's the score used for the HMM-based recognition

- In these experiments the LLR ratio weighted by ϵ so decision based on

$$\mathcal{S}(\mathbf{Y}; \boldsymbol{\lambda}) + \epsilon \left(\log \left(\frac{p(\mathbf{Y}; \boldsymbol{\lambda}_{\mathrm{y}}^{(\omega_l)})}{p(\mathbf{Y}; \boldsymbol{\lambda}_{\mathrm{y}}^{(\omega_j)})} \right) \right)$$

where $\mathcal{S}(\mathbf{Y}; \boldsymbol{\lambda})$ is the score from the SVM for classes ω_l and ω_j

Adapting SVMs to Noise Conditions

- Decision boundary for SVM is ($z_i \in \{-1, 1\}$ label of training example)

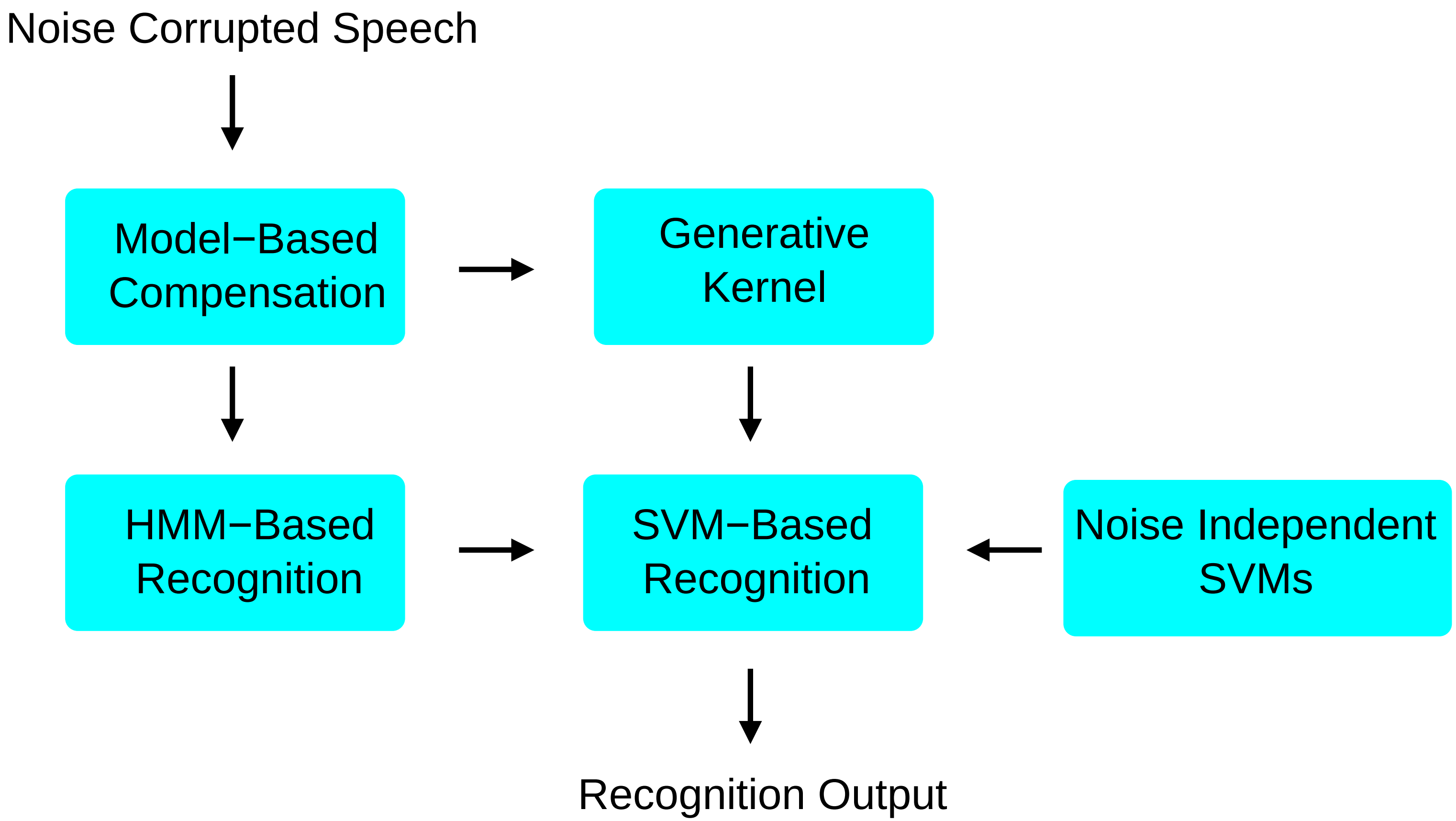
$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i^{\mathtt{svm}} z_i \mathbf{G}^{-1} \boldsymbol{\phi}(\mathbf{Y}_i; \boldsymbol{\lambda})$$

  – $\boldsymbol{\alpha}^{\mathtt{svm}} = \{\alpha_1^{\mathtt{svm}}, \ldots, \alpha_n^{\mathtt{svm}}\}$ set of SVM Lagrange multipliers

- Choice in adapting SVM to noise condition, modify:
  – $\boldsymbol{\alpha}^{\mathtt{svm}}$ - non-trivial though schemes have recently been proposed
  – $\boldsymbol{\lambda}$ - simple, model compensation

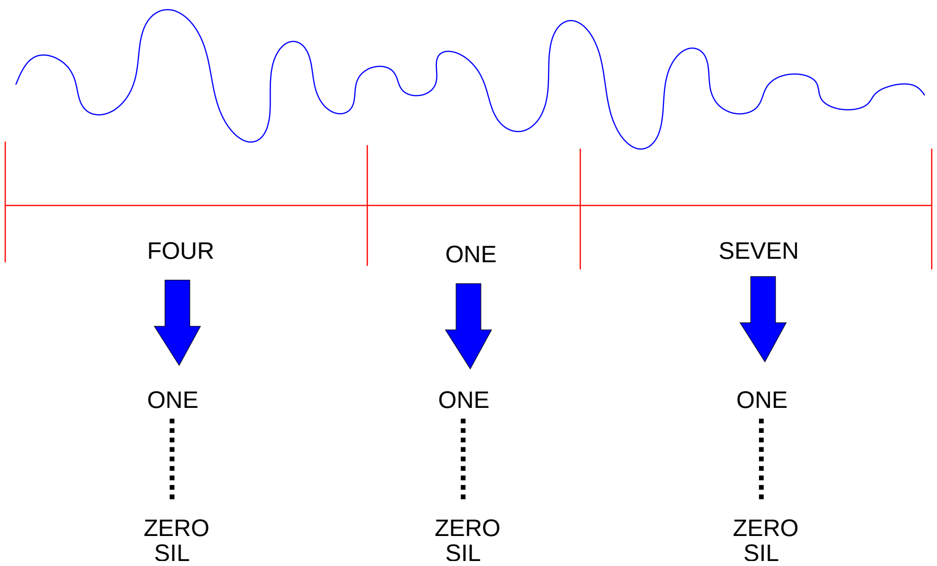
- Approach adopted in this work is to modify generative model parameters, $\boldsymbol{\lambda}$
  – noise-independent SVM Lagrange multipliers
  – noise-dependent generative kernels

SVMs Rescoring Scheme



- How to handle large number of possible classes even for simple digit strings?
  - 6 digit sequence has 10^6 possible classes

Handling Continuous Digit Strings



- Using HMM-based hypothesis
  - "force-align" - word start/end
- Foreach word start/end times
  - find "best" digit + silence
- Can use multi-class SVMs

- Simple approach to combining generative and discriminative models
  - related to acoustic code-breaking
- Initial implementation uses a 1-v-1 voting SVM combination scheme
  - ties between pairs resolved using appropriate SVM output
  - > 2 ties back-off to standard HMM output

AURORA 2 Task

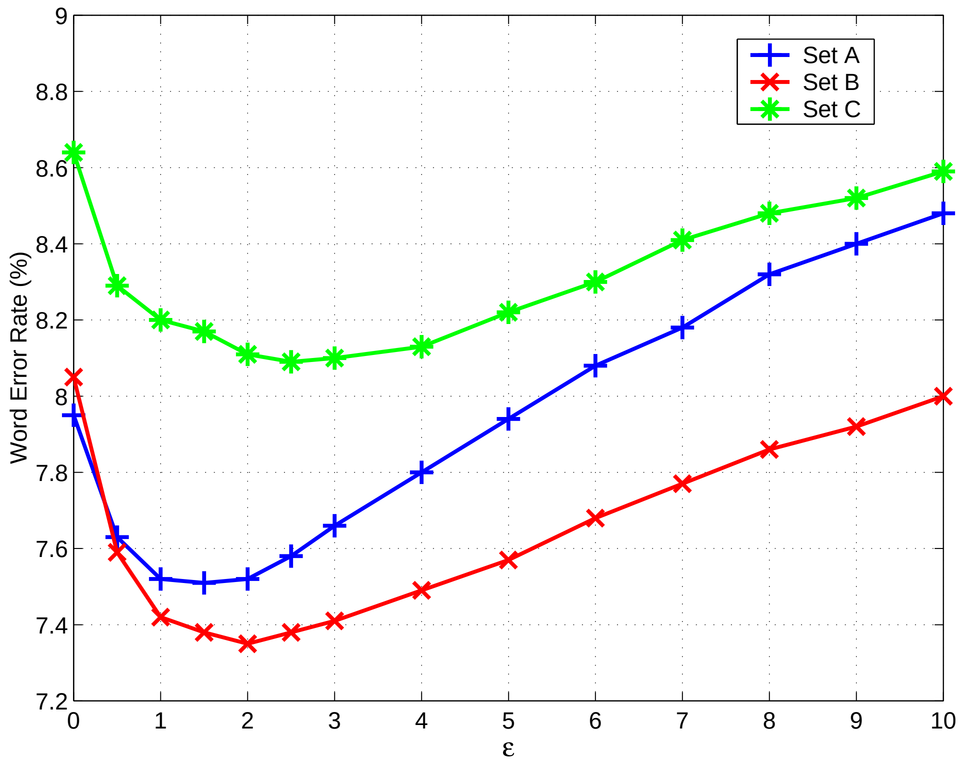
- AURORA 2 small vocabulary digit string recognition task
  - TIDIGITS databases used - utterances of one-seven digits (0-9 + `oh`)
  - clean training data 8440 utterances from 55 male and 55 female speakers
  - Test Set A and B contain 8 different types of additive noise
  - Test Set C contains 2 different types of additive noise plus channel distortion
  - 1001 utterances used for evaluation in each test set
  - Whole-word models, 16 emitting-states with 3 components per state.
- Two forms of MFCC parameterisation investigated HTK and ETSI based
- Noise estimated in a ML-fashion for each utterance using:
  1. initial noise model from first and last 20 frames
  2. hypothesis estimated using compensated models with initial noise model
  3. ML noise models estimated using hypothesis and 1 EM iteration
  4. recognition results based on ML noise model
- Steps (2) and (3) may be repeated for improved noise estimation

HTK-Based VTS and SVM Rescoring Performance

System	Test Set		
	A	B	C
VTS	9.84	9.11	9.53
+SVM	7.52	7.35	8.11

WER (%) averaged over 0-20dB

- VTS used ($\gamma = 1$)
- SVM rescoring used $\epsilon = 2$
- Large gains using SVM



- SVMs trained on subset of multi-style data - test set A N2-N4, 10-20dB SNR
  - Noise-independent SVM performs well on unseen noise conditions
- Graph shows variation of performance with ϵ - $\epsilon = 0$ better than VTS

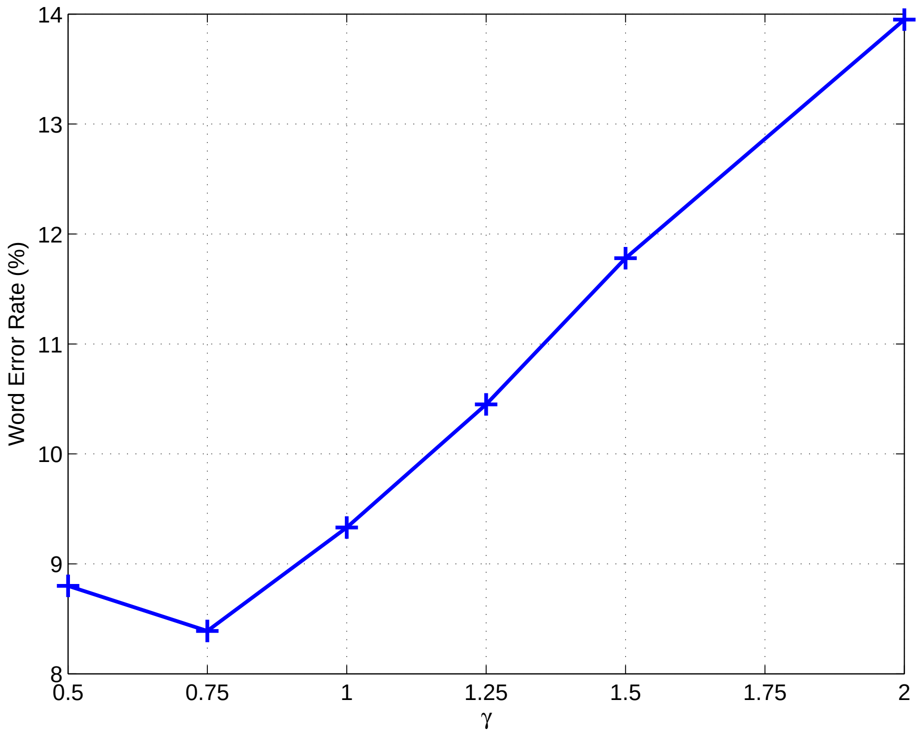
VTS results on Aurora data: HTK and ETSI Front-ends

SNR (dB)	Set A		Set B		Set C	
	HTK	ETSI	HTK	ETSI	HTK	ETSI
20	1.69	1.60	1.46	1.38	1.57	1.63
15	2.36	2.26	2.37	2.23	2.47	2.38
10	4.39	4.27	4.12	3.91	4.49	4.40
05	11.20	10.52	10.05	9.13	10.69	9.64
00	29.55	28.00	27.54	25.95	28.41	26.05
Avg	9.84	9.33	9.11	8.52	9.53	8.82

- ETSI Front-end (ES 201-108) performs better in all condition (but C-20dB)
- Main differences: cepstrum liftering, channels number
  - need to check ETSI performance on more realistic task - Toshiba data

VTS and SVM-Rescoring results using ETSI Front-end

- Possible to tune mismatch function for task (previously used $\gamma = 1$)



System	Test Set		
	A	B	C
VTS	9.33	8.52	8.82
$+\gamma$-opt	8.39	7.98	8.16
$+$SVM	6.99	6.80	7.35

WER (%) averaged over 0-20dB

- SVM rescoring yields further gains
  - $\epsilon = 2$ works well

- Graph shows average performance on test set A as γ varies
  - power-combination $\gamma = 2$ is poorer than magnitude-combination $\gamma = 1$
  - additional gains possible using $\gamma = 0.75$

Toshiba In-car Task

- TREL-CRL04 small/medium sized recognition task
  - Speech collected in the office and in vehicles (idle, city, highway)
  - 3 test sets: phone number, city name and command & control
- Vehicle conditions and phone number test set only considered
  - 30 English speakers (15 male, 15 female) uttering 30 sentences each
  - 35, 25, 18 SNR averages for the idle, city, highway condition, respectively
- Training data from WSJ SI284 to train clean acoustic models
  - 284 speakers for 66 hours of speech data
  - Acoustic models: decision tree clustered states, cross-word triphones,
HTK-based MFCC features
650 states, ≈7k components
  - $\gamma = 1$ used - no performance gain with $\gamma = 0.75$, better than $\gamma = 2$
- Multi-style models trained from clean models using SPR retraining
  - WSJ SI284 corrupted using in car recordings from SpeechDat and Toshiba

VTS/JUD Performance on the Toshiba Data

Comp	Iter	WER% ENON	WER% CITY	WER% HWY
VTS	0	3.35	8.87	13.11
	1	1.24	3.09	3.78
	2	1.37	2.65	3.15
JUD	0	3.37	8.94	14.11
	1	1.29	2.98	4.62
	2	1.42	2.53	4.07

Performance on phone-number task with VTS supervision iteration - JUD 64 base-classes

- 20+20 frames initial decoding significantly better than unadapted system
  - ML estimation of noise gives additional large gains over initial model
  - further gains from improved hypothesis possible
- JUD uses approximately 1/100th "VTS" compensations
  - slight performance degradation at lowest SNR (HWY)

SVM Rescoring on the Toshiba Data

System	VTS iter	WER (%)		
		ENON	CITY	HWY
Clean	—	3.85	31.81	66.18
VTS	1	1.24	3.09	3.78
+SVM		1.25	2.60	3.17
VTS	2	1.37	2.65	3.15
+SVM		1.31	2.14	2.48
MST	—	2.71	6.82	27.50

Performance on phone-number task with SVM rescoring

- SVM rescoring shows consistent gains over VTS compensation
  - larger gains for lower SNR conditions (CITY and HWY)
- Multi-style trained system performance using `MFCC_0_D_A` not impressive
  - significantly better using `MFCC_E_D_A_Z` - WER 7.55% for HWY

Incremental Adaptation Experiments

- Batch-mode adaptation introduces latency into decoding process
  - for some tasks, e.g. in-car command/control, need to minimise latency
  - many tasks require multiple interactions over a short period
- Incremental adaptation introduces no latency:
  1. using current transform recognition new utterance
  2. accumulate statistics from new utterance/hypothesis
  3. estimate new transform to recognise next utterance
- Minimal changes to code required
- Toshiba in-car task recorded in speaker-session
  - subset of speakers where all data from session used
  - task "artificial" - no-one cares about speaking multiple phone numbers!
  - but suitable to get an idea of what is going on

Incremental Adaptation on the Toshiba Data

System	Condition WER (%)		
	ENON	CITY	HWY
VTS	1.22	3.06	4.11
+CMLLR	0.51	2.67	3.38
JUD	1.10	2.86	5.42
+CMLLR	0.50	2.30	3.59
PCMLLR	1.21	2.93	6.14
+CMLLR	0.50	2.30	3.65

Combined incremental predictive and adaptive compensation - JUD/(P)CMLLR 64 base-classes

- For contrast: HWY VTS iteration 1 batch performance WER 3.82%
  - degradation for using incremental
  - "forgetting factor" can yield slight improvement
- Combining predictive and adaptive transforms yields performance gains

Conclusions

- VTS robust to different noise conditions and on different tasks
  - AURORA 2: various (including very low) SNR artificial data
  - Toshiba in-car data: real data
- JUD can get close to VTS performance
  - number regression classes approximately $1/100$ the number of components
- γ optimisation results not consistent between AURORA 2 and Toshiba data
  - but $\gamma = 1$ seems better than $\gamma = 2$, also observed on noise corrupted RM
- SVM rescoring experiments perfomed on AURORA 2 and Toshiba tasks
  - good preliminary results for small vocabulary experiments
  - can also be used for larger tasks - “acoustic code-breaking”
- Combining “adaptive” and “predictive” approaches can be useful