Noise Robust
Automatic Speech Recognition

Jasha Droppo
Microsoft Research

jdroppo@microsoft.com
http://research.microsoft.com/~jdroppo/

Additive Noise

- In controlled experiments, training and testing data can be made quite similar.
- In deployed systems, the test data is often corrupted in new and exciting ways.
Overview

• **Introduction**
  – Standard noise robustness tasks
  – Overview of techniques to be covered
  – General design guidelines

• Analysis of noisy speech features

• Feature-based Techniques
  – Normalization
  – Enhancement

• Model-based Techniques
  – Retraining
  – Adaptation

• Joint Techniques
  – Noise Adaptive Training
  – Joint front-end and back-end training
  – Uncertainty Decoding
  – Missing Feature Theory

Standard Noise-Robust Tasks

• Prove relative usefulness of different techniques

• Can include recipe for acoustic model building.

• Allow you to make sure your code is performing properly.

• Allow others to evaluate your new algorithms.
Standard Noise-Robust Tasks

• Aurora
  – Aurora2: Artificially Noisy English Digits
  – Aurora3: Digits recorded in four European languages, recorded inside cars.
  – Aurora4: Artificially noisy Wall Street Journal.
• SpeechDat Car
  – Noisy digits in European languages within cars.
• SPINE
  – Noises that can be mixed at various levels to clean speech signals to simulate noisy environments.

Feature-Based Techniques

• Only useful if you have the ability to retrain the acoustic model.
• Normalization: simple, yet powerful.
  – Moment matching (CMN, CVN, CHN)
  – Cepstral modulation filtering (RASTA, ARMA)
• Enhancement: more complex, but worth it.
  – Data-driven (POF, SVM, SPLICE)
  – Model-based (Spectral Subtraction, VTS)
Model-Based Techniques

• Retraining
  – When data is available, this is the best option.
• Adaptation
  – Approximate retraining with a low cost.
  – Re-purpose model-free techniques (MLLR, MAP)
  – Specialized techniques use a corruption model (VTS, PMC)

Joint Techniques

• Noise Adaptive Training
  – A hybrid of multi-style training and feature enhancement.
• Uncertainty Decoding
  – Allows the enhancement algorithm to make a “soft decision” when modifying the features.
• Missing Feature Theory
  – The feature extraction can define some spectral bins as unreliable, and exclude them from decoding.
General Design Guidelines

• Spend the effort for good audio capture
  – Close-talking microphones, microphone arrays, and intelligent microphone placement
  – Information lost during audio capture is not recoverable.

• Use training data similar to what is expected from the end-user
  – Matched condition training is best, followed by multistyle and mismatched training.

General Design Guidelines

• Use feature normalization
  – Low cost, high benefit
  – It eliminates signal variability not relevant to the transcription.
  – Not viable unless you control the acoustic model training.
Overview

- Introduction
  - Standard noise robustness tasks
  - Overview of techniques to be covered
  - General design guidelines
- Analysis of noisy speech features
- Feature-based Techniques
  - Normalization
  - Enhancement
- Model-based Techniques
  - Retraining
  - Adaptation
- Joint Techniques
  - Noise Adaptive Training
  - Joint front-end and back-end training
  - Uncertainty Decoding
  - Missing Feature Theory

How Does Noise Affect Speech Feature Distributions?

- Feature distributions trained in one noise condition will not match observations in another noise condition.
A Noisy Environment

- A clean, close-talk microphone signal exists (in theory) at the user’s mouth.
- Reverberation
  - A difficult problem, not addressed in this tutorial.
- Additive noise
  - At reasonable sound pressure levels, acoustic mixing is linear.
  - So, this should be easy, right?

Feature Extraction

- It is harder than you think.
- At the input to the speech recognition system, the corruption is linear.
- The “energy operator” and “logarithmic compression” are non-linear.
- The “mel-scale filterbank” and “discrete cosine transform” are one-way operations.
- The speech recognition features (MFCC) have non-linear corruption.
Analysis of Noisy Speech Features

- Additive noise systematically corrupts speech features.
  - Creates a mismatch between the training and testing data.
  - When the noise is highly variable, it can broaden the acoustic model.
  - When the noise masks dissimilar acoustics so their observations are similar, it can narrow the acoustic model.
- Additive noise affects static features (especially energy) more than dynamic features.
- Speech and noise mix non-linearly in the cepstral coefficients.

Distribution of noisy speech

- “Clean speech” distribution is Gaussian
  - Mean 25dB and sigma of 25, 10, and 5 dB.
- “Noise” distribution is also Gaussian
  - Mean 0db, sigma 2dB
- “Noisy speech” distribution is not Gaussian
  - Can be bi-modal, skewed, or unchanged.
- Smaller standard deviation yields better Gaussian.
- Additive noise decreases the variances in your acoustic model.
Distribution of noisy speech

- “Clean speech” distribution is Gaussian
  - Smaller covariance than previous example.
  - Sigma of 5dB and means of 10 and 5 dB.
- “Noise” distribution is also Gaussian
  - Same as previous example.
  - Sigma 2dB and mean 0dB
- “Noisy speech” distribution is skewed.

Overview

- Introduction
  - Standard noise robustness tasks
  - Overview of techniques to be covered
  - General design guidelines
- Analysis of noisy speech features
  - Feature-based Techniques
    - Normalization
    - Enhancement
  - Model-based Techniques
    - Retraining
    - Adaptation
- Joint Techniques
  - Noise Adaptive Training
  - Joint front-end and back-end training
  - Uncertainty Decoding
  - Missing Feature Theory
Feature-Based Techniques

• Voice Activity Detection
  – Eliminate signal segments that are clearly not speech.
  – Reduces “insertion errors”
  – May take advantage of features that the recognizer might not normally use.
    • Zero crossing rate.
    • Pitch energy.
    • Hysteresis.

Feature-Based Techniques

• Significant VAD Improvements on Aurora 3.

<table>
<thead>
<tr>
<th>Language</th>
<th>Without VAD</th>
<th>&quot;Oracle&quot; VAD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WM</td>
<td>MM</td>
</tr>
<tr>
<td>Danish</td>
<td>82.69</td>
<td>58.20</td>
</tr>
<tr>
<td>German</td>
<td>92.07</td>
<td>80.82</td>
</tr>
<tr>
<td>Spanish</td>
<td>88.06</td>
<td>78.12</td>
</tr>
<tr>
<td>Finnish</td>
<td>94.31</td>
<td>60.33</td>
</tr>
<tr>
<td>Ave</td>
<td>89.28</td>
<td>69.37</td>
</tr>
</tbody>
</table>
Feature-Based Techniques

- Cepstral Normalization
  - Mean Normalization (CMN)
  - Variance Normalization (CVN)
  - Histogram Normalization (CHN)
  - Cepstral Time Smoothing

Cepstral Mean Normalization

- Modify every signal so that its expected value is zero.
- Often coupled with automatic gain normalization (AGN).
- Eliminates variability due to linear channels with short impulse responses.

Further reading:
Cepstral Mean Normalization

- Compute the feature mean across the utterance
- Subtract this mean from each feature
- New features are invariant to linear filtering

\[
\bar{x} = \frac{1}{T} \sum_{t=0}^{T-1} x_t
\]

\[
\hat{x}_t = x_t - \bar{x}_t
\]

\[
y_t = x_t + h
\]

\[
\bar{y} = \frac{1}{T} \sum_{t=1}^{T-1} y_t
\]

\[
= \frac{1}{T} \sum_{t=0}^{T-1} (x_t + h)
\]

\[
= \bar{x} + h
\]

\[
\hat{y}_t = y_t - \bar{y} = \hat{x}_t
\]

Cepstral Variance Normalization

- Modify every signal so that the second central moment is one.
- No physical interpretation.
- Effectively normalizes the range of the input features.
Cepstral Variance Normalization

- Compute the feature mean and covariance across the utterance
- Subtract the mean and divide by the standard deviation
- New features are invariant to linear filtering and scaling

\[ \mu_X = \frac{1}{T} \sum_{t=0}^{T-1} x_t \]
\[ \sigma_X^2 = \frac{1}{T} \sum_{t=0}^{T-1} (x_t - \mu_X)^2 \]
\[ \tilde{x}_t = \frac{1}{\sigma_X} (x_t - \mu_X) \]

Further reading:

Cepstral Histogram Normalization

- Logical extension of CMVN.
  - Equivalent to normalizing each moment of the data to match a target distribution.
- Includes “Gaussianization” as a special case.
- Potentially destroys transcript-relevant information.
Effects of increasing the level of moment matching.

<table>
<thead>
<tr>
<th>Front End</th>
<th>Clean AM Base</th>
<th>Clean AM CMN</th>
<th>Clean AM CHN</th>
<th>Multi-Style AM Base</th>
<th>Multi-Style AM CMN</th>
<th>Multi-Style AM CHN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set A</td>
<td>63.66</td>
<td>67.02</td>
<td>79.89</td>
<td>87.74</td>
<td>89.44</td>
<td>91.08</td>
</tr>
<tr>
<td>Set B</td>
<td>57.83</td>
<td>69.22</td>
<td>81.26</td>
<td>87.63</td>
<td>89.34</td>
<td>90.75</td>
</tr>
<tr>
<td>Clean Test</td>
<td>99.01</td>
<td>99.07</td>
<td>98.94</td>
<td>98.67</td>
<td>98.97</td>
<td>98.13</td>
</tr>
</tbody>
</table>

- Notice first that Multi-Style is better on Set A and Set B, but worse on Clean Test.
- Increased moment matching always improves the accuracy, except on the clean data.
  - Short utterances (1.7s) don’t have enough data to do a stable CHN transformation.

Practical CHN

- CHN works well on long utterances, but can fail when there is not enough data.
- Polynomial Histogram Equalization (PHEQ)
  - Approximates inverse CDF function as a polynomial.
- Quantile-Based Histogram Normalization
  - Fast, on-line approximation to full HEQ.
- Cepstral Shape Normalization (CSN)
  - Starts with CMVN, then applies exponential factor.

Further reading:
Cepstral Time-Smoothing

- The time-evolution of cepstra carries information about both speech and noise.
- High-frequency modulations have more noise than speech.
  - So, low-pass filter the time series (~16Hz)
- Stationary components of the cepstral time-series do not carry relevant information.
  - So, high-pass filter the time series (~1Hz)
  - Similar to CMN?

Further reading:

Cepstral Time-Smoothing

- RASTA
  \[ H(z) = 0.1(z^4)^2 + z^{-1} - z^{-3} - 2z^{-4} \]
  - A fourth-order ARMA filter
  - Passband from 0.26Hz to 14.3Hz.
  - Empirically designed, shown to improve noise robustness considerably.

- MVA
  - Cascades CMVN and ARMA filtering

Further reading:
C.-P. Chen, K. Filali and J.A. Bilmes. Frontend post-processing and backend model enhancement on the Aurora 2.0/3.0 databases. In Int. Conf. on Spoken Language Processing, 2002.
Cepstral Time-Smoothing

• Temporal Shape Normalization
  – Logical extension of fixed modulation filtering.
  – Compute the “average” modulation spectrum from clean speech.
  – Transform each component of the noisy input using a linear filter to match the average modulation spectrum.
  – Better than CMVN alone, RASTA, or MVA.

Further reading:

Overview

• Introduction
  – Standard noise robustness tasks
  – Overview of techniques to be covered
  – General design guidelines
• Analysis of noisy speech features
• Feature-based Techniques
  – Normalization
  – Enhancement
• Model-based Techniques
  – Retraining
  – Adaptation
• Joint Techniques
  – Noise Adaptive Training
  – Joint front-end and back-end training
  – Uncertainty Decoding
  – Missing Feature Theory
Feature Enhancement

• Most useful technique when one doesn’t have access to retraining the recognizer’s acoustic model.
• Also provides some gain for the retraining case.
• Attempts to transform the existing observations into what the observations would have been in the absence of corruption.

Is Enhancement for ASR different?

• Design Constraints are Different than the general speech enhancement problem.
• Can tolerate extra delay.
  – Delayed decisions are generally better.
• ASR more sensitive to artifacts.
  – Less-aggressive parameter settings are needed.
• Can operate in log-Mel frequency domain
  – Fewer parameters, more well-behaved estimators.
Feature Enhancement

• Feature enhancement recipes contain three key ingredients:
  – A noise suppression rule
  – Noise parameter estimation
  – Speech parameter estimation

Noise Suppression Rules

• Estimate of the clean speech observation
  – That would have been measured in the absence of noise.
  – Given noise model and speech model parameters.
• Many Flavors
  – Spectral Subtraction
  – Weiner
  – logMMSE-STSA
  – CMMSE
Wiener Filtering

- Assume both speech and noise are independent, WSS Gaussian processes.
- The spectral time-frequency bins are complex Gaussian.
- MMSE estimate of the complex spectral values.

\[ \xi = \frac{\sigma_x^2}{\sigma_n^2} \]

\[ \hat{x} = \left( \frac{\xi}{\xi + 1} \right) y \]

Log-MMSE STSA

- MMSE of the log-magnitude spectrum.
- Gain is a function of speech parameters, noise parameters, and current observation.
- Similar domain to MFCC

\[ \xi = \frac{\sigma_x^2}{\sigma_n^2}, \gamma = \left| y \right|^2, \nu = \frac{\xi \gamma}{\xi + 1} \]

\[ \hat{x} = G(\xi, \gamma)y, \text{ where} \]

\[ G(\xi, \gamma) = \frac{\xi}{\xi + 1} \exp \left( \frac{1}{2} \int_{-\infty}^{\infty} \frac{e^{-z}}{z} dz \right) \]

Further Reading:
CMMSE

- Ephraim and Malah’s log-MMSE is formulated in the FFT-bin domain.
- Cepstral coefficients are different!
- New derivation by Yu is optimal in the cepstral domain.

Further Reading:
Noise Estimation

- To estimate clean speech, feature enhancement techniques need an estimate of the noise spectrum.
- Useful methods
  - Noise tracker
  - Speech detection
    - Track noise statistics in non-speech regions
    - Interpolate statistics into speech regions
  - Integrated with model-based techniques
  - Harmonic tunneling

The Importance of Noise Estimation

- Simple enhancement algorithms are sensitive to the noise estimation.
- Cheap improvements to noise estimation benefit any enhancement algorithm.
MCRA Noise Estimate

• Least-energetic samples are likely to be (biased) samples of the background noise.
• Components
  – Bias model
  – Per-bin speech activity detector
• Behavior
  – Quickly tracks noise when speech is absent
  – Smooths the estimate when speech is present

Further reading:

Noise Estimation: Speech Detection

• As frames are classified as non-speech, the noise model is updated.

\[
\hat{n}[t] = \begin{cases} 
\hat{n}[t - 1], & y[t] \in \text{speech} \\
\alpha \hat{n}[t - 1] + (1 - \alpha) y[t], & \text{otherwise}
\end{cases}
\]

Jasha Droppo / EUSIPCO 2008
Noise Estimation: Model Based

- Integrated with model-based feature enhancement
  - Uses “VTS Enhancement” theory later in this lecture.
  - Treat noise parameters as values that can be learned from the data.
- Dedicated noise tracking.
  - Uses a model for speech to closely track non-stationary additive noise.

Further reading:

Noise Estimation: Harmonic Tunneling
Noise Estimation: Harmonic Tunneling

• Attempts to solve the problems of
  – Tracking noises during voiced speech
  – Separating speech from noise
• Most speech energy is during voiced segments, in harmonics of the pitch period.
  – If the noise floor is above the valleys between the peaks, the noise spectrum can be estimated.

Further reading:

SPLICE

• The SPLICE transform defines a piecewise linear relationship between two vector spaces.
• The parameters of the transform are trained to learn the relationship between clean and noisy speech.
• The relationship is used to infer clean speech from noisy observations.

Further reading:
SPLICE Framework

SPLICE

- Learns a joint probability distribution for clean and noisy speech.
- Introduces a hidden discrete random variable to partition the acoustic space.
- Assumes the relationship between clean and noisy speech is linear within each partition.
- Standard inference techniques produce
  - MMSE or MAP estimates of the clean speech.
  - Posterior distributions on clean speech given the observation.

\[
\begin{align*}
  p(s) &= \text{constant} \\
  p(y|s) &= N(y; \mu_s, \sigma^2_s) \\
  p(x|y, s) &= N(x; y + r_s, \Gamma_s) \\
  p(x, y, s) &= p(x|y, s)p(y|s)p(s) \\
  \hat{x} &= E[x|y] = y + \sum_s p(s|y)r_s
\end{align*}
\]
SPLICE as Universal Transform

The complete SPLICE transform is a piecewise linear approximation to an arbitrary feature space mapping.

\[ x = \sum_{m} p(m|y)(A_m y + b_m) \]

Without the \( A_m \), it reduces to the usual SPLICE formula,

\[ x = y + \sum_{m} p(m|y)b_m \]

With one mixture component, it reduces to a linear (affine) transform.

\[ x = Ay + b \]

Other SPLICE-like Transforms

• Probabilistic Optimal Filtering (POF)
  – Earliest work on this type of transform for ASR.
  – Transform uses current and previous noisy estimates.

• Region-Dependent Transform

Further reading:
Other SPLICE-like Transforms

- Stochastic Vector Mapping (SVM)
- Multi-environment models based linear normalization (MEMLIN)
  - Generalization for multiple noise types
  - Models joint probability between clean and noisy Gaussians.

Further reading:

Training SPLICE-like Transformations

- Minimum mean-squared error (MMSE)
  - POF, SPLICE
  - Generally need stereo data.
- Maximum mutual information (MMI)
  - SVM, SPLICE
  - Objective function computed from clean acoustic model.
- Minimum phone error (MPE)
  - RDT, fMPE
  - Objective function computed from clean acoustic model.
The Old Way: A Fixed Front End

- Audio → Front End Feature Extraction → Back End Acoustic Model → Text

A Better Way: Trainable Front End

- Discriminatively optimize the feature extraction.
- Front end learns to feed better features to the back end.

Audio → Front End Feature Extraction (Trained) → Back End Acoustic Model (Fixed) → AM Scores → Text

MMI Objective Function
Example

- The digit “2” extracted from the utterance clean1/FAK_3Z82A
- MMI-SPLICE modifies the features to match the canonical “two” model stored in the decoder.
- Four regions are modified:
  - The “t” sound at the beginning is broadened in frequency and smoothed
  - The low-frequency energy is suppressed
  - The mid-range energy is suppressed
  - The tapering at the end of the top formant is smoothed

Since the Objective Function is Clearly Defined ...

- Accuracy, or a discriminative measure like MMI or MPE.
  \[ F_{MMI} = \frac{(p(X|w_c))^\gamma p(w_c)}{\sum_w (p(X|w))^\gamma p(w)} \approx p(w_c|X) \]
- Find derivative of objective function with respect to all parameters in the front end.
  - And I mean all the parameters.
    \[ \theta_k \leftarrow \theta_k + \alpha_k \frac{\partial F_{MMI}}{\partial \theta_k} \]
- Typical 10%-20% relative error rate reduction
  - Depends on the parameter chosen.
Training the Suppression Parameters

- Suppression Rule Modification
  - Parameters are 9x9 sample grid
  - 12% fewer errors on average (Aurora 2)
  - 60% fewer errors on clean test (Aurora 2)

Further Reading:
Overview

- Introduction
  - Standard noise robustness tasks
  - Overview of techniques to be covered
  - General design guidelines
- Analysis of noisy speech features
- Feature-based Techniques
  - Normalization
  - Enhancement
- Model-based Techniques
  - Retraining
  - Adaptation
- Joint Techniques
  - Noise Adaptive Training
  - Joint front-end and back-end training
  - Uncertainty Decoding
  - Missing Feature Theory

Model-based Techniques

- Goal is to approximate matched-condition training.
- Ideal scenario:
  - Sample the acoustic environment
  - Artificially corrupt a large database of clean speech
  - Retrain the acoustic model from scratch
  - Apply new acoustic model to the current utterance.
- Ideal scenario is infeasible, so we can choose to
  - Blindly adapt the model to the current utterance, or
  - Use a corruption model to approximate how the parameters would change, if retraining were pursued.
Retraining on Corrupted Speech

• Matched condition
  – All training data represents the current target condition.
• Multi-condition
  – Training data composed of a set of different conditions to approximate the types of expected target conditions.
• These are simple techniques that should be tried first.
• Requires data from the target environment.
  – Can be simulated

Model Adaptation

• Many standard adaptation algorithms can be applied to the noise robustness problem.
  – CMLLR, MLLR, MAP, etc.
• Consider a simple MLLR transform that is just a bias $h$.

$$\hat{h} = \arg\max_h p(X|\Lambda, h)$$

  – Solution is an E.-M. algorithm where all the means of the acoustic model are tied.
  – Compare to CMN, which blindly computes a bias.

Further reading:
Parallel Model Combination

- Approximates retraining your acoustic model for the target environment.
- Compose a clean speech model with a noise model, to create a noisy speech model.

Further reading:

Parallel Model Combination

“Data Driven”

- Procedure
  - Estimate a model for the additive noise.
  - Run Monte Carlo simulation to corrupt the parameters of your clean speech model.
  - Recognize using the corrupt model parameters.

- Analysis
  - Performance is limited only by the quality of the noise model.
  - More CPU-intensive than model-based adaptation.
Parallel Model Combination  
“Lognormal Approximation”

- Combine clean speech HMM with a model for the noise.
- Bring both models into the linear spectral domain
- Approximate adding of random variables
  - Assume the sum of two lognormal distributions is lognormal
- Convert back into cepstral model space.

Parallel Model Combination  
“Vector Taylor Series Approximation”

- Lognormal approximation is very rough.
- How can we do better?
  - Derive a more precise formula Gaussian adaptation.
  - Use a VTS approximation of this formula to adapt each Gaussian.
Vector Taylor Series Model Adaptation

• Similar in spirit to Lognormal PMC
  – Modify acoustic model parameters as if they had been retrained

• First, build a VTS approximation of $y$.

\[
\begin{align*}
y & = x + g(n - x) \\
g(z) & = \ln(1 + \exp(Dz)) \\
A & = \frac{\partial y}{\partial x} \left(\mu_x, \mu_n\right) \\
I - A & = \frac{\partial y}{\partial n} \left(\mu_x, \mu_n\right) \\
y & \approx \mu_x + g(\mu_n - \mu_x) + A(x - \mu_x) + (I - A)(n - \mu_n)
\end{align*}
\]

Further reading:
Data Driven, Lognormal PMC, and VTS

- Noise is Gaussian with mean 0dB, sigma 2dB
- “Speech” is Gaussian with sigma 10dB

Data Driven, Lognormal PMC, and VTS

- Noise is Gaussian with mean 0dB, sigma 2dB
- “Speech” is Gaussian with sigma 5dB
Vector Taylor Series Model Adaptation

- So, where does the function $g(z)$ come from?

- To answer that, we need to trace the signal through the front-end.
A Model of the Environment

• Recall that acoustic noise is additive.
  \[ y[t] = x[t] + n[t] \]

• For the (framed) spectrum, noise is still additive.
  \[ Y[k] = X[k] + N[k] \]

• When the energy operator is applied, noise is no longer additive.
  \[ |Y[k]|^2 = |X[k]|^2 + |N[k]|^2 + 2|X[k]N[k]|\cos\theta \]

A Model of the Environment

• The mel-frequency filterbank combines
  – Dimensionality reduction
  – Frequency warping

• After logarithmic compression, the noisy \( y[i] \) is what we analyze.
  \[ \exp(y[i]) = \sum_k w_k |Y[k]|^2 \]
Combining Speech and Noise

• Imagine two hypothetical observations
  – \( x[i] \) = The observation that the clean speech would have produced in the absence of noise.
  – \( n[i] \) = The observation that the noise would have produced in the absence of clean speech.

• We have the noisy observation \( y[i] \).

• How do these three variables relate?

A model of the Environment:
Grand Unified Equation

\[
\exp y_i = \exp x_i + \exp n_i + 2\alpha_i \exp \left( \frac{x_i + n_i}{2} \right)
\]

The conventional spectral subtraction model

Stochastic error caused by unknown phase of hidden signals.

Error term is also a function of hidden speech and noise features.

Further reading:
The “phase term” is usually ignored

\[\exp(y_i) = \exp(x_i) + \exp(n_i) + 2\alpha_i \exp \left( \frac{n_i + x_i}{2} \right)\]

\[y_i \approx x_i + \ln(1 + \exp(n_i - x_i))\]

- Most cited reason: because expected value is zero.
  - But, every frame can be significantly different from zero.
  - We’ll revisit this in a few slides.
- Appropriate when either \(x\) or \(n\) dominate the current observation.
- Otherwise, it represents (at best) a gross approximation to the truth.

Mapping to the cepstral space

- But, we’ve ignored the rest of the front end!
- The cepstral rotation
  - A linear (matrix) operation
  - From log mel-frequency filterbank coefficients (LMFB)
  - To mel-frequency cepstral coefficients (MFCC).

\[y_j^{\text{MFCC}} = \sum_i c_{ji} y_i^{\text{LMFB}}\]

- If the right-inverse matrix \(D\) is defined such that \(CD=I\), then the cepstral equation is

\[y = x + C \ln(1 + \exp(D(n-x)))\]

\[= x + g(x-n)\]
Vector Taylor Series: Correctly Incorporating Phase

- Unconditionally setting the phase to zero is a gross approximation.
- How much does it hurt the result?
- Where does it hurt most?
- Can we do better?

What is the Phase Term’s Distribution?

\[ \alpha_i = \frac{\sum_k w_k^i |X[k]| |N[k]| \cos \theta_k}{\sqrt{\sum_k w_k^i |X[k]|^2} \sqrt{\sum_k w_k^i |N[k]|^2}} \approx \sum_k \frac{w_k^i}{\sum_j w_j^i} \cos \theta_k. \]

- Distribution depends on filterbank.
- Approximately Gaussian for high frequency filterbanks.
Theoretical Observation Likelihood

- Observation Likelihood $p(y \mid x, n)$ as a function of $(x-y)$ and $(n-y)$.
- Phase term broadens the distribution near 0dB SNR.
  - $n<y$
  - $x<y$
  - $x>y$ and $n>y$

Check: The Model Matches Real Data
Observation Likelihood

• The model places a hard constraint on four random variables, leaving three degrees of freedom:

\[ p(y|x, n, \alpha) = \delta \left( y - \ln \left( e^x + e^n + 2\alpha e^{x+n} \right) \right). \]

• Third term is dependent on \( x \) and \( n \).
  
  – For \( x \gg n \) and \( x \ll n \), error term is relatively small.

SNR Dependent Variance Model

• Including a Gaussian prior for alpha, and marginalizing, yields:

\[ \ln p(y|x, n) = y - \frac{x + n}{2} - \frac{1}{2} \ln 8\pi \sigma^2_{\alpha} - \frac{(e^y - e^x - e^n)^2}{8\sigma^2_{\alpha} e^{x+n}}. \]

• But, this non-Gaussian posterior is difficult to evaluate properly.

Further reading:
SNR Independent Variance Model

• The SIVM assumes the error term is small, constant, and independent of $x$ and $n$.

  \[ y = \ln(\exp x + \exp n) + \epsilon \]
  \[ \epsilon \sim N(\epsilon; 0, \psi) \]

• Gaussian posterior is easy to derive and to evaluate.

  \[ p(y|x, n) = N(y; \ln(\exp x + \exp n), \psi) \]

Modeling and Complexity Tradeoff

• Models all regions equally well.
• Costly to implement properly.

• Models all regions poorly.
• More economical implementation.
Zero Variance Model

- A special, simpler case of both the SDVM and SIVM.
  - Correct model for high and low instantaneous SNR.
  - Approximate for 0dB SNR.
- Assume the phase term is always exactly zero.
- Introduce a new instantaneous SNR variable $r$.
  \[
  r = x - n \\
  y = \ln(\exp x + \exp n)
  \]
- Replace inference on $x$ and $n$ with inference on $r$.
  \[
  x = y - \ln(e^r + 1) + r \\
  n = y - \ln(e^r + 1).
  \]

Iterative VTS

Choose initial expansion point

Develop approximate posterior (VTS)

Estimate posterior mean

\[
\hat{r} = \int rp(r|y, s, r_0)dr
\]

Use mean as new expansion point

Done

\[
r_0 = \hat{r}
\]

\[
r_0 = E[x - n|s] \\
= \mu_s^0 - \mu^0
\]
Iterative VTS: Behavior

- Iterative VTS is a Gaussian approximation that converges to a local maximum of the posterior.
- The SDVM is a non-Gaussian distribution whose mean and maximum are not co-incident.
- As a result, Iterative VTS fails spectacularly when used with the SDVM.
  - Good inference schemes under the CDVM have been developed [ICSLP 2002] but come at a high computational cost.

Now that we know where \( g(z) \) comes from...

What else is it good for?
Vector Taylor Series Enhancement

• Full VTS model adaptation
  – Can be quite expensive.

• VTS Enhancement
  – Uses the power of VTS to enhance the speech features.
  – Computes MMSE estimate of clean speech given noisy speech, and a noise model.
  – Recognizes with a standard acoustic model.

Further reading:
VTS Enhancement

• The true minimum mean squared estimate for the clean speech should be,

\[ \hat{x} = \sum_s E[x|y, s]p(s|y) \]

• A good approximation for this expectation, given the parameters available from the ZVM, is

\[ E[x|y, s] \approx y - \ln(e^{\hat{\mu}_s} + 1) + \hat{\mu}_s \]

• Since the expectation is only approximate, the estimate is sub-optimal.

Using More Advanced Models with VTS Enhancement

• Hidden Markov models.
  – Replace the (time-independent) GMM mixture component with a Markov chain.
  – Observation probability still dominates.
  – More complex models (e.g., phone-loop or full decoding) propagate their errors forward.
Using More Advanced Models with VTS Enhancement

• Switching Linear Dynamic Models.
  – Inference is difficult (exponentially hard)

\[
p(x_t, s_t|x_{t-1}) = N(x_t; A_s x_{t-1} + b_s, C_s) p(s_t)
\]

\[
p(x_1^T, s_1^T) = p(x_1, s_1) \prod_{t=2}^T p(x_t, s_t|x_{t-1})
\]

Further reading:


Overview

• Introduction
  – Standard noise robustness tasks
  – Overview of techniques to be covered
  – General design guidelines
• Analysis of noisy speech features
• Feature-based Techniques
  – Normalization
  – Enhancement
• Model-based Techniques
  – Retraining
  – Adaptation
• Joint Techniques
  – Noise Adaptive Training
  – Joint front-end and back-end training
  – Uncertainty Decoding
  – Missing Feature Theory

Joint Techniques

• Joint methods address the inefficiency of partitioning the system into feature extraction and pattern recognition.
  – Feature extraction must make a hard decision
  – Hand-tuned feature extraction may not be optimal
• Methods to be discussed
  – Noise Adaptive Training
  – Joint Training of Front and Back Ends
  – Uncertainty Decoding
  – Missing Feature Theory
Noise Adaptive Training

• A combination of multistyle training and enhancement.
• Apply speech enhancement to multistyle training data.
  – Models are tighter, they don’t need to describe all the variability introduced by different noises.
  – Models learn the distortions introduced by the enhancement process.
• Helps generalization
  – Under unseen conditions, the residual distortion can be similar, even if the noise conditions are not.

Further reading:
L. Deng, A. Acero, M. Plumpe and X.D. Huang. Large-vocabulary speech recognition under adverse acoustic environments. In Int. Conf on Spoken Language Processing, Beijing, China, 2000.

Joint Training of Front and Back Ends

• A general discriminative training method for both the front end feature extractor and back end acoustic model of an automatic speech recognition system.
• The front end and back end parameters are jointly trained using the Rprop algorithm against a maximum mutual information (MMI) objective function.

Further reading:
The New Way: Joint Training

• Front end and back end updated simultaneously.
• Can cooperate to find a good feature space.

Joint training is better than either SPLICE or AM alone.
Joint training is better than serial training.

Uncertainty Decoding and Missing Feature Techniques

- Not all observations generated by the front end should be treated equally.
- Uncertainty Decoding
  - Grounded in probability and estimation theory
  - Front-end gives cues to the back-end indicating the reliability of feature estimation
- Missing Feature Theory
  - Grounded in auditory scene analysis
  - Estimates which observations are buried in noise (missing)
  - Mask is created to partition the features into reliable and missing (hidden) data.
  - The missing data is either marginalized in the decoder (similar to uncertainty decoding), or imputed in the front end.
Uncertainty Decoding

• When the front-end enhances the speech features, it may not always be confident.

• Confidence is affected by
  – How much noise is removed
  – Quality of the remaining cues

• Decoder uses this confidence to modify its likelihood calculations.

\[
p(y|m) = \int_{-\infty}^{\infty} p(y|x, m)p(x|m) \, dx
= \alpha \int_{-\infty}^{\infty} N(\hat{x}; x, \sigma^2_{\hat{x}}) N(x; \mu_m, \sigma^2_m) \, dx
= \alpha N(\hat{x}; \mu_m, \sigma^2_m + \sigma^2_{\hat{x}})
\]
Uncertainty Decoding

• The trick is calculating reasonable parameters for $p(y|x)$.

• SPLICE
  – Model the residual variance in addition to bias.
    \[
    r_m = E[x - y|m]
    \]
    \[
    \Gamma_m = E[(x - y)^2|m]
    \]
  – Compute $p(y|x)$ using Bayes’ rule and other approximations.

Further reading:
**Missing Feature: Spectrographic Masks**

- A binary mask that partitions the spectrogram into reliable and unreliable regions.
- The reliable measurements are a good estimate of the clean speech.
- The unreliable measurements are an estimate of an upper bound for clean speech.

\[
\begin{align*}
Y(n,k) &= \log(\exp(X(n,k)) + \exp(N(n,k))) \\
X_r(n,k) &= Y_r(n,k) \\
X_u(n,k) &\leq Y_u(n,k)
\end{align*}
\]

**Further reading:**

---

**Missing Feature: Mask Estimation**

- SNR Threshold and Negative Energy Criterion
  - Easy to compute.
  - Requires estimate of the noise spectrum.
  - Unreliable with non-stationary additive noises.
- Bayesian Estimation
  - Measure SNR and other features for each spectral component.
  - Build a binary classifier for each frequency bin based on these measurements.
  - More complex system, but more reliable with non-stationary additive noises.

**Further reading:**
Missing Feature:
Imputation

• Replace missing values from \( x \) with estimates based on the observed components.

• Decode on reliable and imputed observations.

\[
\hat{X}_m(m, k) = E[X_m(n, k)|X_r(n, k) = Y_r(n, k)]
\]

Further reading:

Missing Feature:
Imputation

• Cluster-based reconstruction
  – Assume time slices of the spectrogram are IID.
  – Build a GMM to describe the PDF of these time slices.
  – Use the spectrographic mask, GMM, and bounds on the missing data to estimate the missing data.

• “Bounded Maximum a priori Approximation”

\[
\hat{X}_m = \arg \max_{X_m} \{ P(X_m|X_r, X_m \leq Y_m) \} \\
\approx \sum_{\nu} P(\nu|X_r, X_m \leq Y_m) \text{BMAP}(\nu|X_r, X_m \leq Y_m; \mu(\nu, \Theta(\nu)))
\]
Missing Feature:
Imputation

• Covariance-based reconstruction
  – Assume spectrogram is a sequence of correlated vector-valued Gaussian random process.
    \[
    \mu(k) = E\{X(n, k)\} \\
    c(\zeta, k_1, k_2) = E\{(X(n, k_1) - \mu_{k_1})(X(n + \zeta, k_2) - \mu_{k_2})\}
    \]

• Compute bounded MAP estimate of missing data.
  – Ideally, simultaneous joint estimation of all missing data.
  – Practically, estimate one frame at a time from neighboring reliable components

Missing Feature:
Classifier Modification

• Marginalization
  – When computing \( p(x|m) \) in the decoder, integrate over possible values of the missing \( x \).
  – Similar to Uncertainty Decoding, when the uncertainty becomes very large.
    \[
    p(x|m) = p(x_r|m)p(x_u|m) \\
    = p(x^r|m) \int_{-\infty}^{x_u} dx_u p(x_u|m)
    \]
Missing Feature: Classifier Modification

• Fragment Decoding
  – Segment the data into two parts
    • “dominated by target speaker” (the reliable fragments)
    • “everything else” (the masker)
  – Decode over the fragments.
  – Not naturally computationally efficient
  – Quite promising for very noisy conditions, especially “competing speaker”.

Further reading:

Summary

• Evaluate on standard tasks
  – Good sanity check for your code
  – Allows others to evaluate your algorithm
• Spend the effort for good audio capture
• Use training data similar to what is expected at runtime
• Implement simple algorithms first, then move to more complex solutions
  – Always include feature normalization
  – When possible, add model adaptation
  – To achieve maximum performance, or if you can’t retrain the acoustic model, implement feature enhancement.