Noise Level Normalization and Reference Adaptation for Robust Speech Recognition

Florian Hilger and Hermann Ney

Lehrstuhl für Informatik VI
RWTH Aachen – University of Technology

September 18, 2000
Previous experiments:

- **Spectral subtraction (SS)** can be combined successfully with **SNR-normalization (SNRN)** [Claes, Compernolle 96].

- **Own refinements:**
  - Turn SS and SNRN off during speech parts of the signal.
  - Switch to a set of unnormalized references when noise is low.

- **Disadvantages:**
  - Many parameters to optimize empirically.
  - Memory for two reference sets is needed.

⇒ Alternative: Noise Level Normalization
Noise Level Normalization

Idea:

- Normalize the noise level on the filter bank outputs before taking the logarithm.
- Carry out the normalization in one step, with less parameters.
- Use only one reference set, that was trained on clean data.

Approach:

- Multiply a filter function $H_k$ to the filter bank output $Y_k$

$$Y_k^{norm}[t] = H_k[t] \cdot Y_k[t]$$

that has the following properties:

- An energy threshold determines, if the current time frame is silence or speech.
- If silence is assumed the filter bank output is scaled down to the fraction of the average speech level that was observed in training.
- Speech frames are left unchanged.
Algorithm:

- In training determine the relation $\gamma$ between the average filter bank output levels in noise $\bar{Y}^n$ and speech $\bar{Y}^s$ parts of the signal:

$$\gamma = \frac{1}{T_{tr}} \sum_t \frac{\bar{Y}^n[t]}{\bar{Y}^s[t]}$$

- In recognition calculate a noise normalizing factor $\nu_k$ for each filter bank output $k$:

$$\nu_k[t] = \min\left( \gamma \cdot \frac{\bar{Y}^s[t]}{\bar{Y}^n_k[t]} , \ 1 \right)$$

- Use sigmoid function of the average output magnitude in the current time frame $\bar{Y}[t]$ as soft noise/speech threshold:

$$s(\bar{Y}[t]) = \frac{1}{1 + \exp\left( -\alpha \left( \frac{\bar{Y}[t]}{(1 - \beta)\bar{Y}^n + \beta\bar{Y}^s} - 1 \right) \right)}$$

- Normalize:

$$Y_k^{\text{norm}}[t] = \left( \nu_k[t] + (1 - \nu_k[t])s[t] \right) \cdot Y_k[t] / H_k[t]$$

Definitions: $\bar{Y}^n[t] = \frac{1}{TK} \sum_t \sum_k Y_k[t_n]$ ; $\bar{Y}^s[t] = \frac{1}{TK} \sum_t \sum_k Y_k[t_s]$ ; $\bar{Y}^n_k[t] = \frac{1}{T} \sum_t Y_k[t_n]$ ; $\bar{Y}^s[t] = \frac{1}{K} \sum_k Y_k[t]$
Normalizing function:

Mapping from $Y_k$ to $Y_{k}^{norm}$ at different noise levels, if $Y_k \sim \bar{Y}$ is assumed:

$$Y_{k}^{norm}[t] = (\nu_k[t] + (1 - \nu_k[t])s[t]) \cdot Y_k[t]$$

N1 : Clean (average noise level 1 arbitrary unit) no normalization
N2 : Noise level 2 units
N3 : Noise level 4 units
Reference Adaptation

Idea:

• The algorithm is based on the observation that the average norm \( l \) of the MFCC feature vectors \( X[t] \):

\[
l = \frac{1}{T} \sum_t \sqrt{||X[t]||^2}
\]

decreases in the presence of additional noise when using cepstral mean normalization.

Algorithm:

• Calculate the average feature vector norm \( l_{tr} \) on the training data.

• Determine \( l_{rec} \) for the utterance that is to be recognized.

• Calculate the normalizing factor

\[
n(l_{rec}) = \min \left( \frac{l_{rec}}{l_{tr}}, 1 \right)
\]

• Multiply the references mean vectors with this factor when calculating the log likelihood:

\[
- \ln p( x \mid n(l_{rec}) \cdot \mu, \sigma )
\]
TI digit string database:

- TI digit string database sampled down to 8kHz
- Clean training data
- Car noise added to the testing data at different SNRs

Recognizer setup:

- MFCC feature extraction
  15 filters;
  12 cepstral coeff. + 12 first deriv. + 1 second deriv.
  = 25 – dimensional feature vector

- Word models for 11 English digits, including “oh”

- Gender dependent modeling

- Laplacian single densities
  - State dependent diagonal covariance matrix
  - No. of densities: $2 \times 358$
Setups:

- **Baseline:** with cepstral mean normalization
- **SS+SNRN:** improved spec. subtraction and SNR–normal.
- **NLN:** noise level normalization
- **RA:** reference adaptation

<table>
<thead>
<tr>
<th>SNR [dB]</th>
<th>baseline</th>
<th>SS+SNRN</th>
<th>NLN</th>
<th>RA</th>
<th>NLN+RA</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>0.6</td>
<td>0.7</td>
<td>0.6</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>20</td>
<td>3.8</td>
<td>2.4</td>
<td>2.3</td>
<td>2.7</td>
<td>1.9</td>
</tr>
<tr>
<td>15</td>
<td>7.3</td>
<td>3.6</td>
<td>3.9</td>
<td>4.3</td>
<td>2.6</td>
</tr>
<tr>
<td>10</td>
<td>15.7</td>
<td>7.8</td>
<td>8.0</td>
<td>8.3</td>
<td>4.1</td>
</tr>
<tr>
<td>5</td>
<td>33.4</td>
<td>20.4</td>
<td>18.8</td>
<td>18.8</td>
<td>9.5</td>
</tr>
<tr>
<td>0</td>
<td>70.1</td>
<td>55.4</td>
<td>40.1</td>
<td>46.2</td>
<td>25.9</td>
</tr>
</tbody>
</table>
German Isolated Word Car Navigation Database:

- **Training data:**
  - Recorded in an office environment
  - Sampled at 16kHz
  - Amount: 13h 37min (60% silence)
  - 86 speakers
  - 12900 different spoken words

- **Testing data:**
  - 3 testing sets: office, city traffic and highway traffic
  - Amount: about 75min each
  - 14 speakers
  - Recognizer vocabulary: 2100 words + 581 pron. variants
    (none of these words occurs in the training data)
Recognizer setup:

- **MFCC feature extraction**
  - 20 filters;
  - 16 cepstral coeff. + 16 first deriv. + 1 second deriv.
  - $= 33$ – dimensional feature vector

- **Linear discriminant analysis:**
  - Reduction from $3 \times 33$ components to 33

- **Classification and regression tree:**
  - 700 tied triphone states

- **Gender independent models**

- **Gaussian mixtures**
  - Pooled diagonal covariance matrix
  - No. of densities: 21k

- **Single word recognition**
Setups:

- Baseline: with cepstral mean normalization
- NLN: noise level normalization
- RA: reference adaptation

<table>
<thead>
<tr>
<th>SNR [dB]</th>
<th>German Car Navigation Database</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WER [％]</td>
</tr>
<tr>
<td></td>
<td>baseline</td>
</tr>
<tr>
<td>office 21</td>
<td>3.8</td>
</tr>
<tr>
<td>city 9</td>
<td>35.5</td>
</tr>
<tr>
<td>highway 6</td>
<td>78.0</td>
</tr>
</tbody>
</table>
Conclusions:

- Noise level normalization outperforms optimized spectral subtraction and SNR-normalization.
- Reference adaptation performs almost as well.
- The effect of NLN and RA can be combined leading to significantly lower error rates.
- The approaches are computationally inexpensive:
  - Suited for real time applications
  - Low memory requirements