33. Environmental Robustness

J. Droppo, A. Acero

When a speech recognition system is deployed outside the laboratory setting, it needs to handle a variety of signal variabilities. These may be due to many factors, including additive noise, acoustic echo, and speaker accent. If the speech recognition accuracy does not degrade very much under these conditions, the system is called *robust*. Even though there are several reasons why real-world speech may differ from clean speech, in this chapter we focus on the influence of the *acoustical environment*, defined as the transformations that affect the speech signal from the time it leaves the mouth until it is in digital format.

Specifically, we discuss strategies for dealing with additive noise. Some of the techniques, like feature normalization, are general enough to provide robustness against several forms of signal degradation. Others, such as feature enhancement, provide superior noise robustness at the expense of being less general. A good system will implement several techniques to provide a strong defense against acoustical variabilities.

33.1 Noise Robust Speech Recognition

This chapter addresses the problem of additive noise at the input to an automatic speech recognition (ASR) system. Parts H and I in this Handbook address how to build microphone arrays for superior sound capture, or how to reduce noise for perceptual audio quality. Both of these subjects are orthogonal to the current discussion.

Microphone arrays are useful in that improved audio capture should be the first line of defense against additive noise. However, despite the best efforts of the system designer, there will always be residual additive noise. In general, speech recognition systems prefer linear array algorithms, such as beam forming, to nonlinear techniques. Although nonlinear techniques can achieve better suppression and perceptual quality, the introduced distortions tend to confuse speech recognition systems.

Furthermore, speech enhancement algorithms designed for improved human perception do not always help ASR accuracy. Most enhancement algorithms introduce some signal distortion, and the type of distortion that can be tolerated by humans and computers can be quite different.

Additive noise is common in daily life, and can be roughly categorized as either stationary or nonstationary. Stationary noise, such as that made by a computer fan or air conditioning, has a frequency spectrum that does not change over time. In contrast, the spectrum of a nonstationary noise changes over time. Some examples of nonstationary noise are a closing door, music, and other speakers' voices. In practice, no noise is perfectly stationary. Even the noises from a computer fan, an air-conditioning system, or a car will change over a long enough time period.

33.1.1 Standard Noise-Robust ASR Tasks

When building and testing noise-robust automatic speech recognition systems, there are a rich set of standards to test against.

The most popular tasks today were generated by the European Telecommunications Standards Institute's technical committee for Speech, Transmission Planning, and Quality of Service (ETSI STQ). Their AURORA digital speech recognition (DSR) working group was formed to develop and standardize algorithms for distributed and noise-robust speech recognition. As a byproduct of their work, they released a series of standard tasks [33.1] for system evaluation. Each task consists of all the necessary components for running an experiment, including data and recipes for building acoustic and language models, and scripts for running evaluations against different testing scenarios.

The Aurora 2 task is the easiest to set up and use, and is the focus of many results in this chapter. The data was derived from the TIDigits corpus [33.2], which consists of continuous English digit strings of varying lengths, spoken into a close-talking microphone. To simulate noisy telephony environments, these clean utterances were first downsampled to 8 kHz, and then additive and convolutional noise was added. The additive noise is controlled to produce noisy signals with a range of signal-to-noise ratios (SNRs) of −5–20 dB.

The noise types include both stationary and nonstationary noises, and are broken down into three sets: set A (subway, babble, car, and exhibition), set B (restaurant, street, airport, and station), and set C (subway and street). Set C contains one noise from set A, and one from set B, and also includes extra convolutional noise. There are two sets of training data, one is clean and the other is noisy. The noisy training data contains noises similar to set A. This represents the case where the system designers are able to anticipate correctly the types of noises that the system will see in practice. Test set B, on the other hand, has four different types of noises.

The acoustic model training recipe that was originally distributed with Aurora 2 was considered to be too weak. As a result, many researchers built better acoustic models to showcase their techniques. However, this made their results incomparable. To rectify this problem, a standard *complex back-end* recipe [33.3] was proposed, which is the proper model to use when performing new experiments on this task.

The Aurora 3 task is similar in complexity to the Aurora 2 task, but covers four other European languages in real car noise scenarios. Because the data is a subset of the SpeechDat car database [33.4], the noise types between Aurora 2 and Aurora 3 are quite different. The noise types chosen for Aurora 2 can be impulsive and nonstationary, but the car noise in Aurora 3 tends to be well modeled by stationary colored noise. Whereas the noisy utterances in Aurora 2 are artificially mixed, the noisy utterances of Aurora 3 were collected in actual noisy environments. Nevertheless, techniques exhibit a strong correlation in performance between Aurora 2 and Aurora 3, indicating that digitally simulated noisy speech is adequate for system evaluation.

The Aurora 4 task was developed to showcase noise-robust speech recognition for larger-vocabulary systems. Whereas the previous Aurora tasks have 10 or 11 word vocabularies, the Aurora 4 task has a 5000-word vocabulary. In much the same way that Aurora 2 was derived from the clean TIDigits corpus, the Aurora 4 task was derived from the clean Wall Street Journal (WSJ) corpus [33.5]. Noises are digitally mixed at several signal-to-noise ratios. Because of the difficulty in setting up the larger system, the Aurora 4 task is not as commonly cited in the literature. The Aurora 4 task is relevant because some techniques that work well on Aurora 2 either become intractable or fail on larger-vocabulary tasks.

Noisex-92 [33.6] is a set of data that is also useful in evaluating noise robust speech recognition systems. It consists of two CD-ROMS of audio recordings, suitable for use in artificially mixing noise with clean speech to produce noisy speech utterances. There is a great variety of noises available with the data, including voice babble, factory noise, F16 fighter jet noise, M109 tank noise, machine gun noise, and others.

Table 33.1 Word accuracy for the Aurora 2 test sets using the clean acoustic model baseline

SNR (dB)	Test set A	Test set B	Test set C	Average
Clean	99.63	99.63	99.60	99.62
20	95.02	91.71	97.02	94.58
15	85.16	78.10	92.38	85.21
10	64.50	55.75	77.78	66.01
5	34.59	29.21	51.36	38.39
0	13.61	9.75	22.82	15.40
−5	5.87	4.08	11.47	7.14
Average (0–20)	58.58	52.90	68.27	58.25

Another common evaluation task for noise robust speech recognition systems is the speech in noisy environments (SPINE) evaluation [33.7]. It was created for the Department of Defense digital voice processing consortium, to support the 2000 SPINE1 evaluation. The corpus contains 9 h 22 min of audio data, collected in simulated noisy environments where users collaborate using realistic handsets and communications channels to seek and shoot targets, similar to the game Battleship.

33.1.2 The Acoustic Mismatch Problem

To understand the extent of the problem of recognizing speech in noise, it is useful to look at a concrete example. Many of the techniques in this chapter are tested on the Aurora 2 task. Table 33.1 contains typical results from the baseline Aurora 2 system. Here, an acoustic model is trained on clean, noise-free data, and tested on data with various digitally simulated noise levels. The accuracy on clean test data averages 99.62%, which may be acceptable for some applications.

As soon as any noise is present in the test data, the system rapidly degrades. Even at a mild 20 dB signal-to-noise ratio (SNR), the system produces more than 14 times as many errors compared to clean data. (The signal-to-noise ratio is defined as the ratio of signal energy to noise energy in the received signal. It is typically measured in decibels (dB), and calculated as $10\log_{10}$[Energy(signal)/Energy(noise)]. An SNR above 30 dB sounds quite noise-free. At 0 dB SNR, the signal and noise are at the same level.) As the SNR decreases further, the problem becomes more intense. Why does an ASR system perform so poorly when presented with even mildly corrupted signals? The answer is deceptively simple. Automatic speech recognition is fundamentally a pattern matching problem. And, when a system is tested on patterns that are unlike anything used to train it, errors are likely to occur. The fundamental problem is the *acoustic mismatch* between the training and testing data.

Figure 33.1 illustrates the severity of the problem. It compares the histograms for C_1 between clean speech and moderately noisy speech. (C_1, the first cepstral coefficient, is typical speech feature used by automatic speech recognition systems.) The two histograms are quite dissimilar. Obviously, a system trained under one condition will fail under the other.

33.1.3 Reducing Acoustic Mismatch

The simplest solution to the acoustic mismatch problem is to build an acoustic model that is a better match for the test data. Techniques that can be helpful in that respect are multistyle training and model adaptation. These types of algorithms are covered in Sect. 33.2.

Another common approach to solving the acoustic mismatch problem is to transform the data so that the training and testing data tend to be more similar. These techniques concentrate on either normalizing

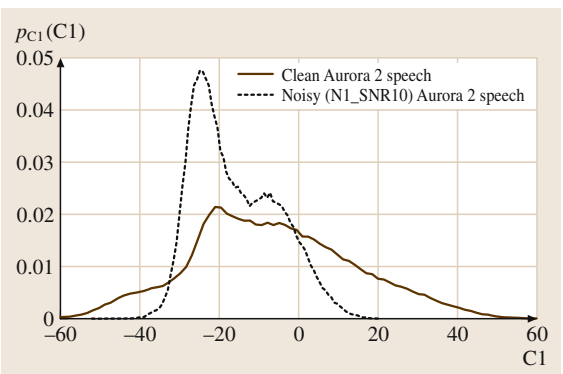

Fig. 33.1 Additive noise creates a mismatch between clean training data and noisy testing data. Here, the histogram for a clean speech feature is strikingly different from the same histogram computed from noisy speech

out the effects of noise, or learning transformations that map speech into a canonical noise-free representation. Examples of such techniques are cepstral mean normalization and stereo piecewise linear compensation for environment (SPLICE), which are covered in Sect. 33.3.

The techniques mentioned so far are powerful, but limited. They do not assume any form for the acoustic mismatch, so they can be applied to compensate for a wide range of corrupting influences. But, because they are unstructured, they need a lot of data to handle a new condition. Section 33.4 introduces a model of how noise corrupts clean speech features. Later, the model is used to derive powerful data-thrifty adaptation and normalization algorithms.

Section 33.5 presents the first class of these algorithms, which adapt the parameters of a clean acoustic model to approximate an acoustic model for noisy speech. Parallel model combination with a log-normal approximation is covered, as is vector Taylor-series (VTS) model adaptation.

The model for how additive noise corrupts clean speech features can also be used to do speech feature enhancement. Section 33.6 covers the classic technique of spectral subtraction as it can be integrated into speech recognition. It also covers vector Taylor-series speech enhancement, which has proven to be an easy and economical alternative to full model adaptation.

The last set of techniques discussed in this chapter are hybrid approaches that bridge the space between model and feature based techniques. In general, the former are more powerful, but the latter are easier to compute. Uncertainty decoding and noise adaptive training are two examples presented in Sect. 33.7, which are more powerful than a purely feature-based approach, but without the full cost of model adaptation.

33.2 Model Retraining and Adaptation

The best way to train any pattern recognition system is to train it with examples that are similar to those it will need to recognize later.

One of the worst design decisions to make is to train the acoustic model with data that is dissimilar to the expected testing input. This usually happens when the training data is collected in a quiet room using a close-talking microphone. This data will contain very good speech, with little reverberation or additive noise, but it will not look anything like what a deployed system will collect with its microphone. It's true that robustness algorithms can ameliorate this mismatch, but it is hard to cover up for a fundamental design flaw.

The methods presented in this section demonstrate that designing appropriate training data can greatly improve the accuracy of the final system.

33.2.1 Retraining on Corrupted Speech

To build an automatic speech recognition system that works in a particular noise condition, one of the best solutions is to find training data that matches this condition, train the acoustic model from this data in the normal way, and then decode the noisy speech without further processing. This is known as *matched condition* training.

If the test conditions are not known precisely, *multistyle training* [33.8] is a better choice. Instead of using a single noise condition in the training data, many different kinds of noises are used, each of which would be reasonable to expect in deployment.

Matched condition training can be simulated with sample noise waveforms from the new environments. These noise waveforms are artificially mixed with clean training data to create a synthetically noisy training data set. This method allows us to adapt the model to the new environment with a relatively small amount of data from the new environment, yet use a large amount of training data for the system.

The largest problem with multistyle training is that it makes components in the acoustic model broader and less discriminative. As a result, accuracy in any one condition is slightly worse than if matched condition training were available, but much better than if a mismatched training were used.

Tables 33.1 and 33.2 present the standard clean condition and multistyle training results from the Aurora 2 tasks. In the clean condition experiments, the acoustic model is built with uncorrupted data, even though the test data has additive noise. This is a good example of mismatched training conditions, and the resulting accuracy is quite low.

The multistyle training results in Table 33.2 are much better than the clean training results of Table 33.1. Even though the noises from set B are different from the training set, their accuracy has been improved considerably over the mismatched clean condition training. Also, notice that the word accuracy of the clean test data is lower

Table 33.2 Word accuracy for the Aurora 2 test sets using the multistyle acoustic model baseline

SNR (dB)	Test set A	Test set B	Test set C	Average
Clean	99.46	99.46	99.46	99.45
20	98.99	98.59	98.78	98.79
15	98.41	97.56	98.12	98.03
10	96.94	94.94	96.05	95.98
5	91.90	88.38	87.92	89.40
0	70.53	69.36	59.35	66.42
−5	30.26	33.05	25.16	29.49
Average (0–20)	91.36	89.77	88.04	90.06

for the multistyle-trained models. This is typical of multistyle training: whereas before, the clean test data was matched to the clean training data, now every type of test data has a slight mismatch.

33.2.2 Single-Utterance Retraining

Taken to the extreme, the retraining approach outlined above could be used to generate a new acoustic model for every noisy utterance encountered.

The first step would be to extract exemplar noise signals from the current noisy utterance. This is then used to artificially corrupt a clean training corpus. Finally, an utterance specific acoustic model is trained on this corrupted data. Such a model should be a good match to the current utterance, and we would expect excellent recognition performance.

Of course, this approach would only be feasible for small systems where the training data can be kept in memory and where the retraining time is small. It would certainly not be feasible for large-vocabulary speaker-independent systems.

The idea of using an utterance-specific acoustic model can be made more efficient by replacing the recognizer retraining step with a structured adaptation of the acoustic model. Each Gaussian component in the acoustic model is adapted to account for how its parameters would change in the presence of noise. This idea is the basis of the techniques such as parallel model combination (PMC) and VTS model adaptation in Sect. 33.5, where a model for noise is composed with the acoustic model for speech, to build a noisy speech model. Although they are less accurate than the brute-force method just described, they are computationally simpler.

33.2.3 Model Adaptation

In the same way that retraining the acoustic model for each utterance can provide good noise robustness, standard unsupervised adaptation techniques can be used to approximate this effect.

Speaker adaptation algorithms, such as maximum a priori (MAP) or maximum likelihood linear regression (MLLR), are good candidates for robustness adaptation. Since MAP is an unstructured method, it can offer results similar to those of matched conditions, but it requires a significant amount of adaptation data. MLLR can achieve reasonable performance with about a minute of speech for minor mismatches [33.9]. For severe mismatches, MLLR also requires a large number of transformations, which, in turn, require a larger amount of adaptation data.

In [33.10], it was shown how MLLR works on Aurora 2. That paper reports a 9.2% relative error rate reduction over the multistyle baseline, and a 25% relative error rate reduction over the clean condition baseline.

33.3 Feature Transformation and Normalization

This section demonstrates how simple *feature normalization* techniques can be used to reduce the acoustic mismatch problem. Feature normalization works by reducing the mismatch between the training and the testing data, leading to greater robustness and higher recognition accuracies.

It has been demonstrated that feature normalization alone can provide many of the benefits of noise robustness specific algorithms. In fact, some of the best results on the Aurora 2 task have been achieved with feature normalization alone [33.11]. Because these techniques are easy to implement and provide impressive results,

they should be included in every noise-robust speech recognition system.

This section covers the most common feature normalization techniques, including voice activity detection, automatic gain normalization, cepstral mean and variance normalization, cepstral histogram normalization, and cepstral filtering.

33.3.1 Feature Moment Normalization

The goal of feature normalization is to apply a transformation to the incoming observation features. This transformation should eliminate variabilities unrelated to the transcription, while reducing the mismatches between the training and the testing utterances. Even if you do not know how the ASR features have been corrupted, it is possible to normalize them to reduce the effects of the corruption.

With moment normalization, a one-to-one transformation is applied to the data, so that its statistical moments are normalized. Techniques using this approach include cepstral mean normalization, cepstral mean and variance normalization, and cepstral histogram normalization. Respectively, they try to normalize the first, the first two, and all the moments of the data. The more moments that are normalized, the more data is needed to prevent loss of relevant acoustic information.

Another type of normalization affects only the energy-like features of each frame. Automatic gain normalization (AGN) is used to ensure that the speech occurs at the same absolute signal level, regardless of the incoming level of background noise or SNR. The simplest of these AGN schemes subtracts the maximum $C0$ value from every frame for each utterance. With this method, the most energetic frame (which is likely to contain speech) gets a $C0$ value of zero, while every other frame gets a negative $C0$.

When using moment normalization, it is sometimes beneficial to use AGN on the energy-like features, and the more-general moment normalization on the rest. For each of the moment normalization techniques discussed below, the option of treating $C0$ separately with AGN is evaluated.

Cepstral Mean Normalization

Cepstral mean normalization is the simplest feature normalization technique to implement, and should be considered first. It provides many of the benefits available in the more-advanced normalization algorithms.

For our analysis, the received speech signal $x[m]$ is used to calculate a sequence of cepstral vectors $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}\}$. In its basic form, cepstral mean normalization (CMN) (*Atal* [33.12]) consists of subtracting the mean feature vector $\boldsymbol{\mu}_{\boldsymbol{x}}$ from each vector $\boldsymbol{x}_t$ to obtain the normalized vector $\hat{\boldsymbol{x}}_t$:

$$\boldsymbol{\mu}_{\boldsymbol{x}} = \frac{1}{T}\sum_t \boldsymbol{x}_t \,, \tag{33.1}$$

$$\hat{\boldsymbol{x}}_t = \boldsymbol{x}_t - \boldsymbol{\mu}_{\boldsymbol{x}} \,. \tag{33.2}$$

As a result, the long-term average of any observation sequence (the first moment) is zero.

It is easy to show that CMN makes the features robust to some linear filtering of the acoustic signal, which might be caused by microphones with different transfer functions, varying distance from user to microphone, the room acoustics, or transmission channels.

To see this, consider a signal $y[m]$, which is the output of passing $x[m]$ through a filter $h[m]$. If the filter $h[m]$ is much shorter than the analysis window used to compute the cepstra, the new cepstral sequence $\{\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{T-1}\}$ will be equal to

$$\boldsymbol{y}_t = \boldsymbol{x}_t + \boldsymbol{h} \,. \tag{33.3}$$

Here, the cepstrum of the filter $\boldsymbol{h}$ is defined as the discrete cosine transform (DCT) of the log power spectrum of the filter coefficients $h[m]$:

$$\boldsymbol{h} = \mathbf{C}\left(\ln|H(\omega_0)|^2 \ldots \ln|H(\omega_M)|^2\right) . \tag{33.4}$$

Since the DCT is a linear operation, it is represented here as multiplication by the matrix $\mathbf{C}$.

The sample mean of the filtered cepstra is also offset by $\boldsymbol{h}$, a factor which disappears after mean normalization:

$$\boldsymbol{\mu}_{\boldsymbol{y}} = \frac{1}{T}\sum_{t=1}^{T-1} \boldsymbol{y}_t = \boldsymbol{\mu}_{\boldsymbol{x}} + \boldsymbol{h} \,, \tag{33.5}$$

$$\hat{\boldsymbol{y}}_t = \boldsymbol{y}_t - \boldsymbol{\mu}_{\boldsymbol{y}} = \hat{\boldsymbol{x}}_t \,. \tag{33.6}$$

As long as these convolutional distortions have a time constant that is short with respect to the front end's analysis window length, and does not suppress large regions of the spectrum below the noise floor (e.g., a severe low-pass filter), CMN can virtually eliminate their effects. As the filter length $h[m]$ grows, (33.3) becomes less accurate and CMN is less effective in removing the convolutional distortion. In practice, CMN can normalize the effect of different telephone channels and microphones, but fails with reverberation times that start to approach the analysis window length [33.13].

Tables 33.3 and 33.4 show the effectiveness of whole-utterance CMN on the Aurora 2 task. In the

Table 33.3 Word accuracy for Aurora 2, using cepstral mean normalization and an acoustic model trained on clean data. CMN reduces the error rate by 31% relative to the baseline in Table 33.1

Energy normalization	Normalization level	Set A	Set B	Set C	Average
CMN	Static	68.55	73.51	69.48	70.72
AGN	Static	69.46	69.84	74.15	70.55
AGN	Full	70.34	70.74	74.88	71.41
CMN	Full	68.65	73.71	69.69	70.88

Table 33.4 Word accuracy for Aurora 2, using cepstral mean normalization and an acoustic model trained on multistyle data. CMN reduces the error rate by 30% relative to the baseline in Table 33.2

Energy normalization	Normalization level	Set A	Set B	Set C	Average
CMN	Static	92.93	92.73	93.49	92.96
AGN	Static	93.15	92.61	93.52	93.01
AGN	Full	93.11	92.63	93.56	93.01
CMN	Full	92.97	92.62	93.32	92.90

best case, CMN reduces the error rate by 31% relative using a clean acoustic model, and 30% relative using a multistyle acoustic model.

Both tables compare applying CMN on the energy feature to using AGN. In most cases, using AGN is better than applying CMN on the energy term. The failure of CMN on the energy feature is most likely due to the randomness it induces on the energy of noisy speech frames. AGN tends to put noisy speech at the same level regardless of SNR, which helps the recognizer make sharp models. On the other hand, CMN will make the energy term smaller in low-SNR utterances and larger in high-SNR utterances, leading to less-effective speech models.

There are also two different stages in which CMN can be applied. One option is to use CMN on the static cepstra, before computing the dynamic cepstra. Because of the nature of CMN, this is equivalent to leaving the dynamic cepstra untouched. The other option is to use CMN on the full feature vector, after dynamic cepstra have been computed from the unnormalized static cepstra. Tables 33.3 and 33.4 both show that it is slightly better to apply the normalization to the full feature vectors.

Cepstral Variance Normalization

Cepstral variance normalization (CVN) is similar to CMN, and the two are often paired as cepstral mean and variance normalization (CMVN). CMVN uses both the sample mean and standard deviation to normalize the cepstral sequence:

$$\sigma_x{}^2 = \frac{1}{T}\sum_{t=0}^{T-1} x_t^2 - \mu_x^2 \,, \tag{33.7}$$

$$x_t = \frac{x_t - \mu_x}{\sigma_x} \,. \tag{33.8}$$

After normalization, the mean of the cepstral sequence is zero, and it has a variance of one.

Unlike CMN, CVN is not associated with addressing a particular type of distortion. It can, however, be shown empirically that it provides robustness against acoustic channels, speaker variability, and additive noise.

Tables 33.5 and 33.6 show how CMVN affects accuracy on the Aurora 2 task. Adding variance normalization to CMN reduces the error rate 8.7% relative using a clean acoustic model, and by 8.6% relative when using a multistyle acoustic model.

As with CMN, CMVN is best applied to the full feature vector, after the dynamic cepstra have been computed. Unlike CMN, the tables show that applying CMVN to the energy term is often better than using whole-utterance AGN. Because CMVN is both shifting and scaling the energy term, both the noisy speech and the noise are placed at a consistent absolute levels.

Cepstral Histogram Normalization

Cepstral histogram normalization (CHN) [33.14] takes the core ideas behind CMN and CVN, and extends them to their logical conclusion. Instead of only normalizing the first or second central moments, CHN modifies the signal such that all of its moments are normalized. As with CMN and CHN, a one-to-one transformation is independently applied to each dimension of the feature vector.

The first step in CHN is choosing a desired distribution for the data, $p_x(x)$. It is common to choose

Table 33.5 Word accuracy for Aurora 2, using cepstral mean and variance normalization and an acoustic model trained on clean data. CMVN reduces the error rate by 8.7% relative to the CMN results in Table 33.3. CMVN is much better than AGN at energy normalization, probably because it provides consistent absolute levels for both speech and noise, whereas AGN only normalizes the speech

Energy normalization	Normalization level	Set A	Set B	Set C	Average
AGN	Static	72.96	72.4	76.48	73.44
CMVN	Static	79.34	79.86	80.8	79.84
CMVN	Full	84.46	85.55	84.84	84.97
AGN	Full	72.77	72.23	77.02	73.40

Table 33.6 Word accuracy for Aurora 2, using cepstral mean and variance normalization and an acoustic model trained on multistyle data. CVMN reduces the error rate by 8.6% relative to the baseline in Table 33.4. The difference between CMVN and AGN for energy normalization is less pronounced than in Table 33.5

Energy normalization	Normalization level	Set A	Set B	Set C	Average
AGN	Static	93.34	92.79	93.62	93.18
CMVN	Static	93.33	92.57	93.24	93.01
CMVN	Full	93.80	93.09	93.70	93.50
AGN	Full	93.37	92.76	93.70	93.19

a Gaussian distribution with zero mean and unit covariance. Let $p_y(y)$ represent the actual distribution of the data to be transformed.

It can be shown that the following function $f(\cdot)$ applied to y produces features with the probability distribution function (PDF) $p_x(x)$:

$$f(y) = F_x^{-1}[F_y(y)] \,. \tag{33.9}$$

Here, $F_y(y)$ is the cumulative distribution function (CDF) of the test data. Applying $F_y(\cdot)$ to y transforms the data distribution from $p_y(y)$ to a uniform distribution. Subsequent application of $F_x^{-1}(\cdot)$ imposes a final distribution of $p_x(x)$. When the target distribution is chosen to be Gaussian as described above, the final sequence has zero mean and unit covariance, just as if CMVN were used. Additionally, every other moment would match the target Gaussian distribution.

Whole-utterance Gaussianization is easy to implement by applying (33.9) independently to each feature dimension.

First, the data is transformed using (33.10) so it has a uniform distribution. The summation counts how many frames have the i-th dimension of $\boldsymbol{y}$ less than the value in frame m, and divides by the number of frames. The resulting sequence of $y_i'[m]$ has a uniform distribution between zero and one:

$$y_i'[m] = \frac{1}{M} \sum_{m'=1}^{M} 1\big(y_i[m'] < y_i[m]\big) \,. \tag{33.10}$$

The second and final step consists of transforming $y_i'[m]$ so that it has a Gaussian distribution. This can be accomplished, as in (33.11), using an inverse Gaussian CDF G_x^{-1}:

$$y_i^{\text{CHN}}[m] = G_x^{-1}\big(y_i'[m]\big) \,. \tag{33.11}$$

Tables 33.7 and 33.8 show the results of applying CHN to the Aurora 2 task. As with CMVN, it is better to apply the normalizing transform to a full feature vector, and to avoid the use of a separate AGN step. In the end, the results are not significantly better than CMVN.

Analysis of Feature Normalization

When implementing feature normalization, it is very important to use enough data to support the chosen technique. In general, with stronger normalization algorithms, it is necessary to process longer segments of speech.

As an example, let us analyze the effect of CMN on a short utterance. Consider an utterance contains a single phoneme, such as the fricative /s/. The mean $\boldsymbol{\mu_x}$ will be very similar to the frames in this phoneme, since /s/ is quite stationary. Thus, after normalization, $\boldsymbol{\mu_x} \approx 0$. A similar result will happen for other fricatives, which means that it would be impossible to distinguish these ultrashort utterances, and the error rate will be very high. If the utterance contains more than one phoneme but is still short, the problem is not insurmountable, but the confusion among phonemes is still higher than if no CMN had been applied.

Table 33.7 Word accuracy for Aurora 2, using cepstral histogram normalization and an acoustic model trained on clean data. As with CMVN, CHN is much better than AGN at energy normalization. The CHN results on Aurora 2 are similar to the CMVN results presented in Table 33.5

Energy normalization	Normalization level	Set A	Set B	Set C	Average
AGN	Static	70.34	71.14	73.76	71.34
CHN	Static	82.49	84.06	83.66	83.35
CHN	Full	83.64	85.03	84.59	84.39
AGN	Full	69.75	70.23	74.25	70.84

Table 33.8 Word accuracy for Aurora 2, using cepstral histogram normalization and an acoustic model trained on multistyle data. The CHN results on Aurora 2 are similar to the CMVN results presented in Table 33.8

Energy normalization	Normalization level	Set A	Set B	Set C	Average
AGN	Static	93.19	92.63	93.45	93.02
CHN	Static	93.17	92.69	93.04	92.95
CHN	Full	93.61	93.21	93.49	93.43
AGN	Full	93.29	92.73	93.74	93.16

If test utterances are too short to support the chosen normalization technique, degradation will be most apparent in the clean-speech recognition results. CMVN and CHN, in particular, can significantly degrade the accuracy of the clean speech tests in Aurora 2. In cases where there is not enough data to support CMN, *Rahim* has shown [33.15] that using the recognizer's acoustic model to estimate a maximum-likelihood mean normalization is superior to conventional CMN.

Empirically, it has been found that CMN does not degrade the recognition rate on utterances from the same acoustical environment, as long as there are at least four seconds of speech frames available. CMVN and CHN require even longer segments of speech.

As we have seen, CMN can provide robustness against additive noise. It is also effective in normalizing acoustic channels. For telephone recordings, where each call has a different frequency response, the use of CMN has been shown to provide as much as 30% relative decrease in error rate. When a system is trained on one microphone and tested on another, CMN can provide significant robustness.

Interestingly, it has been found in practice that the error rate for utterances within the same environment can actually be somewhat lower. This is surprising, given that there is no mismatch in channel conditions. One explanation is that, even for the same microphone and room acoustics, the distance between the mouth and the microphone varies for different speakers, which causes slightly different transfer functions. In addition, the cepstral mean characterizes not only the channel transfer function, but also the average frequency response of different speakers. By removing the long-term speaker average, CMN can act as sort of speaker normalization.

One drawback of CMN, CMVN, and CHN is that they do not discriminate between nonspeech and speech frames in computing the utterance mean. As a result, the normalization can be affected by the ratio of speech to nonspeech frames. For instance, the mean cepstrum of an utterance that has 90% nonspeech frames will be significantly different from one that contains only 10% nonspeech frames. As a result, the speech frames will be transformed inconsistently, leading to poorer acoustic models and decreased recognition accuracy.

An extension to CMN that addresses this problem consists in computing different means for noise and speech [33.16]:

$$\boldsymbol{h}^{j+1} = \frac{1}{N_\mathrm{s}} \sum_{t \in q_\mathrm{s}} \boldsymbol{x}_t - \boldsymbol{m}_\mathrm{s} \,, \tag{33.12}$$

$$\boldsymbol{n}^{j+1} = \frac{1}{N_\mathrm{n}} \sum_{t \in q_\mathrm{n}} \boldsymbol{x}_t - \boldsymbol{m}_\mathrm{n} \,, \tag{33.13}$$

i. e., the difference between the average vector for speech frames in the utterance and the average vector $\boldsymbol{m}_\mathrm{s}$ for speech frames in the training data, and similarly for the noise frames $\boldsymbol{m}_\mathrm{n}$. Speech/noise discrimination could be done by classifying frames into speech frames and noise frames, computing the average cepstra for each, and subtracting them from the average in the training data. This procedure works well as long as the speech/noise classification is accurate. This is best done by the recognizer,

since other speech detection algorithms can fail in high background noise.

33.3.2 Voice Activity Detection

A voice activity detection (VAD) algorithm can be used to ensure that only a small, consistent percentage of the frames sent to the speech recognizer are nonspeech frames. These algorithms work by finding and eliminating any contiguous segments of nonspeech audio in the input.

VAD is an essential component in any complete noise-robust speech recognition system. It helps to normalize the percentage of nonspeech frames in the utterance, which, as discussed above, helps CMN perform better. It also directly reduces the number of extra words, or *insertion errors* produced by the recognition system. Because these nonspeech frames are never sent to the speech recognizer, they can not be mistaken for speech. As a side-effect, because the decoder does not need to process the eliminated frames, the overall recognition process is more efficient.

Most standard databases, such as Aurora 2, have been presegmented to include only a short pause before and after each utterance. As a result, the benefits of using VAD are not apparent on that data. Other tasks, such as Aurora 3, include longer segments of noise before and after the speech, and need a good VAD for optimal performance. For example, [33.17] shows how a VAD can significantly improve performance on the Aurora 3 task.

When designing a VAD, it is important to notice that the cost of making an error is not symmetric. If a nonspeech frame is mistakenly labeled as speech, the recognizer can still produce a good result because the silence hidden Markov model (HMM) may take care of it. On the other hand, if some speech frames are lost, the recognizer cannot recover from this error.

Some VAD use a single feature, such as energy, together with a threshold. More-sophisticated systems use log-spectra or cepstra, and make decisions based on Gaussian mixture models (GMMs) or neural networks. They can leverage acoustic features that the recognizer may not, such as pitch, zero crossing rate, and duration. As a result, a VAD can do a better job than the general recognition system at rejecting nonspeech segments of the signal.

Another common way to reduce the number of insertion errors is to tune the balance of insertion and deletion errors with the recognizer's *insertion penalty* parameter. In a speech recognition system, the insertion penalty is a fixed cost incurred for each recognized word. For a given set of acoustic observations, increasing this penalty causes fewer words to be recognized. In practice, the number of insertion errors can be reduced significantly while only introducing a moderate number of deletion errors.

33.3.3 Cepstral Time Smoothing

CMN, as originally formulated, requires a complete utterance to compute the cepstral mean; thus, it cannot be used in a real-time system, and an approximation needs to be used. In this section we discuss a modified version of CMN that can address this problem, as well as a set of cepstral filtering techniques that attempt to do the same thing.

Because CMN removes any constant bias from the cepstral time series, it is equivalent to a high-pass filter with a cutoff frequency arbitrarily close to zero. This insight suggests that other types of high-pass filters may also be used. One that has been found to work well in practice is the exponential filter, so the cepstral mean $\boldsymbol{\mu}_{\boldsymbol{x}}[m]$ is a function of time:

$$\boldsymbol{\mu}_{\boldsymbol{x}}[m] = \alpha \boldsymbol{x}[m] + (1-\alpha)\boldsymbol{\mu}_{\boldsymbol{x}}[m-1]\,, \tag{33.14}$$

where α is chosen so that the filter has a time constant of at least 5 s of speech. For example, when the analysis frame rate is 100 frames per second, an α of 0.999 creates a filter with a time constant of almost 7 s.

This idea of using a filter to normalizing a sequence of cepstral coefficients is quite powerful, and can be extended to provide even better results.

In addition to a high-pass CMN-like filter, it is also beneficial to add a low-pass component to the cepstral filter. This is because the rate of change of the speech spectrum over time is limited by human physiology, but the interfering noise components are not. Abrupt spectral changes are likely to contain more noise than speech. As a result, disallowing the cepstra from changing too quickly increases their effective SNR. This is the central idea behind relative spectral (RASTA) and autoregressive moving average (ARMA) filtering.

The relative spectral processing or RASTA [33.18] combines both high- and low-pass cepstral filtering into a single noncausal infinite impulse response (IIR) transfer function:

$$H(z) = 0.1(z^4)\frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.98z^{-1}}\,. \tag{33.15}$$

As in CMN, the high-pass portion of the filter is expected to alleviate the effect of convolutional noise introduced in the channel. The low-pass filtering helps to

smooth some of the fast frame-to-frame spectral changes present. Empirically, it has been shown that the RASTA filter behaves similarly to the real-time implementation of CMN, albeit with a slightly higher error rate.

ARMA filtering is similar to RASTA, in that a linear time invariant (LTI) filter is applied separately to each cepstral coefficient. The following equation:

$$H(z) = (z^2)\frac{1+z^{-1}+z^{-2}}{5-z^{-1}-z^{-2}} \tag{33.16}$$

is an example of a second-order ARMA filter. Unlike RASTA, ARMA is purely a low-pass filter. As a result, ARMA should be used in conjunction with an additional explicit CMN operation.

It was shown in [33.11] how this simple technique can do better than much more-complex robustness schemes. In spite of its simplicity, those results were the best of their time.

33.3.4 SPLICE – Normalization Learned from Stereo Data

The SPLICE technique was first proposed [33.19] as a brute-force solution to the acoustic mismatch problem. Instead of blindly transforming the data as CMN, CHN, or RASTA, SPLICE learns the joint probability distribution for noisy speech and clean speech, and uses it to map each received cepstra into a clean estimate. Like CHN, SPLICE is a nonlinear transformation technique. However, whereas CHN implicitly assumes the features are uncorrelated, SPLICE learns and uses the correlations naturally present in speech features.

The SPLICE transform is built from a model of the joint distribution of noisy cepstra $\boldsymbol{y}$ and clean cepstra $\boldsymbol{x}$. The model is a Gaussian mixture model containing K mixture components:

$$p(\boldsymbol{y},\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}|\boldsymbol{y},k)p(\boldsymbol{y},k)\,. \tag{33.17}$$

The distribution $p(\boldsymbol{y},k)$ is itself a Gaussian mixture model on $\boldsymbol{y}$, which takes the form

$$p(\boldsymbol{y},k) = p(\boldsymbol{y}|k)p(k) = N(\boldsymbol{y};\mu_k,\sigma_k)p(k)\,. \tag{33.18}$$

The conditional distribution $p(\boldsymbol{x}|\boldsymbol{y},k)$ predicts the clean feature value given a noisy observation and a Gaussian component index k:

$$p(\boldsymbol{x}|\boldsymbol{y},k) = N(\boldsymbol{x};A_k\boldsymbol{y}+b_k,\Gamma_k)\,. \tag{33.19}$$

Due to their effect on predicting $\boldsymbol{x}$ from $\boldsymbol{y}$, the matrix A_k is referred to as the *rotation matrix*, and the vector b_k is called the *offset vector*. The matrix Γ_k represents the error incurred in the prediction.

Even though the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ is nonlinear, this conditional linear prediction is sufficient. Because the GMM $p(\boldsymbol{y},k)$ effectively partitions the noisy acoustic space into K regions, each $p(\boldsymbol{x}|\boldsymbol{y},k)$ only needs to be accurate in one of these regions.

The SPLICE transform is derived from the joint distribution $p(\boldsymbol{x},\boldsymbol{y},k)$ by finding the expected value of the clean speech $\boldsymbol{x}$, given the current noisy observation $\boldsymbol{y}$. This approach finds the minimum mean-squared error (MMSE) estimate for x under the model, an approach pioneered by the codeword-dependent cepstral normalization (CDCN) [33.20] and multivariate-gaussian-based cepstral normalization (RATZ) [33.21] algorithms:

$$\begin{aligned}\hat{\boldsymbol{x}}_{\text{MMSE}} &= E\{\boldsymbol{x}|\boldsymbol{y}\} = \sum_{k=1}^{K} E\{\boldsymbol{x}|\boldsymbol{y},k\}p(k|\boldsymbol{y}) \\ &= \sum_{k=1}^{K}(A_k\boldsymbol{y}+b_k)p(k|\boldsymbol{y})\,,\end{aligned} \tag{33.20}$$

where the posterior probability $p(k|\boldsymbol{y})$ is given by

$$p(k|\boldsymbol{y}) = \frac{p(\boldsymbol{y},k)}{\sum_{k'=1}^{K} p(\boldsymbol{y},k')}\,. \tag{33.21}$$

Another option is to find an approximate maximum likelihood estimate for $\boldsymbol{x}$ under the model, which is similar to the fixed codeword-dependent cepstral normalization algorithm algorithm [33.20]. A good approximate solution to

$$\hat{\boldsymbol{x}}_{\text{ML}} = \max_x p(\boldsymbol{x}|\boldsymbol{y}) \tag{33.22}$$

is

$$\hat{\boldsymbol{x}}_{\text{ML}} \approx A_{\hat{k}}\boldsymbol{y}+b_{\hat{k}}\,, \tag{33.23}$$

where

$$\hat{k} = \arg\max_k p(\boldsymbol{y},k)\,. \tag{33.24}$$

Whereas the approximate maximum-likelihood (ML) estimate above involves a single affine transformation, the MMSE solution requires K transformations. Even though the MMSE estimate produces more-accurate recognition results, the approximate ML estimate may be substituted when the additional computational cost is prohibitive.

Another popular method for reducing the additional computational cost is to replace the learned rotation matrix A_k with the identity matrix I. This produces systems that are more efficient, with only a modest degradation in performance [33.19].

A number of different algorithms [33.20, 21] have been proposed that vary in how the parameters μ_k, Σ_k, A_k, b_k, and Γ_k are estimated.

If stereo recordings are available from both the clean signal and the noisy signal, then we can estimate μ_k and Σ_k by fitting a mixture Gaussian model to $\boldsymbol{y}$ using standard maximum-likelihood training techniques. Then A_k, b_k, and Γ_k can be estimated directly by linear regression of $\boldsymbol{x}$ and $\boldsymbol{y}$. The FCDCN algorithm [33.20, 22] is a variant of this approach when it is assumed that $\Sigma_k = \sigma^2 I$, $\Gamma_k = \gamma^2 I$, and $A_k = I$, so that μ_k and b_k are estimated through a vector quantization (VQ) procedure and b_k is the average difference $(\boldsymbol{y} - \boldsymbol{x})$ for vectors $\boldsymbol{y}$ that belong to mixture component k.

Often, stereo recordings are not available and we need other means of estimating the parameters μ_k, Σ_k, A_k, b_k, and Γ_k. CDCN [33.22] and VTS enhancement [33.21] are examples of algorithms that use a model of the environment (Sect. 33.4). This model defines a nonlinear relationship between $\boldsymbol{x}$, $\boldsymbol{y}$ and the environmental parameters $\boldsymbol{n}$ for the noise. The CDCN method also uses an MMSE approach where the correction vector is a weighted average of the correction vectors for all classes. Other methods that do not require stereo recordings or a model of the environment are presented in [33.21].

Recent improvements in training the transformation parameters have been proposed using discriminative training [33.23–25]. In this case, the noisy-speech GMM is developed from the noisy training data, or from a larger acoustic model developed from that data. The correction parameters are then initialized to zero, which corresponds to the identity transformation. Subsequently, they are trained to maximize a discriminative criterion, such as minimum classification error (MCE) [33.24], maximum mutual information (MMI) [33.23], or minimum phone error (MPE) [33.25]. It has been shown that this style of training produces superior results to the two-channel MMSE approach, and can even increase accuracy in noise-free test cases. The main disadvantage of this approach is that it is easy to overtrain the transformation, which can actually reduce robustness of the system to noise types that do not occur in the training set. This can be easily avoided by the customary use of regularization and separate training, development, and test data.

Although the SPLICE transform is discussed here, there are many alternatives available. The function to map from $\boldsymbol{y}$ to $\boldsymbol{x}$ can be approximated with a neural network [33.26], or as a mixture of Gaussians as in probabilistic optimum filtering (POF) [33.27], FCDCN [33.28], RATZ [33.21], and stochastic vector mapping (SVM) [33.24].

33.4 A Model of the Environment

Sections 33.2 and 33.3 described techniques that address the problem of additive noise by blindly reducing the acoustic mismatch between the acoustic model and the expected test data. These techniques are powerful and general, and are often used to solve problems other than additive noise. However, the more-effective solutions generally require more data to operate properly.

In this section, we use knowledge of the nature of the degradation to derive the relationship between the clean and observed signals in the power-spectrum, log-filterbank, and cepstral domains. Later, Sects. 33.5 and 33.6 show how several related methods leverage this model to produce effective noise robust speech recognition techniques.

In the acoustic environment, the clean speech signal coexists with many other sound sources. They mix linearly in the air, and a mixture of all these signals is picked up by the microphone. For our purposes, the clean speech signal that would have existed in the absence of noise is denoted by the symbol $x[m]$, and all of the other noises that are picked up by the microphone are represented by the symbol $n[m]$. Because of the linear mixing, the observed signal $y[m]$ is simply the sum of the clean speech and noise:

$$y[m] = x[m] + n[m] \,. \tag{33.25}$$

Unfortunately, the additive relationship of (33.25) is destroyed by the nonlinear process of extracting cepstra from $y[m]$. Figure 33.2 demonstrates the path that the noisy signal takes on its way to becoming a cepstral feature vector observation vector. It consists of dividing the received signal into frames, performing a frequency analysis and warping, applying a logarithmic compression, and finally a decorrelation and dimensionality reduction.

The first stage of feature extraction is framing. The signal is split into overlapping segments of about 25 ms each. These segments are short enough that, within each frame of data, the speech signal is approximately stationary. Each frame is passed through a discrete Fourier transform (DFT), where the time-domain signal becomes a complex-valued function of discrete fre-

quency k. Concentrating our analysis on a single frame of speech $Y[k]$, clean speech and noise are still additive:

$$Y[k] = X[k] + N[k] \,. \tag{33.26}$$

The next processing step is to turn the observed complex spectra into real-valued power spectra, through the application of a magnitude-squared operation. The power spectrum of the observed signal is

$$|Y[k]|^2 = |X[k]|^2 + |N[k]|^2 + 2|X[k]||N[k]| \cos\theta \,, \tag{33.27}$$

a function of the power spectrum of the clean speech and of the noise, as well as a *cross-term*. This cross-term is a function of the clean speech and noise magnitudes, as well as their relative phase θ. When the clean speech and noise are uncorrelated, the expected value of the cross-term is zero:

$$E\{X[k]N^*[k]\} = 0 \,. \tag{33.28}$$

However, for a particular frame it can have a considerable magnitude.

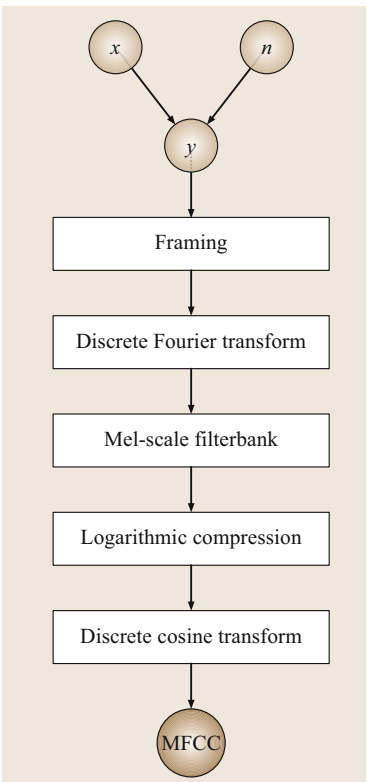

Fig. 33.2 Clean speech x and environmental noise n mix to produce the noisy signal y, which is turned into mel-frequency cepstral coefficients (MFCC) through a sequence of processing steps

It is uncommon to pass the Fourier spectra directly to the speech recognition system. Instead, it is standard to apply a mel-frequency filterbank and a logarithmic compression to create log mel-frequency filterbank (LMFB) features. The mel-frequency filterbank implements dimensionality reduction and frequency warping as a linear projection of the power spectrum. The relationship between the i-th LMFB coefficient y_i and the observed noisy spectra $Y[k]$ is given by

$$y_i = \ln\left(\sum_k w_k^i |Y[k]|^2\right) \,, \tag{33.29}$$

where the scalar w_k^i is the k-th coefficient of the i-th filter in the filterbank.

Figure 33.3 reveals a typical structure of the matrix $\boldsymbol{W}$ containing the scalars w_k^i. It compresses 128 FFT bins into 23 mel-frequency spectral features with a resolution that varies over frequency. At low frequencies, high resolution is preserved by using only a small number of FFT bins for each mel-frequency feature. As the Fourier frequency increases, more FFT bins are used.

Using (33.29) and (33.27), we can deduce the relationship among the LMFB for clean speech, noise, and noisy observation. The noisy LMFB energies are a function of the two unobserved LMFB, and a cross-term that depends on a third nuisance parameter α_i:

$$\exp(y_i) = \exp(x_i) + \exp(n_i) + 2\alpha_i \exp\left(\frac{x_i + n_i}{2}\right) \,, \tag{33.30}$$

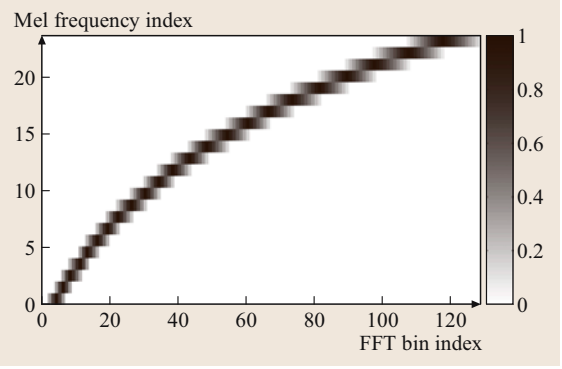

Fig. 33.3 A graphical representation of the mel-frequency filterbank used to compute MFCC features. This filterbank compresses 128 spectral bins into 23 mel-frequency coefficients

where

$$\alpha_i = \frac{\sum_k w_k^i |X[k]N[k]| \cos\theta_k}{\sqrt{\sum_k w_k^i |X[k]|^2}\sqrt{\sum_k w_k^i |N[k]|^2}} \,. \tag{33.31}$$

As a consequence of this model, when we observe y_i there are actually *three* unobserved random variables. The first two are obvious: the clean log spectral energy and the noise log spectral energy that would have been produced in the absence of mixing. The third variable α_i accounts for the unknown phase between the two sources.

If the magnitude spectra are assumed constant over the bandwidth of a particular filterbank, the definition of α_i collapses to a weighted sum of several independent random variables

$$\alpha_i = \frac{\sum_k w_k^i \cos\theta_k}{\sum_j w_j^i} \,. \tag{33.32}$$

According to the central limit theorem, a sum of many independent random variables will tend to be normally distributed. The number of effective terms in (33.32) is controlled by the width of the i-th filterbank. Since filterbanks with higher center frequencies have wider bandwidths, they should be more nearly Gaussian. Figure 33.4 shows the true distributions of α for a range of filterbanks. They were estimated from a joint set of noise and clean speech, and noisy speech taken from the Aurora 2 training data by solving (33.30) for the unknown α_i. As expected, because the higher-frequency

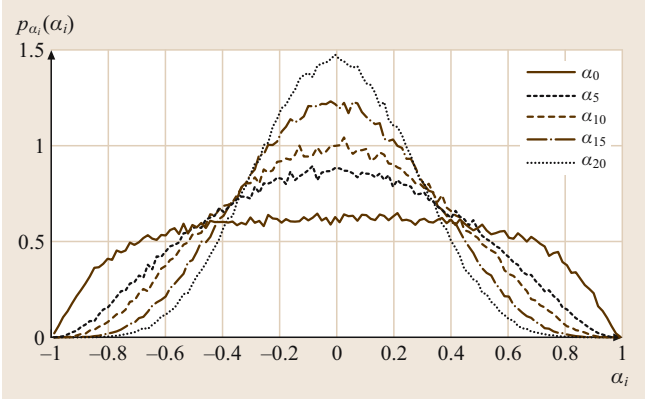

Fig. 33.4 An estimation of the true distribution of α using (33.30) and data from the Aurora 2 corpus. Higher numbered filterbanks cover more frequency bins, are more nearly Gaussian, and have smaller variances

higher-bandwidth filters include more FFT bins, they produce distributions that are more nearly Gaussian.

After some algebraic manipulation, it can be shown that

$$y_i = x_i + \ln\left[1 + \exp(n_i - x_i) + 2\alpha_i \exp\left(\tfrac{n_i - x_i}{2}\right)\right] \,. \tag{33.33}$$

Many current speech enhancement algorithms ignore the cross-term entirely, as in

$$y_i = x_i + \ln[1 + \exp(n_i - x_i)] \,. \tag{33.34}$$

This is entirely appropriate in situations where the observed frequency component is dominated by either noise or speech. In these cases, the cross-term is indeed negligible. But, in cases where the speech and noise components have similar magnitudes, the cross-term can have a considerable effect on the observation.

To create cepstral features from these LMFB features, apply a discrete cosine transform. The j-th cepstral coefficient is calculated with the following sum, where c_{ij} are the DCT coefficients:

$$y_j^{\text{MFCC}} = \sum_i c_{ji} y_i^{\text{LMFB}} \,. \tag{33.35}$$

By convention, this transform includes a truncation of the higher-order DCT coefficients, a process historically referred to as *cepstral liftering*. For example, the Aurora 2 system uses 23 LMFB and keeps only the first 13 cepstral coefficients.

Due to this cepstral truncation, the matrix $\mathbf{C}$ created from the scalars c_{ij} is not square and cannot be inverted. But, we can define a right-inverse matrix $\mathbf{D}$ with elements d_{ij}, such that $\mathbf{CD} = \mathbf{I}$. That is, if $\mathbf{D}$ is applied to a cepstral vector, an approximate spectral vector is produced, from which the projection $\mathbf{C}$ will recreate the original cepstral vector. Using $\mathbf{C}$ and $\mathbf{D}$, (33.34) can be expressed for cepstral vectors $\boldsymbol{x}$, $\boldsymbol{n}$, and $\boldsymbol{y}$ as:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{g}(\boldsymbol{x} - \boldsymbol{n}) \,, \tag{33.36}$$

$$\boldsymbol{g}(z) = \mathbf{C}\ln[1 + \exp(-\mathbf{D}z)] \,. \tag{33.37}$$

Although $\mathbf{CD} = \mathbf{I}$, it is not the case that $\mathbf{DC} = \mathbf{I}$. In particular, (33.37) is not exactly correct as it needs an extra term $\bar{\boldsymbol{D}}\bar{\boldsymbol{z}}$ that contains the missing columns from $\mathbf{D}$ and higher-order cepstral coefficients from $\boldsymbol{z}$. If only the truncated cepstrum $\boldsymbol{z}$ is available, then $\bar{\boldsymbol{D}}\bar{\boldsymbol{z}}$ is a random error term that is generally ignored.

In (33.37), the nonlinearity $\boldsymbol{g}$ was introduced, which maps the signal-to-noise ratio $(\boldsymbol{x} - \boldsymbol{n})$ into the difference between clean and noisy speech $(\boldsymbol{y} - \boldsymbol{x})$. When the noise

cepstrum $\boldsymbol{n}$ has significantly less energy than the speech vector $\boldsymbol{x}$, the function $\boldsymbol{g}(\boldsymbol{x}-\boldsymbol{n})$ approaches zero, and $\boldsymbol{y} \approx \boldsymbol{x}$. Conversely, when noise dominates speech, $\boldsymbol{g}(\boldsymbol{x}-\boldsymbol{n}) \approx \boldsymbol{n}-\boldsymbol{x}$, and $\boldsymbol{y} \approx \boldsymbol{n}$.

33.5 Structured Model Adaptation

This section compares two popular methods for structured model adaptation: log-normal parallel model combination and vector Taylor-series adaptation. Both can achieve good adaptation results with only a small amount of data.

By using a set of clean-speech acoustic models and a noise model, both methods approximate the model parameters that would have been obtained by training with corrupted speech.

33.5.1 Analysis of Noisy Speech Features

Figures 33.5 and 33.6 depict how additive noise can distort the spectral features used in ASR systems. In both figures, the simulated clean speech x and noise n are assumed to follow Gaussian distributions:

$$p_x(x) = N(x; \mu_x, \sigma_x) ,$$
$$p_n(n) = N(n; \mu_n, \sigma_n) .$$

After mixing, the noisy speech y distribution is a distorted version of its clean speech counterpart. It is usually shifted, often skewed, and sometimes bimodal. It is clear that a pattern recognition system trained on clean speech will be confused by noisy speech input.

In Fig. 33.5, the clean speech and noise mix to produce a bimodal distribution. We fix $\mu_n = 0\,\text{dB}$, since it is only a relative level, and set $\sigma_n = 2\,\text{dB}$, a typical value. We also set $\mu_x = 25\,\text{dB}$ and see that the resulting distribution is bimodal when σ_x is very large. Fortunately, for modern speech recognition systems that have many Gaussian components, σ_x is never that large and the resulting distribution is often unimodal.

Figure 33.6 demonstrates the more-common skew and offset distortions. Again, the noise parameters are $\mu_n = 0\,\text{dB}$ and $\sigma_n = 2\,\text{dB}$, but a more-realistic value for $\sigma_x = 5\,\text{dB}$ is used. We see that the distribution is always unimodal, although not necessarily symmetric, particularly for low SNR ($\mu_x - \mu_n$).

33.5.2 Log-Normal Parallel Model Combination

Parallel model combination (PMC) [33.29] is a general method to obtain the distribution of noisy speech given the distribution of clean speech and noise as mixtures of Gaussians.

As discussed in Sect. 33.5.1, even if the clean speech and noise cepstra follow Gaussian distributions, the noisy speech will not be Gaussian. The log-normal PMC method nevertheless assumes that the resulting noisy observation is Gaussian. It transforms the clean speech and noise distributions into the linear spectral domain, mixes them, and trans-

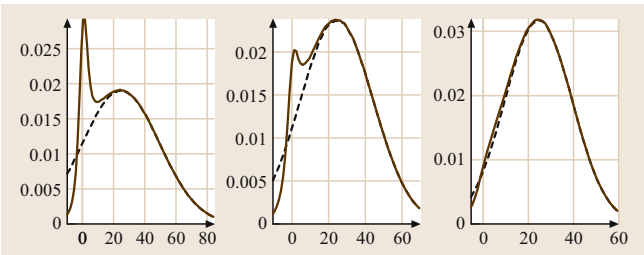

Fig. 33.5 Distributions of the corrupted log-spectra y (*solid lines*) from uncorrupted log-spectra x (*dashed lines*) using simulated data and (33.27). The distribution of the noise log-spectrum n is Gaussian with mean 0 dB and standard deviation of 2 dB. The distribution of the clean log-spectrum x is Gaussian with mean 25 dB and standard deviations of 25, 20, and 15 dB, respectively (the x-axis is expressed in dB). The first two distributions are bimodal, whereas the 15 dB case is more approximately Gaussian

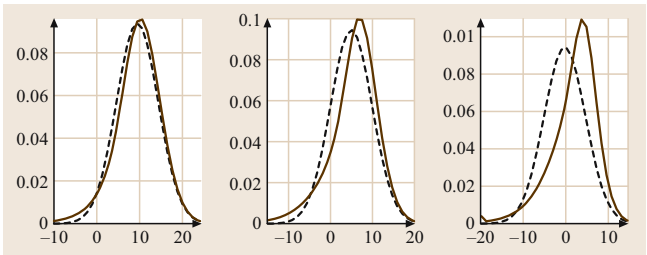

Fig. 33.6 Distributions of the corrupted log-spectra y (*solid lines*) from uncorrupted log-spectra x (*dashed lines*) using simulated data and (33.27). The distribution of the noise log-spectrum n is Gaussian with mean 0 dB and standard deviation of 2 dB. The distribution of the clean log-spectrum is Gaussian with standard deviation of 5 dB and means of 10, 5, and 0 dB, respectively. As the average SNR decreases, the Gaussian experiences more shift and skew

forms them back into a distribution on noisy speech cepstra.

If the mean and covariance matrix of the cepstral noise vector $\boldsymbol{n}$ are given by $\mu_{\boldsymbol{n}}^{\mathrm{c}}$ and $\Sigma_{\boldsymbol{n}}^{\mathrm{c}}$, respectively, its mean and covariance matrix in the log spectral domain can be approximated by

$$\mu_{\boldsymbol{n}}^{\mathrm{l}} = \mathbf{D}\mu_{\boldsymbol{n}}^{\mathrm{c}} , \tag{33.38}$$

$$\Sigma_{\boldsymbol{n}}^{\mathrm{l}} = \mathbf{D}\Sigma_{\boldsymbol{n}}^{\mathrm{c}}\mathbf{D}^{\mathrm{T}} . \tag{33.39}$$

As in Sect. 33.4, $\mathbf{D}$ is a right inverse for the noninvertible cepstral rotation $\mathbf{C}$.

In the linear domain $\boldsymbol{N} = \mathrm{e}^{\boldsymbol{n}}$, the noise distribution is log-normal, with a mean vector $\boldsymbol{\mu}_{\boldsymbol{N}}$ and covariance matrix $\Sigma_{\boldsymbol{N}}$ given by

$$\mu_{\boldsymbol{N}}[i] = \exp\left(\mu_{\boldsymbol{N}}^{\mathrm{l}}[i] + \frac{1}{2}\Sigma_{\boldsymbol{N}}^{\mathrm{l}}[i,i]\right) , \tag{33.40}$$

$$\Sigma_{\boldsymbol{N}}[i,j] = \mu_{\boldsymbol{N}}[i]\mu_{\boldsymbol{N}}[j]\left[\exp\left(\Sigma_{\boldsymbol{N}}^{\mathrm{l}}[i,j]\right) - 1\right] . \tag{33.41}$$

As a result, we have the exact parameters for the distribution of noise features in the linear spectral domain. The cepstral Gaussian distribution for the clean speech can be transformed from $\mu_{\boldsymbol{x}}^{\mathrm{c}}$ and $\Sigma_{\boldsymbol{x}}^{\mathrm{c}}$ to $\mu_{\boldsymbol{X}}$ and $\Sigma_{\boldsymbol{X}}$ using expressions similar to (33.38) through (33.41).

Using the basic assumption that the noise and speech waveforms are additive, the spectral vector $\boldsymbol{Y}$ is given by $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{N}$. Without any approximation, the mean and covariance of $\boldsymbol{Y}$ is given by

$$\mu_{\boldsymbol{Y}} = \mu_{\boldsymbol{X}} + \mu_{\boldsymbol{N}} , \tag{33.42}$$

$$\Sigma_{\boldsymbol{Y}} = \Sigma_{\boldsymbol{X}} + \Sigma_{\boldsymbol{N}} . \tag{33.43}$$

Although the sum of two log-normal distributions is not log-normal, the *log-normal approximation* [33.29] consists in assuming that $\boldsymbol{Y}$ is log-normal. In this case we can apply the inverse formulae of (33.40) and (33.41) to obtain the mean and covariance matrix in the log-spectral domain:

$$\Sigma_{\boldsymbol{y}}^{\mathrm{l}} \approx \ln\left(\frac{\Sigma_{\boldsymbol{Y}}[i,j]}{\mu_{\boldsymbol{Y}}[i]\mu_{\boldsymbol{Y}}[j]} + 1\right) , \tag{33.44}$$

$$\mu_{\boldsymbol{y}}^{\mathrm{l}}[i] \approx \ln(\mu_{\boldsymbol{y}}[i]) - \frac{1}{2}\ln\left(\frac{\Sigma_{\boldsymbol{Y}}[i,j]}{\mu_{\boldsymbol{Y}}[i]\mu_{\boldsymbol{Y}}[j]} + 1\right) , \tag{33.45}$$

and finally return to the cepstrum domain applying the inverse of (33.38) and (33.39):

$$\mu_{\boldsymbol{y}}^{\mathrm{c}} = \mathbf{C}\mu_{\boldsymbol{y}}^{\mathrm{l}} , \tag{33.46}$$

$$\Sigma_{\boldsymbol{y}}^{\mathrm{c}} = \mathbf{C}\Sigma_{\boldsymbol{y}}^{\mathrm{l}}\mathbf{C}^{\mathrm{T}} . \tag{33.47}$$

The log-normal approximation cannot be used directly for the delta and delta–delta cepstrum. Another variant that can be used in this case and is more accurate than the log-normal approximation is the *data-driven parallel model combination* (DPMC) [33.29]. DPMC uses Monte Carlo simulation to draw random cepstrum vectors from both the clean-speech HMM and noise distribution to create cepstrum of the noisy speech by applying (33.36) to each sample point. The mean and covariance of these simulated noisy cepstra are then used as adapted HMM parameters. In that respect, it is similar to other model adaptation schemes, but not as accurate as the matched condition training from Sect. 33.2.1 because the distribution is only an approximation.

Although it does not require a lot of memory, DPMC carries with it large computational burden. For each transformed Gaussian component, the recommended number of simulated training vectors is at least 100 [33.29]. A way of reducing the number of random vectors needed to obtain good Monte Carlo simulations is proposed in [33.30].

33.5.3 Vector Taylor-Series Model Adaptation

Vector Taylor-series (VTS) model adaptation is similar in spirit to the log-normal PMC in Sect. 33.5.2, but instead of a log-normal approximation it uses a first-order Taylor-series approximation of the model presented from Sect. 33.4.

This model described the relationship between the cepstral vectors $\boldsymbol{x}$, $\boldsymbol{n}$, and $\boldsymbol{y}$ of the clean speech, noise, and noisy speech, respectively:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{g}(\boldsymbol{n} - \boldsymbol{x}) , \tag{33.48}$$

where the nonlinear function $\boldsymbol{g}(\boldsymbol{z})$ is given by

$$\boldsymbol{g}(\boldsymbol{z}) = \mathbf{C}\ln[1 + \exp(\mathbf{D}\boldsymbol{z})] . \tag{33.49}$$

As in Sect. 33.4, $\mathbf{C}$ and $\mathbf{D}$ are the cepstral rotation and its right inverse, respectively.

Moreno [33.31] first suggested the use of Taylor series to approximate the nonlinearity in (33.49), though he applies it in the spectral instead of the cepstral domain. Since then, many related techniques have appeared that build upon this core idea [33.32–37]. They mainly differ in how speech and noise are modeled, as well as which assumptions are made in the derivation of the nonlinearity to approximate [33.38].

To apply the vector Taylor-series approximation, first assume that $\boldsymbol{x}$ and $\boldsymbol{n}$ are Gaussian random vectors with means $\{\mu_{\boldsymbol{x}}, \mu_{\boldsymbol{n}}\}$ and covariance matrices $\{\Sigma_{\boldsymbol{x}}, \Sigma_{\boldsymbol{n}}\}$,

Table 33.9 Word accuracy for Aurora 2, using VTS model adaptation on an acoustic model trained on clean data

SNR (dB)	Test set A	Test set B	Test set C	Average
Clean	99.63	99.63	99.63	99.63
20	97.86	97.59	98.17	97.88
15	96.47	95.64	96.99	96.37
10	93.47	91.98	93.30	92.91
5	86.87	85.48	85.49	85.94
0	71.68	68.90	68.72	69.77
−5	36.84	33.04	39.04	35.97
Average (0–20)	89.27	87.92	88.53	88.58

and furthermore that $\boldsymbol{x}$ and $\boldsymbol{n}$ are independent. After algebraic manipulation it can be shown that the Jacobian of (33.48) with respect to $\boldsymbol{x}$ and $\boldsymbol{n}$ evaluated at $\{\boldsymbol{x}=\mu_{\boldsymbol{x}}, \boldsymbol{n}=\mu_{\boldsymbol{n}}\}$ can be expressed as:

$$\left.\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right|_{(\mu_{\boldsymbol{x}},\mu_{\boldsymbol{n}})} = \mathbf{G}\,, \tag{33.50}$$

$$\left.\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{n}}\right|_{(\mu_{\boldsymbol{x}},\mu_{\boldsymbol{n}})} = \mathbf{I}-\mathbf{G}\,, \tag{33.51}$$

where the matrix $\mathbf{G}$ is given by

$$\mathbf{G}=\mathbf{CFD}\,. \tag{33.52}$$

In (33.52), $\mathbf{F}$ is a diagonal matrix whose elements are given by vector $\boldsymbol{f}(\mu)$, which in turn is given by

$$\boldsymbol{f}(\mu_{\mathrm{n}}-\mu_x)=\frac{1}{1+\exp[\mathbf{D}(\mu_{\mathrm{n}}-\mu_x)]}\,. \tag{33.53}$$

Using (33.50) and (33.51) we can then approximate (33.48) by a first-order Taylor-series expansion around $\{\mu_{\mathrm{n}}, \mu_x\}$ as

$$\begin{aligned}\boldsymbol{y} \approx{}& \mu_x+\boldsymbol{g}(\mu_{\mathrm{n}}-\mu_x)+\mathbf{G}(\boldsymbol{x}-\mu_x)\\ &+(\mathbf{I}-\mathbf{G})(\boldsymbol{n}-\mu_{\mathrm{n}})\,.\end{aligned} \tag{33.54}$$

The mean of $\boldsymbol{y}$, μ_y, can be obtained from (33.54) as

$$\mu_y \approx \mu_x+\boldsymbol{g}(\mu_{\mathrm{n}}-\mu_x)\,, \tag{33.55}$$

and its covariance matrix Σ_y by

$$\Sigma_y \approx \mathbf{G}(\Sigma_x)\mathbf{G}^{\mathrm{T}}+(\mathbf{I}-\mathbf{G})\Sigma_n(\mathbf{I}-\mathbf{G})^{\mathrm{T}}\,. \tag{33.56}$$

Note that, even if Σ_x and Σ_n are diagonal, Σ_y is not. Nonetheless, it is common to make a diagonal assumption. That way, we can transform a clean HMM to a corrupted HMM that has the same functional form and use a decoder that has been optimized for diagonal covariance matrices.

To compute the means and covariance matrices of the delta and delta–delta parameters, let us take the derivative of the approximation of $\boldsymbol{y}$ in (33.54) with respect to time:

$$\frac{\partial \boldsymbol{y}}{\partial t} \approx \mathbf{G}\frac{\partial \boldsymbol{x}}{\partial t}\,, \tag{33.57}$$

so that the delta cepstrum computed through $\Delta\boldsymbol{x}_t=\boldsymbol{x}_{t+2}-\boldsymbol{x}_{t-2}$, is related to the derivative [33.39] by

$$\Delta\boldsymbol{x} \approx 4\frac{\partial \boldsymbol{x}}{\partial t}\,, \tag{33.58}$$

so that

$$\mu_{\Delta\boldsymbol{y}} \approx \mathbf{G}\mu_{\Delta\boldsymbol{x}}\,, \tag{33.59}$$

and similarly

$$\Sigma_{\Delta\boldsymbol{y}} \approx \mathbf{G}\Sigma_{\Delta\boldsymbol{x}}\mathbf{G}^{\mathrm{T}}+(\mathbf{I}-\mathbf{G})\Sigma_{\Delta\boldsymbol{n}}(\mathbf{I}-\mathbf{G})^{\mathrm{T}}\,. \tag{33.60}$$

Similarly, for the delta–delta cepstrum, the mean is given by

$$\mu_{\Delta^2\boldsymbol{y}} \approx \mathbf{G}\mu_{\Delta^2\boldsymbol{x}}\,, \tag{33.61}$$

and the covariance matrix by

$$\Sigma_{\Delta^2\boldsymbol{y}} \approx \mathbf{G}\Sigma_{\Delta^2\boldsymbol{x}}\mathbf{G}^{\mathrm{T}}+(\mathbf{I}-\mathbf{G})\Sigma_{\Delta^2\boldsymbol{n}}(\mathbf{I}-\mathbf{G})^{\mathrm{T}}\,. \tag{33.62}$$

Table 33.9 demonstrates the effectiveness of VTS model adaptation on the Aurora 2 task. For each test noise condition, a diagonal Gaussian noise model was estimated from the first and last 200 ms of all the test data. Then, each Gaussian component in the acoustic model was transformed according to the algorithm presented above to create a noise condition specific acoustic model. In theory, better results could have been obtained by estimating a new noise model from each utterance and creating an utterance specific acoustic model. However, the computational cost would be much greater.

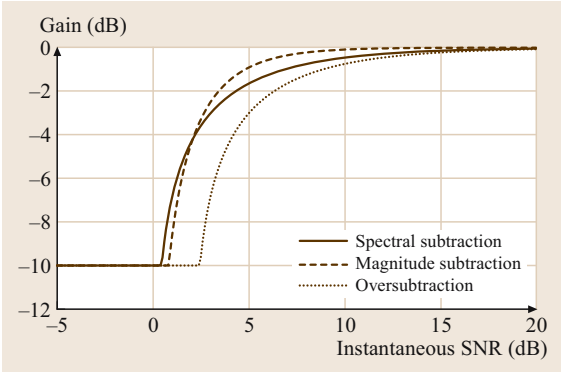

Fig. 33.7 Magnitude of the spectral subtraction filter gain as a function of the input instantaneous SNR for $A = 10$ dB, for the spectral subtraction of (33.68), magnitude subtraction of (33.71), and over-subtraction of (33.72) with $\beta = 2$ dB

Compared to the unadapted results of Table 33.1, the performance of the VTS adapted models is improved at each SNR level on every test set. At 20 dB SNR, the VTS model adaptation reduces the number of errors by 61%. Averaged over 0–20 dB, the adapted system has an average of 73% fewer errors.

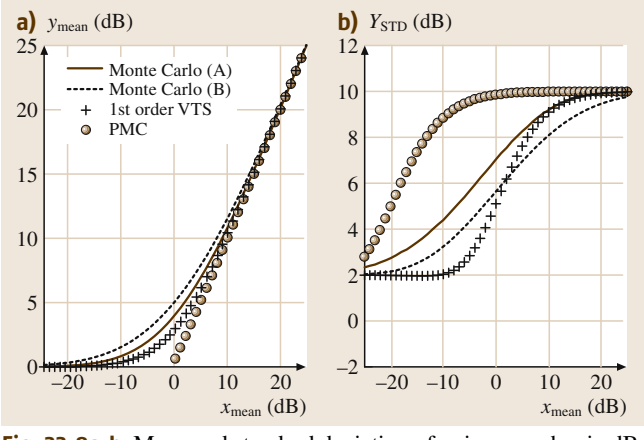

Fig. 33.8a,b Mean and standard deviation of noisy speech y in dB. The distribution of the noise log-spectrum n is Gaussian with mean 0 dB and standard deviation of 2 dB. The distribution of the clean speech log-spectrum x is Gaussian, having a standard deviation of 10 dB and a mean varying from −25 to 25 dB. The first-order VTS approximation is a better estimate for a Monte Carlo simulation of (33.27) when the cross-term is ignored **(b)**, although both VTS and lognormal PMC underestimate the standard deviation compared to when the cross-term is included **(a)**

The VTS model adaptation results also better than any feature normalization technique using clean training data from Sect. 33.3. This is a direct result of using the model from Sect. 33.4 to make better use of the available adaptation data.

33.5.4 Comparison of VTS and Log-Normal PMC

It is difficult to visualize how good the VTS approximation is, given the nonlinearity involved. To provide some insight, Figs. 33.8 and 33.9 provide a comparison between the log-normal approximation, the VTS approximation, and Monte Carlo simulations of (33.27). For simplicity, only a single dimension of the log spectral features are shown.

Figure 33.8 shows the mean and standard deviation of the noisy log-spectral energy y in dB as a function of the mean of the clean log-spectral energy x with a standard deviation of 10 dB. The log-spectral energy of the noise n is Gaussian with mean 0 dB and standard deviation 2 dB.

We see that the VTS approximation is more accurate than the log-normal approximation for the mean, especially in the region around 0 dB SNR. For the standard deviation, neither approximation is very accurate. Because the VTS models fail to account for the cross-term of (33.27), both tend to underestimate the true noisy standard deviation.

Figure 33.9 is similar to Fig. 33.8 except that the standard deviation of the clean log-energy x is only 5 dB, a more-realistic number for a speech recognition system. In this case, both the log-normal approximation and the first-order VTS approximation are good estimates of the mean of y. Again, mostly because they ignore the cross-term, neither approximation gives a reliable estimate for the standard deviation of y in the region between −20 dB and 20 dB.

33.5.5 Strategies for Highly Nonstationary Noises

So far, this section has dealt only with stationary noise. That is, it assumed the noise cepstra for a given utterance are well modeled by a Gaussian distribution. In practice, though, there are many nonstationary noises that do not fit that model. What is worse, nonstationary noises can match a random word in the system's lexicon better than the silence model. In this case, the benefit of using speech recognition vanishes quickly.

One solution to the problem of nonstationary noise is to use a more-complex noise model with standard model adaptation algorithms. For instance, the Gaussian noise model can be replaced with a Gaussian mixture model (GMM) or hidden Markov model (HMM).

With a GMM noise model, the decoding becomes more computationally intensive. Before, a Gaussian noise model transformed each component of the clean speech model into one component in the adapted noisy speech model. Now, assume that the noise is independent of speech and is modeled by a mixture of M Gaussian components. Each Gaussian component in the clean speech model will mix independently with every component of the noise GMM. As a result, the acoustic model will grow by a factor of M.

An HMM noise model can provide a much better fit to nonstationary noises [33.40, 41]. However, to efficiently use an HMM noise model, the decoder needs to be modified to perform a three-dimensional Viterbi search which evaluates every possible speech state and noise state at every frame. The computational complexity of performing this *speech/noise decomposition* is very large, though in theory it can handle nonstationary noises quite well.

Alternatively, dedicated whole-word garbage models can bring some of the advantages of an HMM noise model without the additional cost of a three-dimensional Viterbi search. In this technique [33.42], new words are created in the acoustic and language models to cover nonstationary noises such as lip smacks, throat clearings, coughs, and filler words such as *uhm* and *uh*. These nuisance words can be successfully recognized and ignored during non-speech regions, where they tend to cause the most damage.

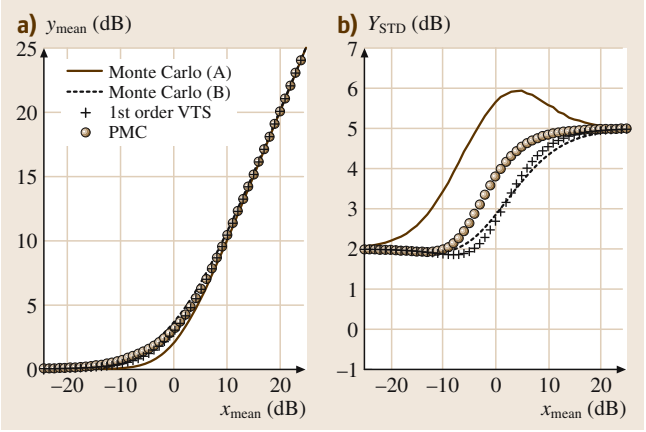

Fig. 33.9a,b Mean and standard deviation of noisy speech y in dB. The distribution of the noise log-spectrum n is Gaussian with mean of 0 dB and standard deviation of 2 dB. The distribution of the clean log-spectrum x is Gaussian with a standard deviation of 5 dB and a mean varying from −25 dB to 25 dB. Both log-normal PMC and the first-order VTS approximation make good estimates compared to a Monte Carlo simulation of (33.27) when the cross-term is ignored **(b)**, although the standard deviation is revealed as an underestimate when the cross-term is included **(a)**

33.6 Structured Feature Enhancement

This section presents several popular methods for structured feature enhancement. Whereas the techniques of Sect. 33.5 adapt the recognizer's acoustic model parameters, the techniques discussed here use mathematical models of the noise corruption to enhance the features before they are presented to the recognizer.

Methods in this class have a rich history. *Boll* [33.43] pioneered the use of spectral subtraction for speech enhancement almost thirty years ago, and it is still found in many systems today. *Ephraim* and *Malah* [33.44] invented their logarithmic minimum mean-squared error short-time spectral-amplitude (logMMSE STSA) estimator shortly afterwards, which forms the basis of many of today's advanced systems. Whereas these earlier approaches use weak speech models, today's most promising systems use stronger models, coupled with vector Taylor series speech enhancement [33.45].

Some of these techniques were developed for speech enhancement for human consumption. Unfortunately, what sounds good to a human can confuse an automatic speech recognition system, and automatic systems can tolerate distortions that are unacceptable to human listeners.

33.6.1 Spectral Subtraction

Spectral subtraction is built on the assumption that the observed power spectrum is approximately the sum of the power spectra for the clean signal and the noise. Although this assumption holds in the expected sense, as we have seen before, in any given frame it is only a rough approximation (Sect. 33.4):

$$|Y(f)|^2 \approx |X(f)|^2 + |N(f)|^2 \,. \qquad (33.63)$$

Using (33.63), the clean power spectrum can be estimated by subtracting an estimate of the noise power spectrum from the noisy power spectrum:

$$|\hat{X}(f)|^2 = |Y(f)|^2 - |\hat{N}(f)|^2 = |Y(f)|^2 H_{\text{ss}}^2(f)\,, \tag{33.64}$$

where the equivalent spectral subtraction filter in the power spectral domain is

$$H_{\text{ss}}^2(f) = 1 - \frac{1}{\text{SNR}(f)}\,, \tag{33.65}$$

and the frequency-dependent signal-to-noise ratio $\text{SNR}(f)$ is

$$\text{SNR}(f) = \frac{|Y(f)|^2}{|\hat{N}(f)|^2}\,. \tag{33.66}$$

The weakest point in many speech enhancement algorithms is the method for estimating the noise power spectrum. More advanced enhancement algorithms can be less sensitive to this estimate, but spectral subtraction is quite sensitive. The easiest option is to assume the noise is stationary, and obtain an estimate using the average periodogram over M frames that are known to be just noise

$$|\hat{N}(f)|^2 = \frac{1}{M}\sum_{i=0}^{M-1} |Y_i(f)|^2\,. \tag{33.67}$$

In practice, there is no guarantee that the spectral subtraction filter in (33.65) is nonnegative, which violates the fundamental nature of power spectra. In fact, it is easy to see that noise frames do not comply. To enforce this constraint, *Boll* [33.43] suggested modifying the filter as

$$H_{\text{ss}}^2(f) = \max\left(1 - \frac{1}{\text{SNR}(f)}, a\right) \tag{33.68}$$

with $a \geq 0$, so that the filter is always positive.

This implementation results in output speech that has significantly less noise, though it exhibits what is called *musical noise* [33.46]. This is caused by frequency bands f for which $|Y(f)|^2 \approx |\hat{N}(f)|^2$. As shown in Fig. 33.7, a frequency f_0 for which $|Y(f_0)|^2 < |\hat{N}(f_0)|^2$ is attenuated by A dB, whereas a neighboring frequency f_1, where $|Y(f_1)|^2 > |\hat{N}(f_1)|^2$, has a much smaller attenuation. These rapid changes of attenuation over frequency introduce tones at varying frequencies that appear and disappear rapidly.

The main reason for the presence of musical noise is that the estimates of $\text{SNR}(f)$ are poor. This is partly because the $\text{SNR}(f)$ are computed independently for every time and frequency, even though it would be more reasonable to assume that nearby values are correlated. One possibility is to smooth the filter in (33.68) over time, frequency, or both. This approach suppresses a smaller amount of noise, but it does not distort the signal as much, and thus may be preferred. An easy way to smooth over time is to mix a small portion of the previous SNR estimate into the current frame:

$$\text{SNR}(f,t) = \gamma\,\text{SNR}(f,t-1) + (1-\gamma)\frac{|Y(f)|^2}{|\hat{N}(f)|^2}\,. \tag{33.69}$$

This smoothing yields more-accurate SNR measurements and thus less distortion, at the expense of reduced noise suppression.

Other enhancements to the basic algorithm have been proposed to reduce the musical noise. Sometimes (33.68) is generalized to

$$f_{\text{ms}}(x) = \left[\max\left(1 - \frac{1}{x^{\frac{\alpha}{2}}}, a\right)\right]^{\frac{1}{\alpha}}\,, \tag{33.70}$$

where $\alpha = 2$ corresponds to the *power spectral subtraction rule* in (33.64), and $\alpha = 1$ corresponds to the *magnitude subtraction rule* (plotted in Fig. 33.7 for $A = 10\,\text{dB}$):

$$g_{\text{ms}}(\overline{x}) = \max\left[20\log_{10}\left(1 - 10^{-\frac{\overline{x}}{5}}\right), -A\right]\,. \tag{33.71}$$

Another variation, called oversubtraction, consists of multiplying the estimate of the noise power spectral density $|\hat{N}(f)|^2$ in (33.67) by a constant $10^{\frac{\beta}{10}}$, where $\beta > 0$, which causes the power spectral subtraction rule to be transformed to another function

$$g_{\text{ms}}(\overline{x}) = \max\left\{10\log_{10}\left[1 - 10^{-\frac{(\overline{x}-\beta)}{10}}\right], -A\right\}\,. \tag{33.72}$$

This causes $|Y(f)|^2 < |\hat{N}(f)|^2$ to occur more often than $|Y(f)|^2 > |\hat{N}(f)|^2$ for frames for which $|Y(f)|^2 \approx |\hat{N}(f)|^2$, and thus reduces the musical noise.

Since the normal cepstral processing for speech recognition includes finding the power spectrum for each frame of data, spectral subtraction can be performed directly in the feature extraction module, without the need for resynthesis. Instead of operating directly on the full-resolution power spectrum, a popular alternative is to use the mel-frequency power spectrum. This does not change the motivation or derivation for spectral subtraction, but the mel-frequency power spectrum is more stable and less susceptible to musical noise.

Because spectral techniques like spectral subtraction can be implemented with very little computational cost, they are popular in embedded applications. In particular, the advanced front-end (AFE) standard produced by the ETSI DSW working group [33.47, 48]

Table 33.10 ETSI advanced front-end word accuracy for Aurora 2. This low-resource front end produces respectable results on the task

Acoustic model	Test set A	Test set B	Test set C	Average
Clean	89.27	87.92	88.53	88.58
Multistyle	93.74	93.26	92.21	93.24

uses a variation of this technique in its noise reduction module. The complete AFE noise reduction process consists of a two-stage time-smoothed frequency-domain filtering [33.49], time-domain noise reduction [33.50], SNR-dependent waveform processing [33.51], and an online variant of CMN [33.52].

Table 33.10 presents the performance of the ETSI AFE on the Aurora 2 clean and multistyle tasks. Despite its low computational cost, the multistyle AFE has only 10% more errors than the best system described in this chapter (Table 33.15). It achieves a average word accuracy of 88.19% when trained with clean data, and 93.24% when trained with multistyle data.

33.6.2 Vector Taylor-Series Speech Enhancement

As shown in Sect. 33.5.3, the VTS approximation can be used to create a noisy speech model from a clean speech model. But, adapting each Gaussian component in a large speech model can be quite computationally expensive.

A lightweight alternative is to leverage a smaller model for speech into a separate enhancement step. The VTS approximation is applied to this smaller model, which is then used to estimate the enhanced features that are sent to an unmodified recognition system. A good comprehensive summary of this approach can be found in [33.45], and the literature is rich with implementations [33.38, 53].

Typically, the clean speech model is a multivariate Gaussian mixture model, and the noise model is a single Gaussian component. The parameters of this prior model include the state-conditional means and variances, $\mu_s^{\boldsymbol{x}}$, $\sigma_s^{\boldsymbol{x}}$, $\mu^{\boldsymbol{n}}$ and $\sigma^{\boldsymbol{n}}$, as well as the mixture component weights $p(s)$:

$$p(\boldsymbol{x}) = \sum_s N\left(\boldsymbol{x}; \mu_s^{\boldsymbol{x}}, \sigma_s^{\boldsymbol{x}}\right) p(s) \,, \tag{33.73}$$

$$p(\boldsymbol{n}) = N(\boldsymbol{n}; \mu^{\boldsymbol{n}}, \sigma^{\boldsymbol{n}}) \,. \tag{33.74}$$

These models are tied together by the nonlinear mixing represented by (33.36), recast as a probability distribution:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{n}) \approx N[\boldsymbol{y}; \boldsymbol{x} + \boldsymbol{g}(\boldsymbol{x} - \boldsymbol{n}), \Psi^2] \,. \tag{33.75}$$

Here, the variance Ψ^2 chiefly represents the error incurred in ignoring the cross-term produced by mixing the speech and noise (Sect. 33.4), and is in general a function of $(\boldsymbol{x} - \boldsymbol{n})$. For an overview of three reasonable approximations for Ψ^2 [33.38].

This Chapter uses the approximation $\Psi^2 = 0$, which is equivalent to ignoring the effects of the cross-term entirely. We call this approximation the zero-variance model (ZVM). This yields a good fit to the data at the extreme SNR regions, and a slight mismatch in the $\boldsymbol{x} \approx \boldsymbol{n}$ region.

If the VTS techniques from Sect. 33.5.3 were applied directly to the variables $\boldsymbol{x}$ and $\boldsymbol{n}$ for VTS enhancement, the result would be quite unstable [33.38]. Instead, the problem is reformulated in terms of $\boldsymbol{r}$, the instantaneous signal-to-noise ratio, defined as

$$\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{n} \,.$$

By performing VTS on the new variable $\boldsymbol{r}$, the stability problems are circumvented. In the end, an estimate for the instantaneous SNR can be mapped back into estimates of $\boldsymbol{x}$ and $\boldsymbol{n}$ through

$$\boldsymbol{x} = \boldsymbol{y} - \boldsymbol{g}(\boldsymbol{r}) \,, \tag{33.76}$$

$$\boldsymbol{n} = \boldsymbol{y} - \boldsymbol{g}(\boldsymbol{r}) - \boldsymbol{r} \,. \tag{33.77}$$

These formulas satisfy the intuition that as the SNR $\boldsymbol{r}$ becomes more positive, $\boldsymbol{x}$ approaches $\boldsymbol{y}$ from below. As the SNR $\boldsymbol{r}$ becomes more negative, $\boldsymbol{n}$ approaches $\boldsymbol{y}$ from below.

The joint PDF for the ZVM is a distribution over the clean speech $\boldsymbol{x}$, the noise $\boldsymbol{n}$, the observation $\boldsymbol{y}$, the SNR $\boldsymbol{r}$, and the speech state s:

$$p(\boldsymbol{y}, \boldsymbol{r}, \boldsymbol{x}, \boldsymbol{n}, s) = p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{n}) p(\boldsymbol{r}|\boldsymbol{x}, \boldsymbol{n}) p(\boldsymbol{x}, s) p(\boldsymbol{n}) \,.$$

The observation and SNR are both deterministic functions of $\boldsymbol{x}$ and $\boldsymbol{n}$. As a result, the conditional probabilities $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{n})$ and $p(\boldsymbol{r}|\boldsymbol{x}, \boldsymbol{n})$ can be represented by Dirac delta functions:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{n}) = \delta[\ln(\mathrm{e}^{\boldsymbol{x}} + \mathrm{e}^{\boldsymbol{n}}) - \boldsymbol{y}] \,, \tag{33.78}$$

$$p(\boldsymbol{r}|\boldsymbol{x}, \boldsymbol{n}) = \delta(\boldsymbol{x} - \boldsymbol{n} - \boldsymbol{r}) \,. \tag{33.79}$$

Table 33.11 Word accuracy for Aurora 2, using VTS speech enhancement and an acoustic model trained on clean data. Enhancement is performed on static cepstra, which are then used to compute the dynamic coefficients used in recognition. This enhancement technique does better than any of the normalization techniques

Acoustic model	Iterations	Set A	Set B	Set C	Average
Clean	0	88.59	88.06	86.92	88.04
Clean	1	89.67	89.05	87.37	88.96
Clean	2	89.92	89.39	87.28	89.18
Clean	3	89.99	89.48	87.05	89.20

Table 33.12 Word accuracy for Aurora 2, using VTS speech enhancement and an acoustic model trained on enhanced multistyle data. As in Table 33.11, the dynamic coefficients are calculated from enhanced static coefficients. In the end, the result is not as good as simple feature normalization alone on a multistyle acoustic model

Acoustic model	Iterations	Set A	Set B	Set C	Average
Multistyle	0	92.58	91.14	92.52	91.99
Multistyle	1	92.79	91.27	92.24	92.07
Multistyle	2	92.86	91.35	92.10	92.11
Multistyle	3	92.82	91.35	91.94	92.06

This allows us to marginalize the continuous variables $\boldsymbol{x}$ and $\boldsymbol{n}$, as in

$$\begin{aligned} p(\boldsymbol{y},\boldsymbol{r},s) &= \int \mathrm{d}\boldsymbol{x} \int \mathrm{d}\boldsymbol{n}\, p(\boldsymbol{y},\boldsymbol{r},\boldsymbol{x},\boldsymbol{n},s) \\ &= N[\boldsymbol{y}-\boldsymbol{g}(\boldsymbol{r});\mu_s^{\boldsymbol{x}},\sigma_s^{\boldsymbol{x}}]p(s) \\ &\quad \times N[\boldsymbol{y}-\boldsymbol{g}(\boldsymbol{r})-\boldsymbol{r};\mu^{\boldsymbol{n}},\sigma^{\boldsymbol{n}}]\,. \end{aligned} \tag{33.80}$$

After marginalization, the only remaining continuous hidden variable is $\boldsymbol{r}$, the instantaneous SNR. The behavior of this joint PDF is intuitive. At high SNR,

$$p(\boldsymbol{y},\boldsymbol{r},s) \approx N(\boldsymbol{y};\mu_s^{\boldsymbol{x}},\sigma_s^{\boldsymbol{x}})p(s)N(\boldsymbol{y}-\boldsymbol{r};\mu^{\boldsymbol{n}},\sigma^{\boldsymbol{n}})\,.$$

That is, the observation is modeled as clean speech, and the noise is at a level $\boldsymbol{r}$ units below the observation. The converse is true for low SNR:

$$p(\boldsymbol{y},\boldsymbol{r},s) \approx N(\boldsymbol{y}-\boldsymbol{r};\mu_s^{\boldsymbol{x}},\sigma_s^{\boldsymbol{x}})p(s)N(\boldsymbol{y};\mu^{\boldsymbol{n}},\sigma^{\boldsymbol{n}})\,.$$

To solve the MMSE estimation problem, the nonlinear function $\boldsymbol{g}(\boldsymbol{r})$ in (33.80) is replaced by its Taylor-series approximation, as in Sect. 33.5.3. For now, the expansion point is the expected a priori mean of $\boldsymbol{r}$:

$$\boldsymbol{r}_s^0 = E\{\boldsymbol{r}|s\} = E\{\boldsymbol{x}-\boldsymbol{n}|s\} = \mu_s^{\boldsymbol{x}}-\mu^{\boldsymbol{n}}\,.$$

Using (33.80) and the VTS approximation, it can be shown that $p(\boldsymbol{y}|s)$ has the same form derived in Sect. 33.5.3. Essentially, we perform VTS adaptation of the clean speech GMM to produce a GMM for noisy speech:

$$p(\boldsymbol{y}|s) = N(\boldsymbol{y};\mu_{\boldsymbol{y}|s},\sigma_{\boldsymbol{y}|s})\,, \tag{33.81}$$

$$\mu_{\boldsymbol{y}|s} = \mu_s^{\boldsymbol{x}} + \boldsymbol{g}\left(-\boldsymbol{r}_s^0\right) + \mathbf{G}_s\left(\mu_s^{\boldsymbol{x}}-\mu_s^{\boldsymbol{n}}-\boldsymbol{r}_s^0\right)\,, \tag{33.82}$$

$$\sigma_{\boldsymbol{y}|s} = \mathbf{G}_s\sigma_s^{\boldsymbol{x}}\mathbf{G}_s^{\mathrm{T}} + (\mathbf{I}-\mathbf{G}_s)\sigma^{\boldsymbol{n}}(\mathbf{I}-\mathbf{G}_s)^{\mathrm{T}}\,. \tag{33.83}$$

When using the recommended expansion point, (33.82) simplifies to

$$\mu_{\boldsymbol{y}|s} = \mu^{\boldsymbol{x}} + \boldsymbol{g}\left(\mu^{\boldsymbol{n}}-\mu_s^{\boldsymbol{x}}\right)\,.$$

It is also straightforward to derive the conditional posterior $p(\boldsymbol{r}|\boldsymbol{y},s)$, as in (33.84). As in the other iterative VTS algorithms, we can use the expected value $E[\boldsymbol{r}|\boldsymbol{y},s] = \mu_s^{\boldsymbol{r}}$ as a new expansion point for the vector Taylor-series parameters and iterate.

$$\begin{aligned} p(\boldsymbol{r}|\boldsymbol{y},s) &= N\left(\boldsymbol{r};\mu_s^{\boldsymbol{r}},\sigma_s^{\boldsymbol{r}}\right)\,, \\ \left(\sigma_s^{\boldsymbol{r}}\right)^{-1} &= (\mathbf{I}-\mathbf{G}_s)^{\mathrm{T}}\left(\sigma_s^{\boldsymbol{x}}\right)^{-1}(\mathbf{I}-\mathbf{G}_s) \\ &\quad + (\mathbf{G}_s)^{\mathrm{T}}(\sigma^{\boldsymbol{n}})^{-1}\mathbf{G}_s\,, \\ \mu_s^{\boldsymbol{r}} &= \mu_s^{\boldsymbol{x}} - \mu^{\boldsymbol{n}} + \sigma_s^{\boldsymbol{r}}\left[(\mathbf{G}_s-\mathbf{I})^{\mathrm{T}}\left(\sigma_s^{\boldsymbol{x}}\right)^{-1}\right. \\ &\quad \left. + (\mathbf{G}_s)^{\mathrm{T}}(\sigma^{\boldsymbol{n}})^{-1}\right](\boldsymbol{y}-\mu_{\boldsymbol{y}|s})\,. \end{aligned} \tag{33.84}$$

After convergence, we compute an estimate of x from the parameters of the approximate model:

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

$$E[\boldsymbol{x}|\boldsymbol{y},s] \approx \boldsymbol{y} - \ln(\mathrm{e}^{\mu_s^{\boldsymbol{r}}}+1) + \mu_s^{\boldsymbol{r}}\,.$$

Here, (33.76) has been used to map $E[\boldsymbol{r}|\boldsymbol{y},s] = \mu_s^{\boldsymbol{r}}$ to $E[\boldsymbol{x}|\boldsymbol{y},s]$. Since the transformation is nonlinear, our estimate for $\hat{\boldsymbol{x}}$ is not the optimal MMSE estimator.

Tables 33.11–33.14 evaluate accuracy on Aurora 2 for several different VTS speech enhancement configurations. For all experiments, the clean speech GMM consisted of 32 diagonal components trained on the clean Aurora 2 training data.

Table 33.13 Word accuracy for Aurora 2, using VTS speech enhancement and an acoustic model trained on clean data. Unlike Table 33.11, the static cepstral coefficients are enhanced and then concatenated with noisy dynamic coefficients. The end result is worse than computing the dynamic coefficients from enhanced static coefficients

Acoustic model	Iterations	Set A	Set B	Set C	Average
Clean	0	83.56	84.60	82.52	83.77
Clean	1	84.38	85.48	82.82	84.51
Clean	2	84.60	85.72	82.76	84.68
Clean	3	84.68	85.81	82.65	84.73

Table 33.14 Word accuracy for Aurora 2, using VTS speech enhancement and an acoustic model trained on enhanced multistyle data. Unlike Table 33.12, the static cepstral coefficients are enhanced and then concatenated with noisy dynamic coefficients. The end result is better than computing the dynamic coefficients from enhanced static coefficients

Acoustic model	Iterations	Set A	Set B	Set C	Average
Multistyle	0	93.64	93.05	93.05	93.29
Multistyle	1	93.76	93.14	92.81	93.32
Multistyle	2	93.72	93.19	92.79	93.32
Multistyle	3	93.57	92.95	92.70	93.15

Table 33.15 Word accuracy for Aurora 2, adding CMVN after the VTS speech enhancement of Table 33.14. The result is a very high recognition accuracy

Acoustic model	Normalization	Set A	Set B	Set C	Average
Multistyle	None	93.72	93.19	92.79	93.32
Multistyle	CMVN	94.09	93.46	94.01	93.83

Of course, using the multistyle training data produces better accuracy. Comparing Tables 33.11 and 33.12, it is apparent that multistyle data should be used whenever possible. Note that the result with the clean acoustic model is better than all of the feature normalization techniques explored in this chapter, and better than the VTS adaptation result of Table 33.9. Because VTS enhancement is fast enough to compute new parameters specific to each utterance, it is able to better adapt to the changing conditions within each noise type.

Although the speech recognition features consist of static and dynamic cepstra, the VTS enhancement is only defined on the static cepstra. As a result, there are two options for computing the dynamic cepstra. In Tables 33.11 and 33.12, the dynamic cepstra were computed from the enhanced static cepstra. In the corresponding Tables 33.13 and 33.14, the dynamic cepstra were computed from the noisy static cepstra. In the former case, the entire feature vector is affected by the enhancement algorithm, and in the latter, only the static cepstra are modified.

Using the noisy dynamic cepstra turns out to be better for the multistyle acoustic model, but worse for the clean acoustic model. Under the clean acoustic model, the benefit of the enhancement outweighs the distortion it introduces. However, the multistyle acoustic model is already able to learn and generalize the dynamic coefficients from noisy speech. Table 33.14 shows that the best strategy is to only enhance the static coefficients.

Finally, consider the effect of adding feature normalization to the system. Normalization occurs just after the full static and dynamic feature vector is created, and before it is used by the recognizer. Table 33.15 presents the final accuracy achieved by adding CMVN to the result of Table 33.14. Even though the accuracy of the multistyle model was already quite good, CMVN reduces the error rate by another 7.6% relative.

33.7 Unifying Model and Feature Techniques

The front- and back-end methods discussed so far can be mixed and matched to good performance. This section introduces two techniques that achieve better accuracy through a tighter in-

tegration of the front- and back-end robustness techniques.

33.7.1 Noise Adaptive Training

Section 33.2 discussed how the recognizer's HMMs can be adapted to a new acoustical environment. Section 33.6 dealt with cleaning the noisy feature without retraining the HMMs. It is logical to consider a combination of both, where the features are cleaned to remove noise and channel effects and then the HMMs are retrained to take into account that this processing stage is not perfect.

It was shown in [33.19] that a combination of feature enhancement and matched condition training can achieve a lower word error rate than feature enhancement alone. This paper demonstrated how, by introducing a variant of the enhancement algorithm from Sect. 33.3.4, very low error rates could be achieved.

These low error rates are hard to obtain in practice, because they assume preknowledge of the exact noise type and level, which in general is difficult to obtain. On the other hand, this technique can be effectively combined with the multistyle training discussed in Sect. 33.2.1.

33.7.2 Uncertainty Decoding and Missing Feature Techniques

Traditionally, the speech enhancement algorithms from Sect. 33.6 output an estimate of the clean speech to be used by the speech recognition system. However, the accuracy of the noise removal process can vary from frame to frame and from dimension to dimension in the feature stream.

Uncertainty decoding [33.54] is a technique where the feature enhancement algorithm associates a confidence with each value that it outputs. In frames with a high signal-to-noise ratio, the enhancement can be very accurate and would be associated with high confidence. Other frames, where some or all of the speech has been buried in noise, would have low confidence.

The technique is implemented at the heart of the speech recognition engine, where many Gaussian mixture components are evaluated. When recognizing uncorrupted speech cepstra, the purpose of these evaluations is to discover the probability of each clean observation vector, conditioned on the mixture index, $p_{\boldsymbol{x}|m}(\boldsymbol{x}|m)$, for each Gaussian component in the speech model used by the recognizer.

If the training and testing conditions do not match, as is the case in noise-corrupted speech recognition, one option is to ignore the imperfections of the noise removal, and evaluate $p_{\boldsymbol{x}|m}(\hat{\boldsymbol{x}}|m)$. This is the classic case of passing the output of the noise removal algorithm directly to the recognizer.

With the uncertainty decoding technique, the joint conditional PDF $p(\boldsymbol{x}, \boldsymbol{y}|m)$ is generated and marginalized over all possible unseen clean-speech cepstra:

$$p(\boldsymbol{y}|m) = \int_{-\infty}^{\infty} p(\boldsymbol{y}, \boldsymbol{x}|m)\,\mathrm{d}\boldsymbol{x}\,. \tag{33.85}$$

Under this framework, instead of just providing cleaned cepstra, the speech enhancement process also estimates the conditional distribution $p(\boldsymbol{y}|\boldsymbol{x}, m)$, as a function of $\boldsymbol{x}$. For ease of implementation, it is generally assumed that $p(\boldsymbol{y}|\boldsymbol{x})$ is independent of m:

$$p(\boldsymbol{y}|\boldsymbol{x}, m) \approx p(\boldsymbol{y}|\boldsymbol{x}) = \alpha N\big(\boldsymbol{x}; \hat{\boldsymbol{x}}(\boldsymbol{y}), \sigma^2_{\hat{\boldsymbol{x}}}(\boldsymbol{y})\big)\,, \tag{33.86}$$

where α is independent of $\boldsymbol{x}$, and therefore can be ignored by the decoder. Note that $\hat{\boldsymbol{x}}$ and $\sigma^2_{\hat{\boldsymbol{x}}}$ are always functions of $\boldsymbol{y}$; the cumbersome notation is dropped for the remainder of this discussion.

Finally, the probability for the observation $\boldsymbol{y}$, conditioned on each acoustic model Gaussian mixture component m, can be calculated:

$$\begin{aligned} p(\boldsymbol{y}|m) &= \int_{-\infty}^{\infty} p(\boldsymbol{y}|\boldsymbol{x}, m)p(\boldsymbol{x}|m)\,\mathrm{d}\boldsymbol{x} \\ &\propto \int_{-\infty}^{\infty} N\big(\hat{\boldsymbol{x}}; \boldsymbol{x}, \boldsymbol{\sigma}^2_{\hat{\boldsymbol{x}}}\big) N\big(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\sigma}^2_m\big)\,\mathrm{d}\boldsymbol{x} \\ &= N\big(\hat{\boldsymbol{x}}; \boldsymbol{\mu}_m, \boldsymbol{\sigma}^2_m + \boldsymbol{\sigma}^2_{\hat{\boldsymbol{x}}}\big)\,. \end{aligned} \tag{33.87}$$

This formula is evaluated for each Gaussian mixture component in the decoder, $p(\boldsymbol{x}|m) = N(\boldsymbol{x}, \boldsymbol{\mu}_m, \boldsymbol{\sigma}^2_m)$.

As can be observed in (33.87), the uncertainty output from the front end increases the variance of the Gaussian mixture component, producing an effective smoothing in cases where the front end is uncertain of the true value of the cleaned cepstra.

Two special cases exist for uncertainty decoding. In the absence of uncertainty information from the noise removal process, we can either assume that there is no uncertainty or that there is complete uncertainty.

If there were no uncertainty, then $\boldsymbol{\sigma}^2_{\hat{\boldsymbol{x}}} = 0$. The probability of the observation $\boldsymbol{y}$ for each acoustic model Gaussian mixture component m simplifies to:

$$p(\boldsymbol{y}|m) = p(\hat{\boldsymbol{x}}|m) = N\big(\hat{\boldsymbol{x}}; \boldsymbol{\mu}_m, \boldsymbol{\sigma}^2_m\big)\,. \tag{33.88}$$

This is the traditional method of passing features directly from the noise removal algorithm to the decoder.

If there were complete uncertainty of any of the cepstral coefficients, the corresponding $\sigma_{\hat{x}}^2$ would approach infinity. That coefficient would have no effect on the calculation of $p(\boldsymbol{y}|m)$. This is desirable behavior, under the assumption that the coefficient could not contribute to discrimination.

Both of these extreme cases are similar to the computations performed when using hard thresholds with missing-feature techniques [33.55]. There has been some success in incorporating heuristic soft thresholds with missing-feature techniques [33.56], but without the benefits of a rigorous probabilistic framework.

33.8 Conclusion

To prove a newly developed system, it can be tested on any one of a number of standard noise-robust speech recognition tasks. Because a great number of researchers are publishing systematic results on the same tasks, the relative value of complete solutions can be easily assessed.

When building a noise-robust speech recognition system, there exist several simple techniques that should be tried before more-complex strategies are invoked. These include the feature normalization techniques of Sect. 33.3, as well as the model retraining methods of Sect. 33.2.

If state-of-the-art performance is required with a small amount of adaptation data, then the structured techniques of Sects. 33.5 and 33.6 can be implemented. Structured model adaptation carries with it an expensive computational burden, whereas structured feature enhancement is more lightweight but less accurate.

Finally, good compromises between the accuracy of model adaptation and the speed of feature enhancement can be achieved through a tighter integration of the front and back end, as shown in Sect. 33.7.

References

33.1 H.G. Hirsch, D. Pearce: The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy conditions, ISCA ITRW ASR2000 "Automatic Speech Recognition: Challenges for the Next Millennium" (2000)

33.2 R.G. Leonard, G. Doddington: *Tidigits* (Linguistic Data Consortium, Philadelphia 1993)

33.3 D. Pierce, A. Gunawardana: Aurora 2.0 speech recognition in noise: Update 2. Complex backend definition for Aurora 2.0, http://icslp2002.colorado.edu/special_sessions/aurora (2002)

33.4 A. Moreno, B. Lindberg, C. Draxler, G. Richard, K. Choukri, J. Allen, S. Euler: SpeechDat-Car: A large speech database for automotive environments, Proc. 2nd Int. Conf. Language Resources and Evaluation (2000)

33.5 J. Garofalo, D. Graff, D. Paul, D. Pallett: *CSR-I (WSJ0) Complete* (Linguistic Data Consortium, Philadelphia 1993)

33.6 A. Varga, H.J.M. Steeneken, M. Tomlinson, D. Jones: The NOISEX-92 study on the effect of additive noise on automatic speech recognition. Tech. Rep. Defence Evaluation and Research Agency (DERA) (Speech Research Unit, Malvern 1992)

33.7 A. Schmidt-Nielsen: *Speech in Noisy Environments (SPINE) Evaluation Audio* (Linguistic Data Consortium, Philadelphia 2000)

33.8 R.P. Lippmann, E.A. Martin, D.P. Paul: Multi-style training for robust isolated-word speech recognition, Proc. IEEE ICASSP (1987) pp. 709–712

33.9 M. Matassoni, M. Omologo, D. Giuliani: Hands-free speech recognition using a filtered clean corpus and incremental HMM adaptation, Proc. IEEE ICASSP (2000) pp. 1407–1410

33.10 G. Saon, J.M. Huerta, E.-E. Jan: Robust digit recognition in noisy environments: The Aurora 2 system, Proc. Eurospeech **2001** (2001)

33.11 C.-P. Chen, K. Filali, J.A. Bilmes: Frontend post-processing and backend model enhancement on the Aurora 2.0/3.0 databases, Int. Conf. Spoken Language Process. (2002)

33.12 B.S. Atal: Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification, J. Acoust. Soc. Am. **55**(6), 1304–1312 (1974)

33.13 B.W. Gillespie, L.E. Atlas: Acoustic diversity for improved speech recognition in reverberant environments, Proc. IEEE ICASSP **I**, 557–560 (2002)

33.14 A. de la Torre, A.M. Peinado, J.C. Segura, J.L. Perez-Cordoba, M.C. Benítez, A.J. Rubio: Histogram equalization of speech representation for robust speech recognition, IEEE Trans. Speech Audio Process. **13**(3), 355–366 (2005)

33.15 M.G. Rahim, B.H. Juang: Signal bias removal by maximum likelihood estimation for robust telephone speech recognition, IEEE Trans. Speech Audio Process. **4**(1), 19–30 (1996)

33.16 A. Acero, X.D. Huang: Augmented cepstral normalization for robust speech recognition, Proc. IEEE Workshop on Automatic Speech Recognition (1995)

33.17 J. Ramírez, J.C. Segura, C. Benítez, L. García, A. Rubio: Statistical voice activity detection using a multiple observation likelihood ratio test, IEEE Signal Proc. Lett. **12**(10), 689–692 (2005)

33.18 H. Hermansky, N. Morgan: RASTA processing of speech, IEEE Trans. Speech Audio Process. **2**(4), 578–589 (1994)

33.19 L. Deng, A. Acero, M. Plumpe, X.D. Huang: Large-vocabulary speech recognition under adverse acoustic environments, Int. Conf. Spoken Language Process. (2000)

33.20 A. Acero: *Acoustical and Environmental Robustness in Automatic Speech Recognition* (Kluwer Academic, Boston 1993)

33.21 P. Moreno: *Speech Recognition in Noisy Environments*, Ph.D. Thesis (Carnegie Mellon University, Pittsburgh 1996)

33.22 A. Acero, R.M. Stern: Environmental robustness in automatic speech recognition, Proc. IEEE ICASSP (1990) pp. 849–852

33.23 J. Droppo, A. Acero: Maximum mutual information SPLICE transform for seen and unseen conditions, Proc. Interspeech Conf. (2005)

33.24 J. Wu, Q. Huo: An environment compensated minimum classification error training approach and its evaluation on Aurora 2 database, Proc. ICSLP **1**, 453–456 (2002)

33.25 D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, G. Zweig: fMPE: Discriminatively trained features for speech recognition, Proc. IEEE ICASSP (2005)

33.26 S. Tamura, A. Waibel: Noise reduction using connectionist models, Proc. IEEE ICASSP (1988) pp. 553–556

33.27 L. Neumeyer, M. Weintraub: Probabilistic optimum filtering for robust speech recognition, Proc. IEEE ICASSP **1**, 417–420 (1994)

33.28 A. Acero, R.M. Stern: Robust speech recognition by normalization of the acoustic space, Proc. IEEE ICASSP **2**, 893–896 (1991)

33.29 M.J. Gales: *Model Based Techniques for Noise Robust Speech Recognition*, Ph.D. Thesis (Cambridge University, Cambridge 1995)

33.30 E.A. Wan, R.V.D. Merwe, A.T. Nelson: Dual estimation and the unscented transformation. In: *Advances in Neural Information Processing Systems*, ed. by S.A. Solla, T.K. Leen, K.R. Muller (MIT Press, Cambridge 2000) pp. 666–672

33.31 P.J. Moreno, B. Raj, R.M. Stern: A vector taylor series approach for environment independent speech recognition, Proc. IEEE ICASSP (1996) pp. 733–736

33.32 B.J. Frey, L. Deng, A. Acero, T. Kristjansson: ALGONQUIN: Iterating Laplace's method to remove multiple types of acoustic distortion for robust speech recognition, Proc. Eurospeech (2001)

33.33 J. Droppo, A. Acero, L. Deng: A nonlinear observation model for removing noise from corrupted speech log mel-spectral energies, Proc. Int. Conf. Spoken Language Process. (2002)

33.34 C. Couvreur, H. Van Hamme: Model-based feature enhancement for noisy speech recognition, Proc. IEEE ICASSP **3**, 1719–1722 (2000)

33.35 J. Droppo, A. Acero: Noise robust speech recognition with a switching linear dynamic model, Proc. IEEE ICASSP (2004)

33.36 B. Raj, R. Singh, R. Stern: On tracking noise with linear dynamical system models, Proc. IEEE ICASSP **1**, 965–968 (2004)

33.37 H. Shimodaira, N. Sakai, M. Nakai, S. Sagayama: Jacobian joint adaptation to noise, channel and vocal tract length, Proc. IEEE ICASSP **1**, 197–200 (2002)

33.38 J. Droppo, L. Deng, A. Acero: A comparison of three non-linear observation models for noisy speech features, Proc. Eurospeech Conf. (2003)

33.39 R.A. Gopinath, M.J.F. Gales, P.S. Gopalakrishnan, S. Balakrishnan-Aiyer, M.A. Picheny: Robust speech recognition in noise – performance of the IBM continuous speech recognizer on the ARPA noise spoke task, Proc. ARPA Workshop on Spoken Language Systems Technology (1995) pp. 127–133

33.40 A.P. Varga, R.K. Moore: Hidden markov model decomposition of speech and noise, Proc. IEEE ICASSP (1990) pp. 845–848

33.41 A. Acero, L. Deng, T. Kristjansson, J. Zhang: HMM adaptation using vector taylor series for noisy speech recognition, Int. Conf. Spoken Language Processing (2000)

33.42 W. Ward: Modeling non-verbal sounds for speech recognition, Proc. Speech and Natural Language Workshop (1989) pp. 311–318

33.43 S.F. Boll: Suppression of acoustic noise in speech using spectral subtraction, IEEE T. Acoust. Speech **24**(April), 113–120 (1979)

33.44 Y. Ephraim, D. Malah: Speech enhancement using a minimum mean-square error log-spectral amplitude estimator, IEEE Trans. Acoust. Speech Signal Process., Vol. ASSP-33 (1985) pp. 443–445

33.45 V. Stouten: *Robust Automatic Speech Recognition in Time-varying Environments*, Ph.D. Thesis (K. U. Leuven, Leuven 2006)

33.46 M. Berouti, R. Schwartz, J. Makhoul: Enhancement of speech corrupted by acoustic noise, Proc. IEEE ICASSP (1979) pp. 208–211

33.47 ETSI ES 2002 050 Recommendation: Speech processing, transmission and quality aspects (STQ);

distributed speech recognition; advanced front-end feature extraction algorithm (2002)
33.48 D. Macho, L. Mauuary, B. Noê, Y.M. Cheng, D. Ealey, D. Jouvet, H. Kelleher, D. Pearce, F. Saadoun: Evaluation of a noise-robust DSR front-end on Aurora databases, Proc. ICSLP (2002) pp. 17–20
33.49 A. Agarwal, Y.M. Cheng: Two-stage mel-warped Wiener filter for robust speech recognition, Proc. ASRU (1999)
33.50 B. Noê, J. Sienel, D. Jouvet, L. Mauuary, L. Boves, J. de Veth, F. de Wet: Noise reduction for noise robust feature extraction for distributed speech recognition, Proc. Eurospeech (2001) pp. 201–204
33.51 D. Macho, Y.M. Cheng: SNR-Dependent waveform processing for improving the robustness of ASR front-end, Proc. IEEE ICASSP (2001) pp. 305–308
33.52 L. Mauuary: Blind equalization in the cepstral domain for robust telephone based speech recognition, Proc. EUSPICO **1**, 359–363 (1998)
33.53 M. Afify, O. Siohan: Sequential estimation with optimal forgetting for robust speech recognition, IEEE Trans. Speech Audio Process. **12**(1), 19–26 (2004)
33.54 J. Droppo, A. Acero, L. Deng: Uncertainty decoding with SPLICE for noise robust speech recognition, Proc. IEEE ICASSP (2002)
33.55 M. Cooke, P. Green, L. Josifovski, A. Vizinho: Robust automatic speech recognition with missing and unreliable acoustic data, Speech Commun. **34**(3), 267–285 (2001)
33.56 J.P. Barker, M. Cooke, P. Green: Robust ASR based on clean speech models: An evaluation of missing data techniques for connected digit recognition in noise, Proc. Eurospeech **2001**, 213–216 (2001)