

# ADAPTATION TO ENVIRONMENT AND SPEAKER USING MAXIMUM LIKELIHOOD NEURAL NETWORKS\*

DongSuk Yuk<sup>†‡</sup> James Flanagan<sup>†</sup> Mahesh Krishnamoorthy<sup>†</sup> Krishna Dayanidhi<sup>†</sup>

<sup>†</sup>CAIP Center, Rutgers University, Piscataway, NJ

<sup>‡</sup>Department of Computer Science, Rutgers University, New Brunswick, NJ  
yuk@caip.rutgers.edu

## ABSTRACT

When there is a mismatch between training and testing conditions, statistical speech recognition algorithms suffer from severe degradation in recognition accuracy. The mismatch could be due to the interference from acoustical environments where systems are actually used or from speakers themselves. In this paper, a neural network based transformation approach is studied to handle the data distribution mismatches between training and testing conditions. The conditional probability that comes from hidden Markov model (HMM) based recognizers is used for the objective function of a neural network. It maximizes the likelihood of the data from a testing environment, and allows global optimization of the network when used with HMM-based recognizers. The new objective function can be used to transform speech feature vectors, or the mean vectors and covariance matrices of a recognizer. The proposed algorithm is evaluated on a noisy distant-talking version of the Resource Management database.

## 1. INTRODUCTION

When there is convolutional and additive noise in a testing environment, the observed feature value,  $x^{(f)}$ , in a linear frequency domain is

$$x^{(f)} = a^{(f)} \times \bar{x}^{(f)} + b^{(f)}, \quad (1)$$

where  $a^{(f)}$  and  $b^{(f)}$  are some constants, and  $\bar{x}^{(f)}$  is the corresponding clean speech feature value, and  $f$  represents a frequency bin. This relation becomes non-linear in logarithm domain (e.g., cepstrum domain) as follow;

$$\log x^{(f)} = \log [a^{(f)} \times \bar{x}^{(f)} + b^{(f)}] \quad (2)$$

$$= \log a^{(f)} + \log \left[ \bar{x}^{(f)} + \frac{b^{(f)}}{a^{(f)}} \right] \quad (3)$$

$$\neq \log a^{(f)} + \log c^{(f)} \times \log \bar{x}^{(f)}. \quad (4)$$

The additive term inside the logarithm,  $b^{(f)}/a^{(f)}$ , is not equivalent to some multiplicative term,  $\log c^{(f)}$ , outside the logarithm. Since a neural network, especially multi-layer perceptron (MLP), is known to be able to approximate any function [4][5], it can be used to establish the non-linear relation between clean speech and degraded speech in cepstrum domain [11]. In this paper, a neural network is applied to the parameter adaptation of HMM-based recognizers. The conditional probability of adaptation data for a given recognizer is used for an objective function of a neural network [10][13]. Since the likelihood of the data is maximized by the neural network, it is called *maximum likelihood neural network* (MLNN). The advantages of neural network based transformation

methods are as follows. It avoids retraining of a speech recognizer, which is an expensive task in terms of training data collection and computation time. By using a neural network, the adaptation process requires a small amount of adaptation data. Without any knowledge about the distortion in an environment, it automatically learns the mapping function between training and testing environments from examples. It can handle ambient noise, reverberation, channel mismatches, and their combinations [9].

Neural networks have been used in conjunction with speech recognizers in various ways for robust speech recognition. Tamura and Waibel [8] are one of the first to use neural networks to reduce noise in noisy speech signals. Bengio *et al.* [1], Biem *et al.* [2], and Rahim *et al.* [6] used neural networks as front-end for HMM-based speech recognizers for feature transformation. Previously [10][13], the MLNN is used for mean vector transformation. In this paper, it is extended to variance transformation and unsupervised speaker adaptation. In Section 2, the motivation behind the MLNN is described. In Section 3 and 4, it is applied to the transformation of mean vectors and covariance matrices to best match the testing environment. The experimental results are analyzed in Section 6.

## 2. MAXIMUM LIKELIHOOD NEURAL NETWORK

The error criterion of neural networks is traditionally represented by a mean squared error (MSE) [7]. The feature transformation neural network [11], which transforms distorted feature vectors to the corresponding clean speech vectors, is trained to minimize the mean squared error  $E$ ,

$$E = \frac{1}{2} \sum_i (\bar{x}_i - x_i)^2, \quad (5)$$

which is the summation of the squared difference between the  $i$ -th dimension target value,  $\bar{x}_i$  (i.e., clean speech), and the corresponding network output,  $x_i$  (i.e., approximated clean speech). On the other hand, continuous speech recognition is accomplished by finding the word sequence,  $\bar{U}$ , which gives the highest probability for the given feature vector sequence  $\bar{X}$ ;

$$\bar{U} = \arg \max_{U \in U^*} P(U|\bar{X}) \quad (6)$$

$$= \arg \max_{U \in U^*} \frac{P(\bar{X}|U)P(U)}{P(\bar{X})} \quad (7)$$

$$= \arg \max_{U \in U^*} P(\bar{X}|U)P(U), \quad (8)$$

where  $P(\bar{X}|U)$  is the acoustic score of the vector sequence  $\bar{X}$  for the given word sequence  $U$ ,  $P(U)$  is the score for the word sequence, which is usually computed independently using a stochastic language model. The likelihood  $P(\bar{X}|U)$  is the only term in equation (8), which can be affected by a feature or model transformation. The acoustic score of a word sequence is the product of

\* This research was supported by DARPA contract DABT63-93-C-0037.

† Now with IBM T. J. Watson Research Center, Yorktown Heights, NY.

each individual word acoustic score. In continuous speech recognition, the likelihood of the best word sequence is usually approximated by the likelihood of the best state sequence using a *Viterbi* algorithm [12];

$$P(X|U) = \sum_{S \in S^*} P(X, S|U) \quad (9)$$

$$\approx \max_{S \in S^*} P(X, S|U) \quad (10)$$

$$= \max_{S \in S^*} P(X|S)P(S|U) \quad (11)$$

$$= \max_{S \in S^*} \prod_{s \in S} P(x|s)P(s|\dot{s}) \quad (12)$$

where  $S^*$  is all the possible state sequences for the given utterance  $U$ ,  $P(X, S|U)$  is the acoustic score of  $X$ , coming from the state sequence  $S$ , for the given utterance  $U$ . Equation (12) is a result of the first-order dependency of an HMM; i.e., a current score depends on only one previous state.  $P(s|\dot{s})$  is the state transition probability from the state  $\dot{s}$  to the state  $s$ , where  $\dot{s}$  is the predecessor of the state  $s$  in the state sequence  $S$ . Assuming that the state transition probabilities are not changing, the feature or model transformation affects only  $P(x|s)$ , which is the likelihood of the observation vector  $x$  given the state  $s$ .  $P(x|s)$  is usually represented using a mixture of Gaussian distributions with diagonal covariances;

$$P(x|s) = \sum_p g_{s,p} e^{-\frac{1}{2} \sum_i \frac{(x - m_{s,p,i})^2}{v_{s,p,i}}} \quad (13)$$

$$g_{s,p} = c_{s,p} \frac{1}{\sqrt{(2\pi)^n \prod_i v_{s,p,i}}} \quad (14)$$

where  $c_{s,p}$  is the  $p$ -th distribution weight, and  $m_{s,p,i}$  and  $v_{s,p,i}$  are the  $i$ -th dimension mean and variance, respectively, in the  $p$ -th distribution of state  $s$ . Minimizing the mean squared error in equation (5) does not necessarily means maximizing the likelihood in equation (13).

One way to increase the likelihood in equation (13) is to train a feature transformation neural network to maximize the likelihood [10][13]. The probability density function of equation (13) can be directly used as an objective function of the neural network. Using the modified version of the *error back-propagation* (EBP) algorithm [7], the weights of the network,  $w$ , are updated at each training iteration to increase the conditional probability of each state;

$$w \leftarrow w + \Delta w \quad (15)$$

$$\Delta w = \eta \frac{\partial P(X|U)}{\partial w} \quad (16)$$

$$\frac{\partial P(X|U)}{\partial w} \propto \sum_{s \in \bar{S}} \frac{\partial P(x|s)P(s|\dot{s})}{\partial w} \quad (17)$$

This type of network may suffer from trainability problem [9]; i.e. it may not learn an extremely complex inverse function using only a limited amount of training data. In this paper, the conditional probability is used to transform the parameters of a speech recognizer instead of speech feature vectors. The HMM-base speech recognizers that use mixture Gaussian distributions have three types of parameters; state transition probabilities (including weights of the PDF's in a mixture distribution), mean vectors, and covariance matrices. In the next sections, the MLNN is applied to mean and variance transformations.

### 3. MEAN TRANSFORMATION MLNN

A model transformation may be an easier function than a feature transformation because it does not invert distorted speech back to

clean speech. Instead, it transforms clean speech mean vectors to the corresponding distorted speech mean vectors to approximate matched training and testing conditions. The conditional probability,  $P(x|s)$ , can be used for the mean transformation as well as the feature transformation. For mean transformation, the observation  $x_i$  is fixed, and the mean  $m_{s,p,i}$  becomes a variable (i.e., input and output of the neural network). For an output layer's weight update rule, the logarithm of equation (13) is differentiated with respect to the output neuron's weight  $w_{i,j}$ ;

$$\frac{\partial \ln P(x|s)}{\partial w_{i,j}} = \sum_p \frac{\partial \ln P(x|s)}{\partial m_{s,p,i}} \frac{\partial m_{s,p,i}}{\partial \sigma_{p,i}} \frac{\partial \sigma_{p,i}}{\partial w_{i,j}} \quad (18)$$

where  $\sigma_{p,i}$  is an input to the sigmoid function of the  $i$ -th neuron for the  $p$ -th PDF. Note that  $\sigma$  is dependent on  $p$  because each mean vector of the state  $s$  is provided to the network sequentially, producing a different  $\sigma$  value for each PDF. The weight update rule for a hidden layer can be derived by first expressing the likelihood  $P(x|s)$  in terms of hidden layer's weight,  $w_{j,k}$ , then differentiating it with respect to the weight;

$$\frac{\partial \ln P(x|s)}{\partial w_{j,k}} = \sum_p \frac{\partial \ln P(x|s)}{\partial h_{p,j}} \frac{\partial h_{p,j}}{\partial \sigma_{p,j}} \frac{\partial \sigma_{p,j}}{\partial w_{j,k}} \quad (19)$$

$$= \sum_p \sum_i \left[ \frac{\partial \ln P(x|s)}{\partial m_{s,p,i}} \frac{\partial m_{s,p,i}}{\partial \sigma_{p,i}} \frac{\partial \sigma_{p,i}}{\partial h_{p,j}} \right] \times \frac{\partial h_{p,j}}{\partial \sigma_{p,j}} \frac{\partial \sigma_{p,j}}{\partial w_{j,k}} \quad (20)$$

where  $h_{p,j}$  is the output of  $j$ -th hidden node for the  $p$ -th PDF. Note that  $h_{p,j}$  is a function of hidden layer's weights. Equation (18) and (20) are plugged into equation (17) to complete the weight update rule for mean transformation MLNN. The first term of equation (18) and (20) can be rewritten as follows for a diagonal covariance matrix case;

$$\frac{\partial \ln P(x|s)}{\partial m_{s,p,i}} = \frac{P(x|s,p)}{P(x|s)} \frac{m_{s,p,i} - x_i}{v_{s,p,i}} \quad (21)$$

The amount of the weight change is then proportional to how important the PDF is, i.e.,  $P(x|s,p)/P(x|s)$ , at current iteration, and how far the current mean is from the observation, i.e.,  $(m_{s,p,i} - x_i)/v_{s,p,i}$ , in *Mahalanobis* distance space.

It can be considered that there is a separate network for each PDF, and their weights are shared among them. This is mathematically equivalent to feeding the mean vector of each PDF sequentially and update the weight according to the accumulated weight change. The mean transformation MLNN can be used where the inverse function may not be physically realizable or where the network can not be well-trained with a limited amount of data.

### 4. VARIANCE TRANSFORMATION MLNN

The conditional probability,  $P(x|s)$ , can also be applied to a variance transformation. Following the same procedure as in the mean transformation case, the weight update rule can be derived for the output layer's weights as follows;

$$\frac{\partial \ln P(x|s)}{\partial w_{i,j}} = \sum_p \frac{\partial \ln P(x|s)}{\partial v_{s,p,i}} \frac{\partial v_{s,p,i}}{\partial \sigma_{p,i}} \frac{\partial \sigma_{p,i}}{\partial w_{i,j}} \quad (22)$$

and for the hidden layer's weights;

$$\frac{\partial \ln P(x|s)}{\partial w_{j,k}} = \sum_p \frac{\partial \ln P(x|s)}{\partial h_{p,j}} \frac{\partial h_{p,j}}{\partial \sigma_{p,j}} \frac{\partial \sigma_{p,j}}{\partial w_{j,k}} \quad (23)$$

$$= \sum_p \sum_i \left[ \frac{\partial \ln P(x|s)}{\partial v_{s,p,i}} \frac{\partial v_{s,p,i}}{\partial \sigma_{p,i}} \frac{\partial \sigma_{p,i}}{\partial h_{p,j}} \right]$$

$$\times \frac{\partial h_{p,j}}{\partial \sigma_{p,j}} \frac{\partial \sigma_{p,j}}{\partial w_{j,k}}, \quad (24)$$

where the first term of equation (22) and (24) becomes as follows for the diagonal covariance case;

$$\frac{\partial \ln P(x|s)}{\partial v_{s,p,i}} = \frac{P(x|s,p) ((x_i - m_{s,p,i})^2 - v_{s,p,i})}{P(x|s) v_{s,p,i}^2}, \quad (25)$$

Note that the variable is now  $v_{s,p,i}$ . Equation (22) and (24) are substituted into equation (17) for the weight update rule of variance transformation MLNN. Even though one neural network can be used to transform both mean and variance, separate neural network is used for each transformation in this paper. The variance transformation is done after mean vectors are transformed.

## 5. HYBRID NEURAL NETWORKS

When model parameters are adapted to match the current adverse environment, it suffers from a lower performance upper bound than feature adaptation approaches. When only a limited amount of data is available, the feature transformation MLNN fails because it may not learn complex inverse function from a small amount of examples [9]. In order to overcome this difficulty as well as the anomaly of mean squared error criterion described in Section 2, a hybrid approach is proposed, which is the combination of the feature transformation neural network that uses stereophonically collected data and the model transformation MLNN. Figure 1 shows the combination of these two types of neural networks. The stereo data used

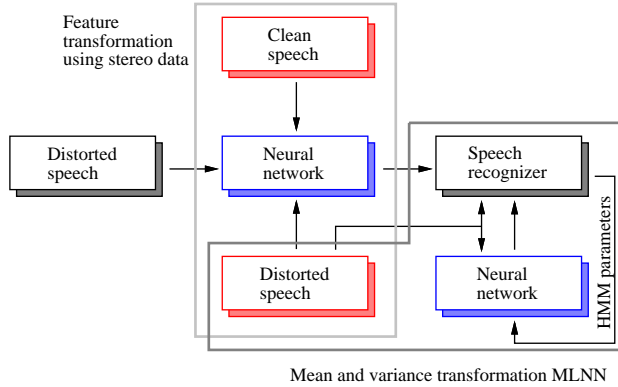


Figure 1: Combination of feature transformation and model transformation neural networks.

to train the feature transformation network can also be used to compute the Viterbi path for the model transformation MLNN.

## 6. EXPERIMENTAL RESULTS

The proposed algorithms have been evaluated on a noisy distant-talking version of Resource Management database [9]. The SNR of this database is about 20dB, and the distance from a microphone and a speaker is about 5.4–5.8 meters. Figure 2 shows the empirical and transformed distributions of a sound “g”. The distributions are plotted using a middle state of a triphone model of which base phone is “g”. Only the first dimension of *mel-frequency cepstral coefficients* (MFCC) is shown in the figure. Figure 2 (a) is a clean speech distribution. Figure 2 (b) is for the same sound after the cepstral mean normalization (CMN). It can be seen that the mean of the distribution is shifted. Figure 2 (c) and (d) are the transformed distributions by the mean transformation MLNN and the variance transformation MLNN, respectively. The variance transformation

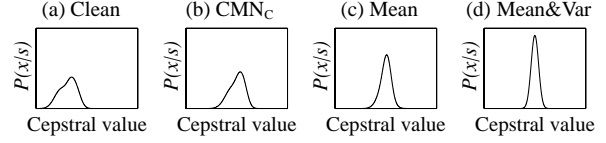


Figure 2: Empirical & transformed distribution of clean speech sound “g”.

MLNN is applied after the mean transformation MLNN. In this example, the variance becomes smaller, producing sharper distribution for the sound “g”. Figure 3 is a noisy distant-talking speech distribution of the same sound. Figure 3 (a) is before the CMN and

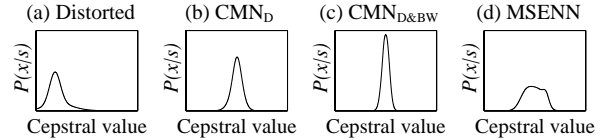


Figure 3: Distributions of noisy distant-talking sound “g”.

Figure 3 (b) is after the CMN. Figure 3 (c) is the result of several iterations of the Baum-Welch reestimation algorithm starting from Figure 3 (b). It can be observed in these figures that the original clean speech distribution, Figure 2 (b), is successfully converted to Figure 2 (d) which looks more similar to the noisy distant-talking speech distribution, Figure 3 (c).

Figure 4 shows the example of transformed distributions by the hybrid neural networks. The distorted speech, Figure 3 (c), is

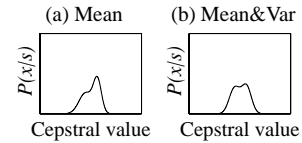


Figure 4: Transformed distributions by the hybrid neural networks.

transformed to an intermediate distribution (Figure 3 (d)) by the feature transformation network that uses stereo data (the first network in Figure 1). This intermediate distribution is to approximate the clean speech distribution, Figure 3 (a). Then, the statistics of a speech recognizer is transformed to match this distribution. Figure 4 (a) is the result of mean transformation, and Figure 4 (b) is the result of mean and variance transformation. It can be seen that the final output of the hybrid model transformation, Figure 4 (b), and the output of feature transformation, Figure 3 (d), are quite matched. These distributions are more similar to the original clean speech distribution, Figure 2 (a), than Figure 2 (d) or Figure 3 (c).

Table 1 compares the word recognition accuracy of various adaptation approaches. Except in the retraining case, a clean speech recognizer is used for testing, and 5 minutes of data is used for adaptation. The feature transformation neural network using stereo data and MSE as its objective function boosts the performance from 23.7% to 64.3%. The feature transformation MLNN (45.8%) does not work better than the ones that make use of stereo data. However, the mean transformation MLNN (63.7%) is quite comparable. The variance transformation on top of the mean transformation further improves the recognition accuracy to 77.4%. The tandem use of MSE network and MLNN improves the word recognition accuracy to 73.0% for mean transformation, and to 78.7%

condition	accuracy (%)
clean (matched)	94.5
distorted (matched)	79.0
distorted (mismatched)	8.1
CMN	23.7
MLLR	62.8
MSENN	64.3
MLNN <sub>F</sub>	45.8
MLNN <sub>M</sub>	63.7
MLNN <sub>M&amp;V</sub>	77.4
MSENN+MLLR	71.9
MSENN+MLNN <sub>F</sub>	50.1
MSENN+MLNN <sub>M</sub>	73.0
MSENN+MLNN <sub>M&amp;V</sub>	78.7
MSENN+MLNN <sub>M&amp;V+US</sub>	83.2

Table 1: Word recognition accuracies (%) of various adaptation schemes. “clean” is for clean speech training and testing. “distorted (matched)” is for noisy speech training and testing. “distorted (mismatched)” is for clean speech training and noisy speech testing. The rest are using clean speech recognizer and some adaptation scheme. “CMN” is for cepstral mean normalization. All the adaptation schemes are applied after the CMN. “MLLR” is for maximum likelihood linear regression. “MLNN<sub>F</sub>”, “MLNN<sub>M</sub>”, “MLNN<sub>M&V</sub>”, are feature transformation, mean transformation, and mean and variance transformation MLNN, respectively. The rest are the combination of feature transformation neural network using stereo data and some adaptation scheme.

for mean and variance transformation. This result is quite competitive with a retrained recognizer (79.0%). However, note that the retrained recognizer uses about 40 times more data. Using the hypotheses of 78.7% accuracy, one mean transformation MLNN is built for each speaker in unsupervised way. The unsupervised speaker adaptation improves the performance (83.2%) beyond the retrained recognizer.

## 7. CONCLUSIONS AND FUTURE WORK

In this paper, the anomaly of traditional mean squared error criterion for the objective function of neural networks in speech recognition is analyzed. The conditional probability density function of feature vector, given an HMM state, is more consistent with HMM-based recognizer, and allows global optimization mechanism in synergetic use of neural network and HMM for robust speech recognition. The feature transformation that uses stereo data can learn complex inverse transformation function, while the feature transformation MLNN may not in practice. The mean and variance transformation MLNN learn the distortion function that degrades clean speech statistics to distorted speech. It has been found that the MSE network using stereo data and the model transformation MLNN is complementary, and the tandem use of the networks improves the performance further. The training can be done in supervised way or unsupervised fashion by making use of recognized output for unknown speech. The proposed algorithm has been applied to large vocabulary continuous speech recognition under adverse acoustical environments, which involves background noise, reverberation, and difference in microphones. The model transformation MLNN experiment done in this research uses only one network to transform all mean vectors (or covariance matrices) of a recognizer. This can be modified to be state-dependent, where each state has its own MLNN to transform its parameters. The states can be grouped using tree structure so that those states that do not have enough training data can share an MLNN. The MLNN is a good candidate for discriminative training, because alternative hypotheses or confusing pairs can be provided from a speech rec-

ognizer. In this case, *mutual information* or *maximum a posteriori probability* criteria may be considered for the objective functions. Current unsupervised adaptation methods use the output of a mismatched recognizer. When the adaptation algorithms run iteratively, the recognizer tends to make same mistakes repeatedly. More intelligent techniques to exploit the multiple hypotheses from possibly multiple recognizers need to be investigated. Incorporating boosting algorithm [3] can be one research direction.

## 8. REFERENCES

- [1] Y. Bengio, R. DeMori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transactions on Neural Networks*, 3(2):252–259, March 1992.
- [2] A. Biem and S. Katagiri. Feature extraction based on minimum classification error/generalized probabilistic descent method. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2:275–278, April 1993.
- [3] Y. Freund and R. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [4] K. Hornik, M. Stinchcombe, and H. White. Multilayer feed-forward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
- [5] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.
- [6] M. Rahim and C. Lee. Simultaneous ANN feature and HMM recognizer design using string-based minimum classification error (MCE) training. *International Conference on Spoken Language Processing*, 3:1824–1827, October 1996.
- [7] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In J. McClelland D. Rumelhart, editor, *Parallel Distributed Processing: Exploration in the Micro-Structure of Cognition*, volume 1, pages 318–362. MIT Press, 1986.
- [8] S. Tamura and A. Waibel. Noise reduction using connectionist models. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:553–556, April 1988.
- [9] D. Yuk. *Robust Speech Recognition Using Neural Networks and Hidden Markov Models*. PhD thesis, Rutgers University, 1999.
- [10] D. Yuk, C. Che, and J. Flanagan. Robust speech recognition using maximum likelihood neural networks and continuous density hidden Markov models. *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 474–481, December 1997.
- [11] D. Yuk, C. Che, L. Jin, and Q. Lin. Environment-independent continuous speech recognition using neural networks and hidden Markov models. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 6:3358–3361, May 1996.
- [12] D. Yuk, C. Che, P. Raghavan, S. Chennoukh, and J. Flanagan. N-best breadth search for large vocabulary continuous speech recognition using a long span language model. *136th meeting of Acoustical Society of America*, October 1998.
- [13] D. Yuk and J. Flanagan. Telephone speech recognition using neural networks and hidden Markov models. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:157–160, March 1999.