# ICA IN SPEECH RECOGNITION USING HMM's

Joel Pinto, R. Muralishankar, A.G Ramakrishnan
Department of Electrical Engineering, Indian Institute of Science, Bangalore, India
joel@ragashri.ee.iisc.ernet.in, sripad@ee.iisc.ernet.in, ramkiag@ee.iisc.ernet.in

## Abstract

*Hidden Markov models (HMM) are widely used in speech recognition these days. For all practical purposes, viz, mathematical tractability and reducing computational complexity, the observation density function in a continuous HMM is assumed to be mixture gaussian with diagonal covariance matrices. The feature transformation plays an important role in achieving this decorrelation. Different feature transformations like discrete cosine transform (DCT) and principal component analysis (PCA) have been tried. DCT is a suboptimal transform that nearly diagonalizes the covariance matrix and retains higher order correlations. PCA on the other hand diagonalizes the covariance matrix but again retains the higher order dependencies. Independent component analysis (ICA), used widely in blind source separation problems can be used as a feature transform. ICA makes the feature vectors independent and thereby fully decorrelated. Experiments are conducted on a 50 word vocabulary in a multi-speaker mode. The results are compared with the baseline system where DCT is used as a feature transform. A 2% increase in recognition accuracy is observed using ICA as a feature transformation over DCT.*

## 1 Introduction

The idea behind Independent Component Analysis (ICA) is to reduce the redundancy in the original feature vector components (log spectral energies). The data model for linear ICA is [4]

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

where $\mathbf{x}$ is the observed feature vector, $\mathbf{s}$ is the underlying independent sources and $\mathbf{A}$ is the mixing matrix. The ICA algorithm estimates a demixing matrix $\mathbf{W}$ such that components of $\mathbf{y}$ are independent.

$$\mathbf{y} = \mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{A}\mathbf{s} \tag{2}$$

When $\mathbf{WA} = \mathbf{I}$ we have achieved perfect independence. Generally unsupervised methods of implementing ICA framework can ensure independence fulfilling the relaxed condition

$$\mathbf{WA} = \mathbf{PD} \tag{3}$$

where $\mathbf{P}$ is the permutation matrix and $\mathbf{D}$ is the diagonal scaling matrix.

## 2 Baseline System

In the baseline system, features are extracted using the mel frequency cepstral coefficients (MFCC) [8]. The speech signal, sampled at 8000 Hz, is windowed using a Hamming window of 20 msec duration, with an overlap of 10 msec. This is Fourier transformed using a 256-point FFT and filtered using a 20 channel mel filter bank and energy in each filter is then calculated. These log spectral energies are then decorrelated using DCT and 13 MFCC coefficients are retained.

Hidden Markov model pattern classifier is used for recognition. The observation density in each state is assumed to be a mixture gaussian with diagonal covariance matrices.

$$b_j(\mathbf{O}) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\mathbf{O};\mu_{\mathbf{jm}},\mathbf{U_{jm}}) \quad 1 \le j \le N \tag{4}$$

where $N$ is the number of states, $M$ the number of mixtures, $c_{jm}$ the mixture weight and $\mathcal{N}(\mathbf{O};\mu_{jm},\mathbf{U}_{jm})$ gaussian distribution with mean $\mu_{jm}$ and covariance matrix $\mathbf{U}_{jm} = diag\{\sigma^2_{jm1}...\sigma^2_{jmD}\}$. $D$ is the dimensionality of the feature. Training is done using segmental K-means algorithm. A 6 state, 5 mixture left to right HMM without skips is used.

The system is trained using segmental K-means algorithm which has the following steps: model initialization, state sequence segmentation using Viterbi algorithm and model re-estimation. A transition probability matrix is initialized randomly. In the first iteration, observations are segmented equally into different states. In the subsequent iterations, the segmentation is by applying the Viterbi algorithm [6][5][7]. All the observations falling into one state are clustered into M mixtures using K-means clustering algorithm. The weight, mean and variances of each cluster are estimated using the following equations:

$$c_m = \tfrac{K_m}{K} \quad m = 1, 2, ....M \tag{5}$$

$$\mu_m = \frac{1}{K_m}\sum_{k=1}^{K_m}\mathbf{X}_{mk} \;\; m = 1,2,...M \tag{6}$$

$$\mathbf{U}_m = \frac{1}{K_m - 1}\sum_{k=1}^{K_m}(\mathbf{X}_{mk} - \mu_m)(\mathbf{X}_{mk} - \mu_m)^T \tag{7}$$

$$m = 1,2,...M$$

where $M$ is the number of mixtures, $K_m$ is the number of samples in the $m^{th}$ cluster and $K$ is the total number of samples in a state. Eqn. 7 gives us the estimate of the covariance matrix. Because we have assumed a diagonal covariance matrix, the cross variances are made zero. The transition matrix is updated using the re-estimation equation

$$a_{ij} = \frac{total\;\; transitions\;\; from\;\; state\;\; i\;\; to\;\; state\;\; j}{total\;\; transitions\;\; from\;\; state\;\; i} \tag{8}$$

During K-segmental training procedure, a word model is generated for each of the $W$ words. These word models are represented by $\lambda_1, \lambda_2, ...\lambda_W$. When a new test pattern is encountered, the likelihood of its observation, given each of the word model is computed. The test pattern is assigned that class, $W^*$ whose likelihood is maximum.

$$W^* = \arg\max_W [P(\mathbf{O} \mid \lambda_W)] \tag{9}$$

## 3 Independent component Analysis

Modeling the output probability density function as a Gaussian mixture with diagonal covariance matrix is equivalent to the assumption that the feature vectors are statistically uncorrelated [3]. Independent Component Analysis (ICA), like PCA, is an unsupervised, linear, data dependent transformation. However, unlike PCA which removes only second order correlation, ICA removes all higher order statistical dependencies making the feature vector truly uncorrelated.

Let $\mathbf{s} = [s_1, s_2, ...s_N]^T$ be a $N$-dimensional zero-mean vector, with the components $s_i$ being mutually independent. Its multivariate p.d.f. can be written as [4][2]

$$f_{\mathbf{s}}(s_1, ....s_N) = \prod_{i=1}^{N} f_i(s_i) \tag{10}$$

The data vector $\mathbf{x} = [x_1, x_2, ...x_N]^T$ is observed, such that

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{11}$$

where $\mathbf{A}$ is a full rank $NXN$ scalar matrix. The components of the observed vectors are no longer independent and multivariate p.d.f. will not satisfy the p.d.f. product equality as in eqn. 10. The mutual information of the observed vector is given by Kullback-Leibler divergence of the multivariate density from the product of the univariate densities [4].

$$I(\mathbf{x}) = \int f(\mathbf{x}) \ln \frac{f(\mathbf{x})}{\prod_{i=1}^{N} f_i(x_i)} d\mathbf{x} \tag{12}$$

The mutual information is always positive and becomes zero when the components are independent. The goal of ICA is to find a linear transformation $W$ of the mixed signal $\mathbf{x}$ that makes the outputs as independent as possible

$$\mathbf{y} = \mathbf{W}\mathbf{x} = \mathbf{W}\mathbf{A}\mathbf{s} \tag{13}$$

where $\mathbf{y}$ is the estimate of the sources. The sources are exactly recovered when $\mathbf{W}$ is the inverse of $\mathbf{A}$ up to a permutation and scale change. The fundamental restriction in ICA is that the independent components must be nongaussian. The classical measure of nongaussianity are kurtosis and negentropy. The kurtosis of $Y$ is defined by

$$kurt(Y) = E\{Y^4\} - 3(E\{Y^2\})^2 \tag{14}$$

Even though kurtosis can be easily estimated, it is not an robust measure of nongaussianity because when kurtosis is estimated from a measured sample, its value may depend on the few samples in the tails of the distribution.

In the FastICA algorithm, negentropy is used as the measure of nongaussianity. Negentropy is based on the information-theoretic quantity of differential entropy. Negentropy $J$ is defined as

$$J(\mathbf{Y}) = H(\mathbf{Y}_g) - H(\mathbf{Y}) \tag{15}$$

where $\mathbf{Y}$ is a continuous random vector and $\mathbf{Y}_g$ is a Gaussian random vector of the same covariance matrix. $H(Y)$ is the differential entropy of $\mathbf{Y}$ and is defined as

$$H(\mathbf{Y}) = -\int f(\mathbf{y}) \log f(\mathbf{y}) d\mathbf{y} \tag{16}$$

where $f(\mathbf{y})$ is the probability density function of the random variable $\mathbf{Y}$. Gaussian variable has the largest entropy among all random variables of equal variance. Hence negentropy is always positive and zero for a gaussian random vector. Estimating negentropy using eqn. 15 would require an estimate of the pdf and therefore, simpler approximations are used. The most general approximation of negentropy is given by

$$J(\mathbf{Y}) \approx \sum_{i=1}^{p} k_i [E\{G_i(\mathbf{Y})\} - E\{G_i(\mathbf{v})\}]^2 \tag{17}$$

where $k_i$ are some positive constants, and $\mathbf{v}$ is a Gaussian random variable of zero mean and unit variance, and the functions $G_i$ are some nonquadratic functions. In the case where we use only one nonquadratic function $G$, eqn. 17 reduces to

$$J(\mathbf{Y}) \propto [E\{G(\mathbf{Y})\} - E\{G(\mathbf{v})\}]^2 \tag{18}$$

Taking $\mathbf{G}(y) = y^4$, we obtain a kurtosis based approximation, which is not very robust. By choosing $G$ that does not grow very fast, we obtain more robust estimators [1]. The following nonquadratic functions have been used in our experiments:

$$G_1(z) = \frac{1}{a} \log \cosh az \tag{19}$$

$$G_2(z) = -\exp(-z^2/2) \tag{20}$$

where $a$ is a suitable constant such that $1 \leq a \leq 2$.

## 4 Fast ICA Algorithm

The FastICA uses negentropy as the measure of nongaussianity. The data must be centered and whitened in order to simplify the algorithm [2][5]. Centering is by subtracting the mean $\mathbf{M} = E\{\mathbf{X}\}$ so as to make $\mathbf{X}$ a zero-mean variable. After estimating the mixing matrix $\mathbf{A}$ using the centered data, we add mean vector of $\mathbf{S}$ i.e, $\mathbf{A}^{-1}\mathbf{M}$ back to the centered estimates of $\mathbf{S}$.

The centered data is whitened using the Eigen Value Decomposition of the covariance matrix. We use the transformation $\widetilde{\mathbf{X}} = \widetilde{\mathbf{A}}\mathbf{X}$ such that $E\{\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T\} = \mathbf{I}$ where

$$\widetilde{\mathbf{A}} = \mathbf{E}\mathbf{D}^{-0.5}\mathbf{E}^T\mathbf{A} \tag{21}$$

where $\mathbf{E}$ is the orthogonal matrix of the eigenvectors of $E\{\mathbf{X}\mathbf{X}^T\}$ and $\mathbf{D}$ is diagonal matrix of its eigenvalues, $\mathbf{D} = diag\{d_1, d_2 ... d_D\}$. Whitening reduces the number of parameters to be estimated from $D^2$ to $D(D-1)/2$, where $D$ is the dimensionality of the mixing matrix $A$.

### FastICA for one unit

Here ”unit” refers to an artificial neuron, having a weight $\mathbf{w}$ that it is able to update by a learning rule. The FastICA learning rule finds a direction, i.e, a unit vector such that the projection $\mathbf{w}^T\mathbf{x}$ maximizes nongaussianity. We have used negentropy $J(\mathbf{w}^T\mathbf{x})$ given by eqn 18 as a measure of nongaussianity. The variance of $\mathbf{w}^T\mathbf{x}$ is constrained to be unity. For whitened data, this is equivalent to constraining the norm of $\mathbf{w}$ to be unity.

The FastICA is based on a fixed point iteration scheme for finding the maximum of nongaussianity of $\mathbf{w}^T\mathbf{x}$, as in eqn 18. Let $g$ denote the derivative of the nonquadratic function $G$ used in eqn 20 and $1 \leq a \leq 2$ is a suitable constant. Thus

$$g_1(z) = \tanh(az) \tag{22}$$

$$g_2(z) = z \exp(-z^2/2) \tag{23}$$

The basic form of FastICA algorithm is as follows:
Let $\mathbf{w}^{(n)}$ denote the weight vector after $n$ iterations and $g'$ denote the first derivative of $g$.

1. Set $n = 1$
   Initialize weight vector $\mathbf{w}^{(1)}$ randomly
2. $\mathbf{w}^{(n+1)} = E\{\mathbf{x}g(\mathbf{w}^{T(n)}\mathbf{x})\} - E\{g'(\mathbf{w}^{T(n)}\mathbf{x})\}\mathbf{w}^{(n)}$
3. $\mathbf{w}^{(n+1)} = \mathbf{w}^{(n+1)} / \left\|\mathbf{w}^{(n+1)}\right\|$
4. If $\mathbf{w}^{T(n)}\mathbf{w}^{(n+1)}$ is close to unity, then stop.
   Else $n = n + 1$; Go to step 2.

### FastICA for several units

The one-unit algorithm estimates only one of the independent components. To estimate several independent components, we need to run one unit FastICA algorithm using several units (neurons) with weights $\mathbf{w}_1, \mathbf{w}_2, ... \mathbf{w}_N$.

Deflation technique is used to decorrelate the outputs $\mathbf{w}_1^T\mathbf{x}, \mathbf{w}_2^T\mathbf{x} ... \mathbf{w}_N^T\mathbf{x}$ after every iteration to prevent different vectors from converging to the same maxima. The deflation scheme is based on Gram-Schmidt decorrelation technique. The independent components are calculated one by one. When $p$ independent components are calculated, the one-unit fixed point algorithm is run for $\mathbf{w}_{p+1}$ and after every iteration, subtract from $\mathbf{w}_{p+1}$ the projections $\mathbf{w}_{p+1}^T\mathbf{w}_j\mathbf{w}_j$, $j = 1, 2, ... p$ of the previously estimated $p$ vectors, $\mathbf{w}_{p+1}$ is normalized

$$\mathbf{w}_{p+1} = \mathbf{w}_{p+1} - \sum_{i=1}^{p} \mathbf{w}_{p+1}^T\mathbf{w}_i\mathbf{w}_i \tag{24}$$

$$\mathbf{w}_{p+1} = \mathbf{w}_{p+1} / \sqrt{\mathbf{w}_{p+1}^T\mathbf{w}_{p+1}} \tag{25}$$

## 5 Experimental results and discussion

The proposed technique was tested in a multi-speaker, limited vocabulary, recognition scenario. It was plugged to the speech recognition unit of a voice controlled ultrasound scanner used for diagnostic purposes. The vocabulary consisted of 50 phrases, most of them containing two or three words. Fig. 1 gives the plot of the duration of the phrases and its frequency of occurrence. The database for the experiment was recorded from 5 speakers. Each speaker has uttered a word 8 times. Speech was recorded at a sampling frequency of 8 kHz and stored at 16 bits per sample. 5 repetitions of each word from a speaker were used for training and the rest 3 for testing. The FastICA package was downloaded from the internet [9] and used for this work.

Figs. 2 and 3 show the improvement in the recognition accuracy using ICA over DCT as feature transformation in

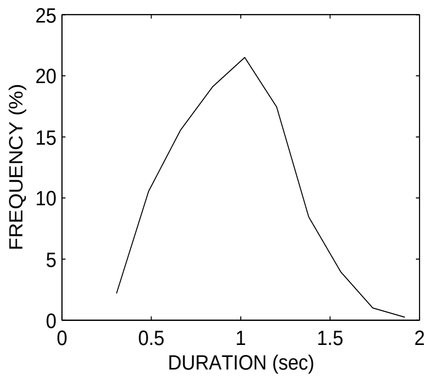

Figure 1: Plot of the relative frequency of the duration of each phrase in the vocabulary

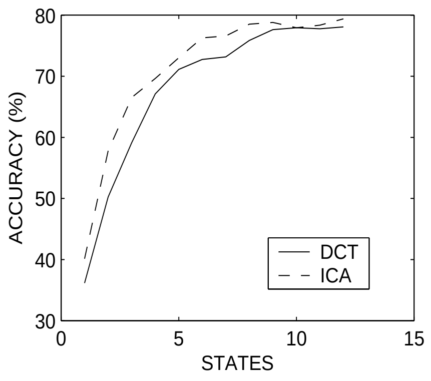

Figure 2: Plot of recognition accuracy verses states for ICA and DCT as feature transformation (No. of training samples/speaker = 3)

MFCC. The graphs have been plotted for different number of states. DCT as a feature transform gave a maximum accuracy of 89% while ICA gave 91% accuracy. The improvement in accuracy is more pronounced in Fig. 2 where only three training samples per word per speaker has been used.

## 6 Conclusions

In this work we have used a used a simple pattern classifier wherein we have forced the same number of states for each of the words. The improvement in the recognition accuracy is because of the sophisticated feature extraction. ICA transforms the features such that it removes all higher order dependencies. This satisfies the assumptions made in the continuous HMM based recognition framework. This is confirmed by the increase in the recognition accuracy. The improvement is achieved at the expense of higher computational complexity during training. However, there is no increase in the computational complexity during testing. The

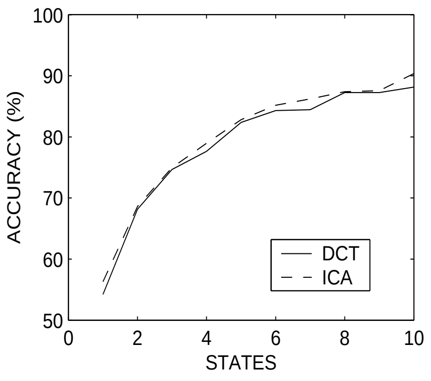

Figure 3: Plot of recognition accuracy verses states for ICA and DCT as feature transformation (No. of training samples/speaker = 5).

performance can be improved further by using noise model in the ICA framework.

## References

[1] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley and Sons, 2001.

[2] A. Hyvarinen and E. Oja. Independent Component Analysis : A Tutorial. `http://www.cis.hut.fi/projects/ica/`, 1999.

[3] J. H. Lee, H. Y. Jung, T. W. Lee, and S. Y. Lee. Speech feature extraction using independent component analysis. In *Proceedings of ICASSP*, volume 3, pages 1631–1634, 2000.

[4] T. W. Lee. *Independent Component Analysis, Theory and Application*. Kluwer Academic Publishers, Boston, September 1998.

[5] J. Pinto. ICA in Speech Recognition using HMM's. Master's thesis, Indian Institute of Science, Bangalore, India, January 2003.

[6] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, volume 77(2), pages 257–286, February 1989.

[7] L. R. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Eaglewood Cliffs, NJ, 1993.

[8] R. Vergin, D. O'Shaughnessy, and A. Farhat. Generalised Mel Frequency Cepstral Coefficients for Large Vocabulary Speaker Independent Continuous Speech Recognition. *IEEE Transactions Speech and Audio Processing*, 7(5):525–532, September 1999.

[9] The FastICA package for MATLAB. `http://www.cis.hut.fi/projects/ica/fastica`.