ICA Mixture Models for Unsupervised Classification of Non-Gaussian Classes and Automatic Context Switching in Blind Signal Separation

Te-Won Lee, *Member*, *IEEE*, Michael S. Lewicki, and Terrence J. Sejnowski, *Fellow*, *IEEE*

Abstract—An unsupervised classification algorithm is derived by modeling observed data as a mixture of several mutually exclusive classes that are each described by linear combinations of independent, non-Gaussian densities. The algorithm estimates the density of each class and is able to model class distributions with non-Gaussian structure. The new algorithm can improve classification accuracy compared with standard Gaussian mixture models. When applied to blind source separation in nonstationary environments, the method can switch automatically between classes, which correspond to contexts with different mixing properties. The algorithm can learn efficient codes for images containing both natural scenes and text. This method shows promise for modeling non-Gaussian structure in high-dimensional data and has many potential applications.

Index Terms—Unsupervised classification, Gaussian mixture model, independent component analysis, blind source separation, image coding, automatic context switching, maximum likelihood.

1 Introduction

Recently, Blind Source Separation (BSS) by Independent Component Analysis (ICA) has been applied to signal processing problems including speech enhancement, telecommunications, and medical signal processing. ICA finds a linear nonorthogonal coordinate system in multivariate data determined by second- and higher-order statistics. The goal of ICA is to linearly transform the data such that the transformed variables are as statistically independent from each other as possible [20], [11], [5], [10], [21]. ICA generalizes the technique of Principal Component Analysis (PCA) and, like PCA, has proven to be a useful tool for finding structure in data.

One limitation of ICA is the assumption that the sources are independent. Here, we present an approach for relaxing this assumption using mixture models. In a mixture model (see, for example, [12]), the observed data can be categorized into several mutually exclusive classes. When the data in each class are modeled as multivariate Gaussian, it is called a Gaussian mixture model. We generalize this by assuming the data in each class are generated by a linear combination of independent, non-Gaussian sources, as in the case of ICA. We call this model an ICA mixture model. This allows modeling of classes with non-Gaussian structure, e.g., platykurtic or leptokurtic probability density functions. The algorithm for learning[1] the parameters of the model uses gradient ascent to maximize the log-likelihood function. In previous applications, this approach showed improved performance in data classification problems [24] and learning efficient codes for representing different types of images [25].

This paper derives learning rules for the ICA mixture model and demonstrates that it can accurately classify unlabeled data using both synthetic and real data sets. Blind source separation is also shown in nonstationary environments. This is particularly useful when abrupt changes occur and fast adaptation to new environments is required. The ICA mixture model can also be used to find efficient codes to represent different image types. An example is given for how the codes can be used as features for unsupervised image classification and image compression. Finally, we discuss limitations of this approach and present future research directions.

2 The ICA Mixture Model

Assume that the data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ are drawn independently and generated by a mixture density model [12]. The likelihood of the data is given by the joint density

- *T.-W. Lee is with the Howard Hughes Medical Institute, Computational Neurobiology Laboratory, The Salk Institute, La Jolla, California 92037, and the Institute for Neural Computation, University of California, San Diego, La Jolla, CA 92093. E-mail: tewon@salk.edu.*
- *M.S. Lewicki is with the Department of Computer Science and Center for the Neural Basis of Cognition, Carnegie Mellon University, 4400 Fifth Ave., Pittsburgh, PA 15213. E-mail: lewicki@cnbc.cmu.edu.*
- *T.J. Sejnowski is with the Howard Hughes Medical Institute, Computational Neurobiology Laboratory, The Salk Institute, La Jolla, CA 92037, the Department of Biology, and the Institute for Neural Computation, University of California, San Diego, La Jolla, CA 92093. Email: terry@salk.edu.*

Manuscript received 15 Mar. 1999; revised 6 Apr. 2000; accepted 6 July 2000. Recommended for acceptance by C.A. Bouman.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 109423.

1. Note that an algorithm for learning the parameters is a terminology often used in artificial neural networks that refers to a method for estimating the parameters by iteratively updating the parameters. This terminology is used throughout the paper and is interchangeable with *parameter estimation* or *gradient update rules.*

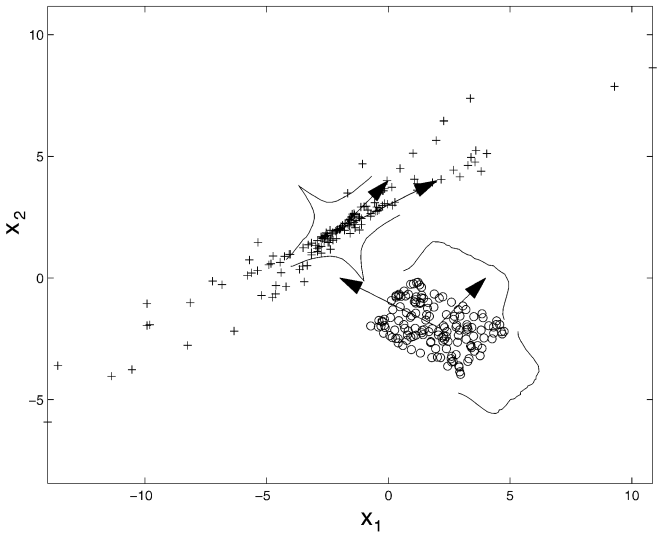

Fig. 1. A simple example for classifying an ICA mixture model. There are two classes, "+" and "o"; each class was generated by two independent variables, two bias terms, and two basis vectors. Class "o" was generated by two uniformly distributed sources as indicated next to the data class. Class "+" was generated by two Laplacian distributed sources with a sharp peak at the bias and heavy tails. The inset graphs show the distributions of the source variables, $\mathbf{s}_{i,k}$, for each basis vector.

$$p(\mathbf{X}|\Theta) = \prod_{t=1}^{T} p(\mathbf{x}_t|\Theta). \tag{1}$$

The mixture density is

$$p(\mathbf{x}_t|\Theta) = \sum_{k=1}^{K} p(\mathbf{x}_t|C_k, \theta_k)p(C_k), \tag{2}$$

where $\Theta = (\theta_1, \ldots, \theta_K)$ are the unknown parameters for each $p(\mathbf{x}|C_k, \theta_k)$, called the component densities. C_k denotes the class k and it is assumed that the number of classes, K, is known in advance. Assume that the component densities are non-Gaussian and the data within each class are described by:

$$\mathbf{x}_t = \mathbf{A}_k\mathbf{s}_k + \mathbf{b}_k, \tag{3}$$

where $\mathbf{A}_k$ is a $N \times M$ scalar matrix[2] and $\mathbf{b}_k$ is the bias vector for class k. The vector $\mathbf{s}_k$ is called the source vector[3] (these are also the coefficients for each basis function).

It is assumed that the individual sources $s_{k,i}$ within each class are mutually independent across a data ensemble. For simplicity, we consider the case where the number of sources (M) is equal to the number of linear combinations (N). Fig. 1 shows a simple example of a dataset describable by an ICA mixture model. Each class was generated from (3) using a different $\mathbf{A}_k$ and $\mathbf{b}_k$. Class "o" was generated by two uniformly distributed sources, whereas class "+" was generated by two Laplacian distributed sources ($p(s) \propto \exp(-|s|)$). The task is to classify the unlabeled data points and to determine the parameters for each class, $(\mathbf{A}_k, \mathbf{b}_k)$ and the probability of each class $p(C_k|\mathbf{x}_t, \Theta)$ for each data point.

The iterative learning algorithm (derived in Appendix A) which performs gradient ascent on the total likelihood of the data in (2) has the following steps:

- Compute the log-likelihood of the data for each class:

$$\log p(\mathbf{x}_t|C_k, \theta_k) = \log p(\mathbf{s}_k) - \log(\det|\mathbf{A}_k|), \tag{4}$$

  where $\theta_k = \{\mathbf{A}_k, \mathbf{b}_k\}$. Note that $\mathbf{s}_k = \mathbf{A}_k^{-1}(\mathbf{x}_t - \mathbf{b}_k)$ is implicitly modeled for the adaptation of $\mathbf{A}_k$.
- Compute the probability for each class given the data vector $\mathbf{x}_t$:

$$p(C_k|\mathbf{x}_t, \Theta) = \frac{p(\mathbf{x}_t|\theta_k, C_k)p(C_k)}{\sum_k p(\mathbf{x}_t|\theta_k, C_k)p(C_k)}. \tag{5}$$

- Adapt the basis functions $\mathbf{A}_k$ and the bias terms $\mathbf{b}_k$ for each class. The basis functions are adapted using gradient ascent:

$$\begin{aligned}\Delta\mathbf{A}_k &\propto \nabla_{\mathbf{A}_k} \log p(\mathbf{x}_t|\Theta) \\ &= p(C_k|\mathbf{x}_t, \Theta)\nabla_{\mathbf{A}_k} \log p(\mathbf{x}_t|C_k, \theta_k).\end{aligned} \tag{6}$$

  This gradient can be approximated using an ICA algorithm, as shown below. The gradient can also be summed over multiple data points. An approximate update rule was used for the bias terms (see the Appendix for an online update version for $\mathbf{b}_k$ and the derivations):

$$\mathbf{b}_k = \frac{\sum_t \mathbf{x}_t p(C_k|\mathbf{x}_t, \Theta)}{\sum_t p(C_k|\mathbf{x}_t, \Theta)}, \tag{7}$$

  where t is the data index ($t = 1, \ldots, T$).

The gradient of the log of the component density in (6) can be modeled using an ICA model. There are several methods for adapting the basis functions in the ICA model [11], [10], [6], [19], [22]. One of the differences between the various ICA algorithms are the use of higher-order statistics such as cumulants versus models that use a predefined density model. In our model, we are interested in iteratively adapting the class parameters and modeling a wider range of distributions. The extended infomax ICA learning rule is able to blindly separate unknown sources with sub- and super-Gaussian distributions.[4] This is achieved by using a simple type of learning rule first derived by Girolami [17]. The learning rule in [22] uses the stability analysis of [10] to switch between sub- and super-Gaussian regimes

$$\Delta\mathbf{A}_k \propto -p(C_k|\mathbf{x}_t, \Theta)\mathbf{A}_k\left[\mathbf{I} - \mathbf{K}\tanh(\mathbf{s}_k)\mathbf{s}_k^T - \mathbf{s}_k\mathbf{s}_k^T\right], \tag{8}$$

where k_i are elements of the N-dimensional diagonal matrix $\mathbf{K}$ and $\mathbf{s}_k = \mathbf{A}_k^{-1}(\mathbf{x}_t - \mathbf{b}_k)$. $\mathbf{W}_k = \mathbf{A}_k^{-1}$ is called the filter matrix. The adaptation of the source density parameters are the $k_{k,i}$s [22]

$$k_{k,i} = \text{sign}\Big(E\{\text{sech}^2(s_{k,i,t})\}E\{s_{k,i,t}^2\} - E\{[\tanh(s_{k,i,t})]s_{k,i,t}\}\Big). \tag{9}$$

2. This matrix is called the *mixing matrix* in ICA papers and specifies the linear combination of independent sources. Here, we refer to $\mathbf{A}$ as the basis matrix to distinguish this from the word *mixture* in the mixture model.

3. Note that we have omitted the data index t for $\mathbf{s}_{k,t}$.

4. A distribution that is more sharply peaked than a Gaussian around the mean and has heavier tails is called super-Gaussians (leptokurtic distributions) and a distribution with a flatter peak such as a uniform distribution is called sub-Gaussian (platykurtic distribution).

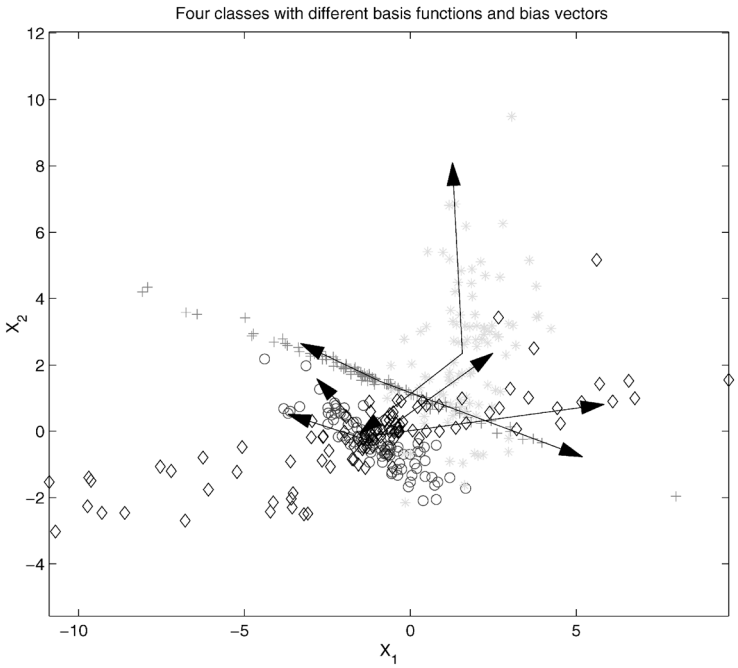

Fig. 2. An example of classification of a mixture of non-Gaussian densities, each describable as a linear combination of independent components. There are four different classes, each generated by two randomly chosen independent variables and bias terms. The algorithm is able to find the independent directions (basis vectors) and bias terms for each class.

The source distribution is super-Gaussian when $k_{k,i} = 1$ and sub-Gaussian when $k_{k,i} = -1$. For the log-likelihood estimation in (4), the term $\log p(\mathbf{s}_k)$ can be modeled as follows:

$$\log p(\mathbf{s}_{k,t}) \propto -\sum_{i=1}^{N}\left(k_{k,i}\log(\cosh s_{k,i,t}) - \frac{s_{k,i,t}^2}{2}\right). \qquad (10)$$

Super-Gaussian densities are approximated by a density model with heavier tail than the Gaussian density; sub-Gaussian densities are modeled by a bimodal density [17]. This source density approximation is adequate for most problems [22].[5] The extended infomax algorithm can be used for finding the parameters in Fig. 1. A continuous parameter is inferred that fits a wide range of distributions.

When only sparse representations are needed, a Laplacian prior ($p(s) \propto \exp(-|s|)$) or Laplacian source density can be used for the weight update, which simplifies the infomax learning rule:

$$\begin{aligned} \Delta\mathbf{A}_k &\propto p(C_k|\mathbf{x}_t,\Theta)\mathbf{A}_k\left[\mathbf{I} - \operatorname{sign}(\mathbf{s}_k)\mathbf{s}_k^T\right], \\ \log p(\mathbf{s}_k) &\propto -\sum_i |s_{k,i}| \qquad \text{Laplacian prior.} \end{aligned} \qquad (11)$$

A complete derivation of the learning algorithm is in Appendix B.

3 Unsupervised Classification

To demonstrate the performance of the learning algorithm, we generated random data drawn from different classes and used the proposed method to learn the parameters and to classify the data. Fig. 2 shows an example of four classes in a two-dimensional data space. Each class was generated from (3) using random choices for the class parameters. The task for the algorithm was to learn the four basis matrices and bias vectors given only the unlabeled two-dimensional data set. The parameters were randomly initialized. The algorithm always converged after 300 to 500 iterations depending on the initial conditions. During the adaptation process, the data log-likelihood was measured as a function of the number of iterations as shown in Fig. 3. The arrows in Fig. 2 are the basis matrices $\mathbf{A}_k$ and the bias vectors $\mathbf{b}_k$ found by the algorithm and these parameters matched the parameters which were used to generate the data for each class. The likelihood function usually increases monotonically, but, here, we were annealing the learning rate or stepsize during the adaptation process and that caused the little fluctuations. The annealing of the stepsize results in faster convergence.

The classification was tested by processing each instance with the learned parameters $\mathbf{A}_k$ and $\mathbf{b}_k$. The probability of the class $p(C_k|\mathbf{x}_t,\theta_k)$ was computed and the corresponding instance label was compared to the highest class probability. For this example, in which the classes had several overlapping areas, the algorithm was run 10 times with random initial conditions, in which it converged nine times to the correct solution and became stuck in a local maximum one time. The classification error on the whole data set averaged over nine trials was 17.8 percent ± 1 percent. The Gaussian mixture model used in AutoClass [37] gave an error of 20.8 percent ± 0.8 percent and converged in all 10 trials. For the k-means (Euclidean distance measure) clustering algorithm, the error was 36.3 percent. The classification error with the original parameters was 15.5 percent.

5. Recently, we have replaced this with a more general density using an exponential power distribution [23].

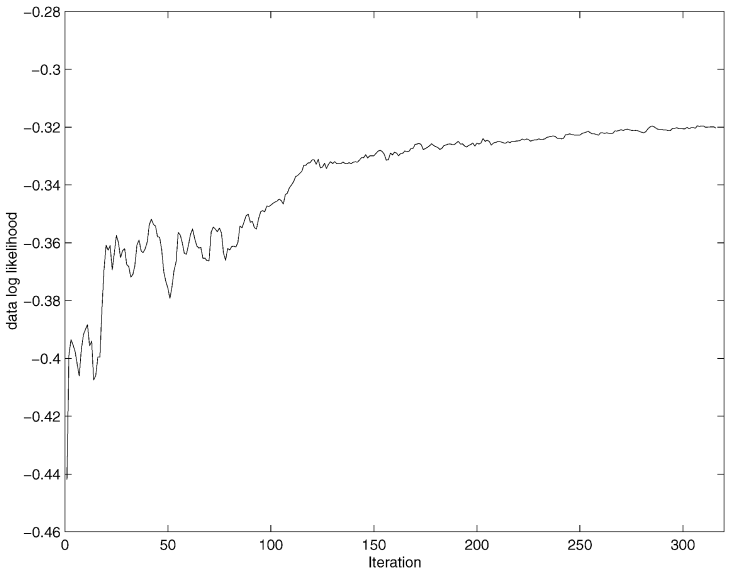

Fig. 3. The data log-likelihood as a function of the number of iterations. With increasing number of iterations, the adapted model parameters fit the observed data and the log-likelihood is maximized.

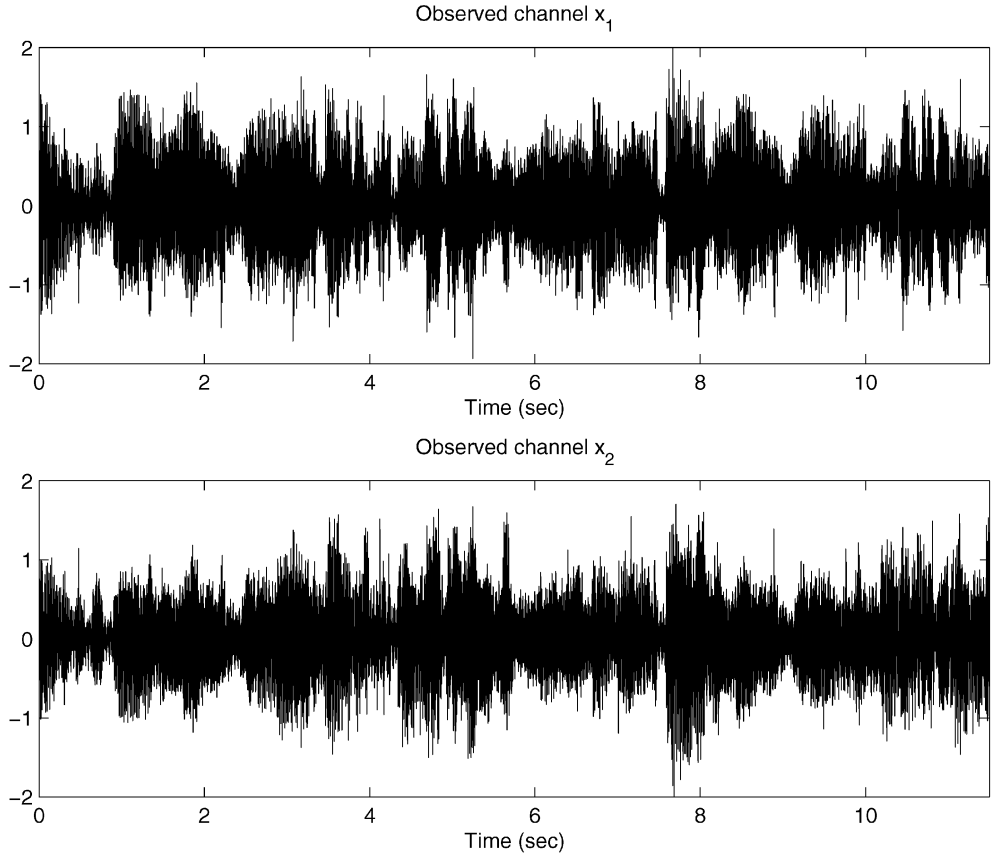

Fig. 4. The two observed channels x_1 and x_2 sampled at 8 kHz. Each channel contains the voices of person number 1 and number 2 and the music signal.

3.1 Iris Data Classification

The proposed method was compared to other algorithms on the classification of real data from the machine learning benchmark presented in [31]. The example given here is the well-known iris flower data set [13], which contains three classes with four numeric attributes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two, but the other two are not linearly separable from each other. Note that, from the viewpoint of the algorithm, the data are unlabeled and learning is unsupervised. We applied the ICA mixture model with the extended infomax ICA algorithm. The algorithm converged after one hundred passes through the data with a classification error of 3.3 percent ± 1.3 percent

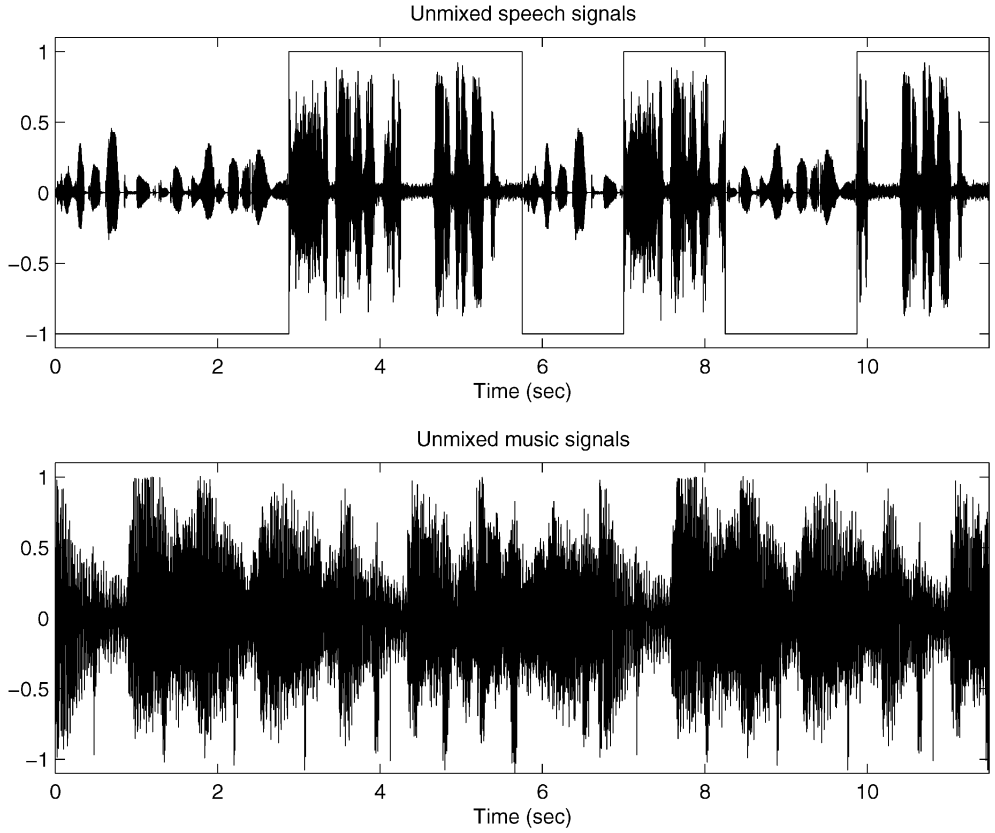

Fig. 5. The time course of the recovered signals using the mixture model and a block size of 2,000 samples for estimating the class probability. (Top) The two speech signals with correct labels (rectangular envelope) indicating which speaker was talking. (Bottom) The time course of the background music signal.

compared with 4.7 percent ± 1.3 percent error using AutoClass and 4.7 percent error using k-means clustering.

4 Context Switching in Blind Source Separation

The ICA mixture model can be used to automatically identify different contexts in blind source separation problems. Imagine there are two people talking to each other while they are listening to music in the background. Two microphones are placed somewhere in the room to record the conversation. The conversation alternates so that person number 1 talks while person number 2 listens, then person number 1 listens to person number 2 and so on. The basis matrix changes as a function of the location of the speaker. In this case, the voice of person number 1 overlaps with the background music signal with $\mathbf{A}_1$, while the voice of person number 2 overlaps with the music signal with $\mathbf{A}_2$. Fig. 4 shows the two observed channels x_1 and x_2. Each channel contains the voices of person number 1 and number 2 and the music signal. Although there are three different source signals, at any given moment there are only two in the observed data.

The algorithm was adapted on 11 seconds sampled at 8 kHz to learn two classes of ICA representations. The two basis vectors $\mathbf{A}_1$ and $\mathbf{A}_2$ were randomly initialized. For each gradient in (6), a stepsize was computed as a function of the amplitude of the basis vectors and the number of iterations.

The time course of the recovered signals using the ICA mixture model is shown in Fig. 5. The top plot shows the two speech signals with correct labels indicating which speaker was talking. The bottom plot shows the time course of the background music signal.

Fig. 6 (top) shows the class conditional probability, $p(C_2|\mathbf{x}_t, \theta_2) = 1 - p(C_1|\mathbf{x}_t, \theta_1)$, for each sample (data vector) in the series. Note that a single sample typically does not contain enough information to unambiguously assign class membership. The intermediate values for the class probability represent uncertainty about the class membership. A threshold at $p(C_2|\mathbf{x}_t, \theta_2) = 0.5$ can be used to determine the class membership. Using this threshold for single samples in Fig. 6 (top) gave an error rate of 27.4 percent. This can be improved using the a priori knowledge that a given context persists over many samples. This information could be incorporated into a more complex temporal model for $p(C_k)$, but, here, we use the crude but simple procedure of computing the class membership probability for an n-sample block. This value is plotted for a block size of 100 samples in Fig. 6 (middle). The value provides a more accurate estimate of class membership (6.5 percent error). The error rate dropped to zero when the block size was increased to 2,000 samples (Fig. 6 (bottom)); the correct class probabilities were recovered and matched those in Fig. 5 (top).

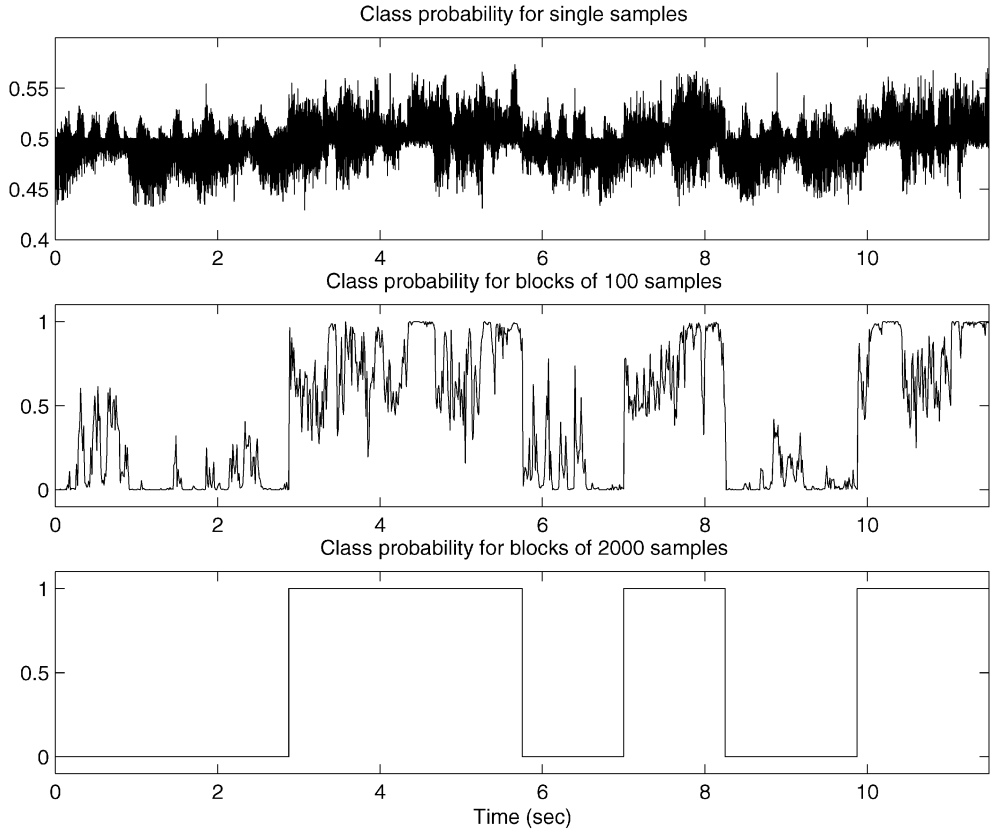

Fig. 6. The class conditional probability $p(C_2|\mathbf{x}_t, \theta_2)$. (Top) Class probability for single samples. (Middle) Class probability for blocks of 100 samples. (Bottom) Class probability for blocks of 2,000 samples.

The Signal to Noise Ratio (SNR)[6] for the experiment with a block size of 100 samples was 20.8 dB and 21.8 dB using the context switching ICA mixture model. Standard ICA algorithms are able to learn only one basis matrix. The SNR using infomax ICA [5] was 8.3 dB and 6.5 dB, respectively.

5 Learning Efficient Codes for Images

Recently, several methods have been proposed to learn image codes that utilize a set of linear basis functions. Olshausen and Field [32] used a sparseness criterion and found codes that were similar to localized and oriented receptive fields. Similar results were presented by Bell and Sejnowski [6] using the infomax ICA algorithm and by Lewicki and Olshausen [26] using a Bayesian approach. By applying the ICA mixture model, we present results that show a higher degree of flexibility in encoding the images. In this example, we used the ICA algorithm with the Laplacian prior on the source coefficients. The sparse prior leads to efficients codes.

We used images of natural scenes obtained from Olshausen and Field [32] and text images of scanned newspaper articles. The data set consisted of 12 by 12 pixel patches selected randomly from both image types. Fig. 7 illustrates examples of those image patches. Two complete basis vectors $\mathbf{A}_1$ and $\mathbf{A}_2$ were randomly initialized. Then, for each gradient in (6), a stepsize was computed as a function of the amplitude of the basis vectors and the number of iterations. The algorithm converged after 100,000 iterations and learned two classes of basis functions. Fig. 8 (top) shows the learned basis functions corresponding to natural images. The basis functions show Gabor[7]-like structure as previously reported [32], [5], [26]. However, the basis functions corresponding to text images (Fig. 8 (bottom)) resemble bars with different lengths and widths that capture the high-frequency structure present in the text images. Note that unlike the case in k-means clustering or clustering with spherical Gaussians, the classes can be spatially overlapping. In the example of the natural images and newspaper text, both classes had zero mean and the pattern vectors were only distinguished by their relative probabilities under the different classes.

5.1 Comparing Coding Efficiency

We have compared the coding efficiency between the ICA mixture model and similar models using Shannon's theorem to obtain a lower bound on the number of bits required to encode the pattern [29], [28].

$$\#\text{bits} \geq -\log_2 P(\mathbf{x}_t|\mathbf{A}_k) - N\log_2(\sigma_x), \qquad (12)$$

where N is the dimensionality of the input pattern $\mathbf{x}_t$ and σ_x is the coding precision (standard deviation of the noise

6. The SNR measures the difference in signal power between the original signal and the noise signal. The noise signal is computed as the difference between the original signal and the recovered signal.

7. A Gaussian modulated sinusoid.

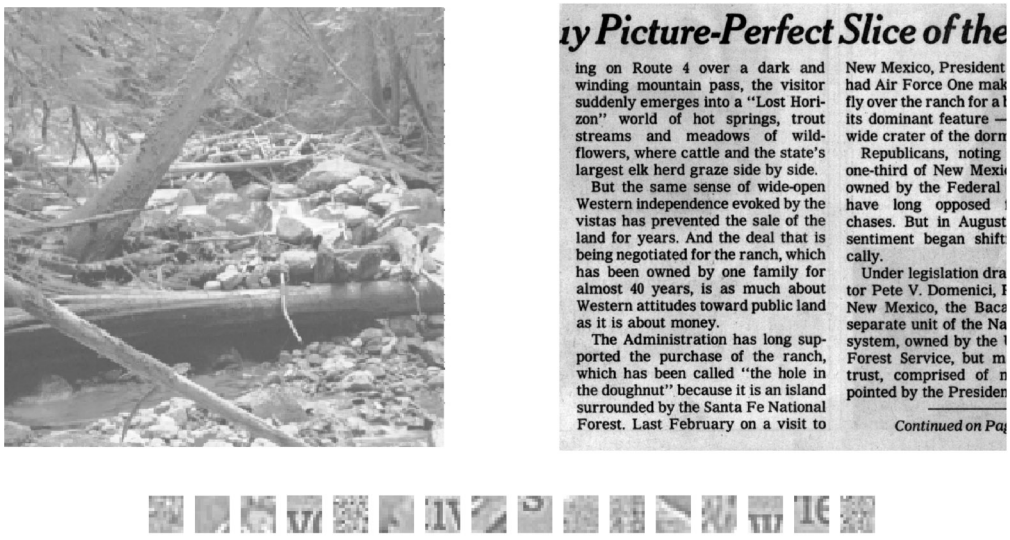

Fig. 7. Example of natural scene and text image. The 12 by 12 pixel image patches were randomly sampled from the images and used as inputs to the ICA mixture model.

introduced by errors in encoding). Table 1 compares the coding efficiency of five different methods. It shows the number of bits required to encode three different test data sets (5,000 image patches from natural scenes, 5,000 image patches from text images, and 5,000 image patches from both image types) using five different encoding methods (ICA mixture model, nature-adapted ICA, text-adapted ICA, nature and text-adapted ICA, and PCA-adapted on all three test sets). The ICA basis functions adapted on natural scene images exhibited the best encoding only for natural scenes (column: nature). The same occurred when text images were used for adapting and testing (column: text). Note that text adaptation yielded a reasonable basis for both data sets but nature adaptation gave a good basis only for nature data. The ICA mixture model gave the same encoding power for the individual test data sets and it had the best encoding when both image types are present. The difference in coding efficiency between the ICA mixture model and PCA was significant (more than 20 percent). ICA mixtures yielded a small improvement over ICA adapted on both image types. We expect the size of the improvement to be greater in situations where there are greater differences among the classes. An advantage of the mixture model is that each image patch is automatically classified.

6 Discussion

The new algorithm for unsupervised classification presented here is based on a mixture model using ICA to model the structure of the classes. The parameters are estimated using maximum likelihood. We have demonstrated that the algorithm can learn efficient codes to represent different image types such as natural scenes and text images and was a a significant improvement over PCA encoding. Single-class ICA models showed image compression rates comparable to or better than traditional image compression algorithms, such as JPEG [28]. Using ICA mixture to learn image codes should yield additional improvement in coding efficiency.

The ICA mixture model is a nonlinear model in which the data structure within each class is modeled using linear superposition of basis functions. The choice of class, however, is nonlinear because the classes are assumed to be mutually exclusive. This model is, therefore, a type of nonlinear ICA model and it is one way of relaxing the independence assumption over the entire data set. The ICA mixture model is a conditional independence model, i.e., the independence assumption holds only within each class and there may be dependencies among the classes. A different view of the ICA mixture model is to think of the classes as an overcomplete representation. Compared to the approach of Lewicki and Sejnowski [27], [29], the main difference is that the basis functions learned here are mutually exclusive, i.e., each class used its own (complete) set of basis functions.

This method is similar to other approaches including the mixture density networks by Bishop [7] in which a neural network was used to find arbitrary density functions. This algorithm reduces to the Gaussian mixture model when the source priors are Gaussian. A purely Gaussian structure, however, is rare in real data sets. Here, we have used super-Gaussian and sub-Gaussian densities as priors. These priors could be extended as proposed by Attias [4]. The model was used for learning a complete set of basis functions without additive noise. However, the method can be extended to take into account additive Gaussian noise and an overcomplete set of basis vectors [27], [29]. The structure of the ICA mixture model is also similar to the mixtures of factor analyzers proposed by Ghahramani and Hinton [15]. Here, the difference is that the coefficient distribution $p(s)$ and, hence, the distribution $p(\mathbf{X}|\Theta)$ are assumed to be non-Gaussian.

Several experiments using ICA mixture models have been performed on benchmark data sets for classification problems [24]. The results were comparable to or improved over those obtained by AutoClass [37] which

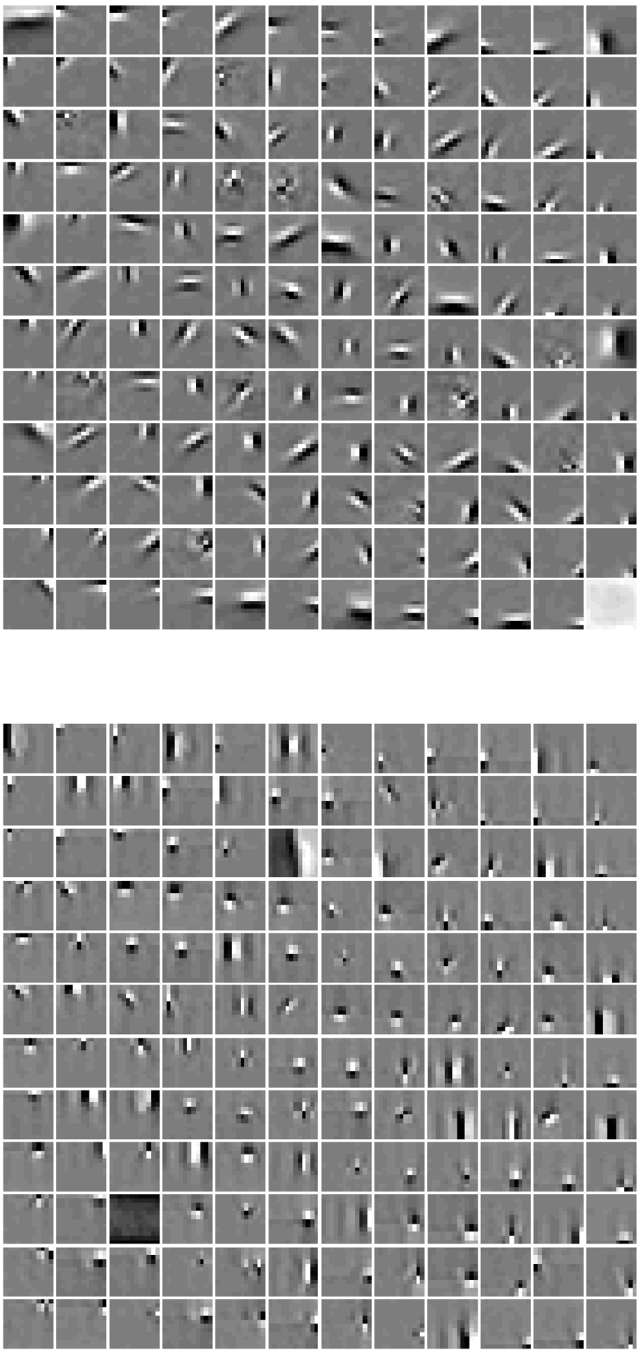

Fig. 8. (Left) Basis function class corresponding to natural images. (Right) Basis function class corresponding to text images.

uses a Gaussian mixture model. The algorithm has also been applied to blind source separation in nonstationary environments, where it can switch automatically between learned basis matrices in different environments [24]. Potential applications of the proposed method include noise removal and filling in missing pixels. Another application is the automatic detection of sleep stages by observing EEG signals. The method can identify these stages due to the changing source priors and their basis matrices.

This method provides greater flexibility than Gaussian mixture models in modeling structure in high-dimensional data and has many potential applications.

APPENDIX A
DERIVATION OF THE ICA MIXTURE MODEL ALGORITHM

We assume that $p(\mathbf{X}|\Theta)$ as given by (1) is a differentiable function of Θ. The log-likelihood L is then

$$L = \sum_{t=1}^{T} \log p(\mathbf{x}_t|\Theta) \tag{13}$$

and using (2), the gradient for the parameters of each class k is

$$\begin{aligned}\nabla_{\theta_k} L &= \sum_{t=1}^{T} \frac{1}{p(\mathbf{x}_t|\Theta)} \nabla_{\theta_k} p(\mathbf{x}_t|\Theta) \\ &= \sum_{t=1}^{T} \frac{\nabla_{\theta_k}\left[\sum_{k=1}^{K} p(\mathbf{x}_t|C_k,\theta_k)p(C_k)\right]}{p(\mathbf{x}_t|\Theta)} \\ &= \sum_{t=1}^{T} \frac{\nabla_{\theta_k} p(\mathbf{x}_t|C_k,\theta_k)p(C_k)}{p(\mathbf{x}_t|\Theta)}.\end{aligned} \tag{14}$$

Using the Bayes relation, the class probability for a given data vector $\mathbf{x}_t$ is

$$p(C_k|\mathbf{x}_t,\Theta) = \frac{p(\mathbf{x}_t|\theta_k,C_k)p(C_k)}{\sum_k p(\mathbf{x}_t|\theta_k,C_k)p(C_k)}. \tag{15}$$

Substituting (15) in (14) leads to

$$\begin{aligned}\nabla_{\theta_k} L &= \sum_{t=1}^{T} p(C_k|\mathbf{x}_t,\Theta) \frac{\nabla_{\theta_k} p(\mathbf{x}_t|\theta_k,C_k)p(C_k)}{p(\mathbf{x}_t|\theta_k,C_k)p(C_k)} \\ &= \sum_{t=1}^{T} p(C_k|\mathbf{x}_t,\Theta)\nabla_{\theta_k} \log p(\mathbf{x}_t|C_k,\theta_k).\end{aligned} \tag{16}$$

The log-likelihood function in (16) is the log-likelihood for each class. For the present model, the class log-likelihood is given by the log-likelihood for the standard ICA model:

$$\begin{aligned}\log p(\mathbf{x}_t|\theta_k,C_k) &= \log \frac{p(\mathbf{s}_t)}{|\det \mathbf{A}_k|} \\ &= \log p(\mathbf{A}_k^{-1}(\mathbf{x}_t - \mathbf{b}_k)) - \log|\det \mathbf{A}_k|.\end{aligned} \tag{17}$$

Gradient ascent is used to estimate the parameters that maximize the log-likelihood. The gradient parameters for each class are the gradient of the basis functions and the gradient of the bias vector $\nabla_{\theta_k} L = \{\nabla_{\mathbf{A}_k} L, \nabla_{\mathbf{b}_k} L\}$. We consider each in turn.

A.1 Estimating the Basis Matrix

Adapt the basis functions $\mathbf{A}_k$ for each class with (16).

$$\nabla_{A_k} L = \sum_{t=1}^{T} p(C_k|\mathbf{x}_t,\Theta)\nabla_{\mathbf{A}_k} \log p(\mathbf{x}_t|C_k,\theta_k). \tag{18}$$

The adaptation is performed by using gradient ascent with the gradient of the component density with respect to the basis functions giving

$$\Delta\mathbf{A}_k \propto p(C_k|\mathbf{x}_t,\Theta)\nabla_{\mathbf{A}_k} \log p(\mathbf{x}_t|C_k,\theta_k). \tag{19}$$

In the basis functions adaptation, the gradient of the component density with respect to the basis functions $\mathbf{A}_k$

TABLE 1
Comparing Coding Efficiency

	Test data (bits/pixel)		
Data set and model	**Nature**	**Text**	**Nature and Text**
ICA mixtures	4.7	5.2	5.0
Nature-adapted ICA	4.7	9.6	7.2
Text-adapted ICA	5.0	5.2	5.1
Nature- and text-adapted ICA	4.8	5.3	5.1
PCA	6.2	6.0	6.1

Coding efficiency (bits per pixel) of five methods is compared for three test sets. Coding precision was set to 7 bits (Nature: $\sigma_x = 0.016$ and Text: $\sigma_x = 0.029$).

is weighted by $p(C_k|\mathbf{x}_t, \Theta)$. This computation for $\nabla_{\mathbf{A}_k} \log p(\mathbf{x}_t|C_k, \theta_k)$ will be further detailed in Appendix B.

A.2 Estimating the Bias Vectors

We can use (16) to adapt the bias vectors $\mathbf{b}_k$ for each class.

$$\nabla_{\mathbf{b}_k} L = \sum_{t=1}^{T} p(C_k|\mathbf{x}_t, \Theta) \nabla_{\mathbf{b}_k} \log p(\mathbf{x}_t|C_k, \theta_k). \tag{20}$$

The adaptation is performed by using gradient ascent with the gradient of the component density with respect to the bias vector $\mathbf{b}_k$ giving

$$\Delta \mathbf{b}_k \propto p(C_k|\mathbf{x}_t, \Theta) \nabla_{\mathbf{b}_k} \log p(\mathbf{x}_t|C_k, \theta_k). \tag{21}$$

Using (17) in (21), we can adapt $\mathbf{b}_k$ as follows:

$$\Delta \mathbf{b}_k \propto p(C_k|\mathbf{x}_t, \Theta) \nabla_{\mathbf{b}_k} \left[\log p(\mathbf{A}_k^{-1}(\mathbf{x}_t - \mathbf{b}_k)) - \log |\det \mathbf{A}_k| \right]. \tag{22}$$

Instead of using the gradient, we may also use an approximate method for the adaptation of the bias vectors. The maximum likelihood estimate $\hat{\Theta}$ must satisfy the condition

$$\sum_{t=1}^{T} p(C_k|\mathbf{x}_t, \hat{\Theta}) \nabla_{\theta_k} \log p(\mathbf{x}_t|C_k, \hat{\theta}_k) = 0, \qquad k = 1, \ldots, K. \tag{23}$$

We can use (23) to adapt the bias vector or mean vector $\mathbf{b}_k$.

$$\begin{aligned} \nabla_{\mathbf{b}_k} L &= 0 \\ \sum_{t=1}^{T} p(C_k|\mathbf{x}_t, \Theta) \nabla_{\mathbf{b}_k} \log p(\mathbf{x}_t|\theta_k, C_k) &= 0. \end{aligned} \tag{24}$$

Substituting (17) into (24) shows that the gradient of the first term in (17) must be zero. From this, it follows that

$$\nabla_{\mathbf{b}_k} \log p(\mathbf{A}_k^{-1}(\mathbf{x}_t - \mathbf{b}_k)) = 0. \tag{25}$$

Assuming that we observe a large amount of data $\mathbf{x}_t$ and the probability density function (p.d.f.) of the prior $p(\mathbf{s}_t)$ is symmetric and differentiable, then $\log p(\mathbf{s}_t)$ will be symmetric as well and the bias vector can be approximated by the weighted average of the data samples

$$\mathbf{b}_k = \frac{\sum_t \mathbf{x}_t p(C_k|\mathbf{x}_t, \Theta)}{\sum_t p(C_k|\mathbf{x}_t, \Theta)}. \tag{26}$$

APPENDIX B
ICA LEARNING ALGORITHM

The gradient of the log component density for each class $\log p(\mathbf{x}_t|C_k, \theta_k)$ can be computed using ICA. This section sketches the ICA learning algorithm in [17] and [22].

Assume that there is an M-dimensional zero mean vector for each class.[8] $\mathbf{s}_t = [s_1(t), \cdots, s_M(t)]^T$, whose components are mutually independent. The vector $\mathbf{s}_t$ corresponds to M independent scalar-valued source signals $s_{i,t}$. We can write the multivariate p.d.f. of the vector as the product of marginal independent distributions

$$p(\mathbf{s}) = \prod_{i=1}^{M} p_i(s_i). \tag{27}$$

A data vector $\mathbf{x}_t = [x_1(t), \cdots, x_T(t)]^T$ is observed at each time point t, such that

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t, \tag{28}$$

where $\mathbf{A}$ is a $N \times M$ scalar matrix. As the components of the observed vectors are no longer independent, the multivariate p.d.f. will not satisfy the product equality of (27).

8. For simplicity, we have omitted the class index k for the derivation of the ICA learning rule.

The goal of ICA is to find a linear transformation $\mathbf{W}$ of the dependent sensor signals $\mathbf{x}$ that makes the outputs $\mathbf{u}$ as independent as possible

$$\mathbf{u}_t = \mathbf{W}\mathbf{x}_t = \mathbf{W}\mathbf{A}\mathbf{s}_t, \tag{29}$$

so that $\mathbf{u}$ is an 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 learning algorithm can be derived using the maximum-likelihood estimation (MLE). The MLE approach to blind source separation was first proposed by Gaeta and Lacoume [14], Pham and Garrat [36], and was pursued more recently by MacKay [30], Peralmutter and Parra [34], and Cardoso [8]. The probability density function of the observations $\mathbf{x}$ can be expressed as [33], [2]:

$$p(\mathbf{x}) = |\det(\mathbf{W})| p(\mathbf{u}), \tag{30}$$

where $p(\mathbf{u}) = \prod_{i=1}^{N} p_i(u_i)$ is the hypothesized distribution of $p(\mathbf{s})$. The log-likelihood of (30) is

$$L(\mathbf{u}, \mathbf{W}) = \log|\det(\mathbf{W})| + \sum_{i=1}^{N} \log p_i(u_i). \tag{31}$$

Maximizing the log-likelihood with respect to $\mathbf{W}$ gives a learning algorithm for $\mathbf{W}$ [5]:

$$\Delta\mathbf{W} \propto \left[(\mathbf{W}^T)^{-1} - \varphi(\mathbf{u})\mathbf{x}^T\right], \tag{32}$$

where

$$\varphi(\mathbf{u}) = -\frac{\frac{\partial p(\mathbf{u})}{\partial \mathbf{u}}}{p(\mathbf{u})} = \left[-\frac{\frac{\partial p(u_1)}{\partial u_1}}{p(u_1)}, \cdots, -\frac{\frac{\partial p(u_N)}{\partial u_N}}{p(u_N)}\right]^T. \tag{33}$$

An efficient way to maximize the log-likelihood is to follow the "natural" gradient [1]:

$$\Delta\mathbf{W} \propto \frac{\partial L(\mathbf{u}, \mathbf{W})}{\partial \mathbf{W}} \mathbf{W}^T\mathbf{W} = \left[\mathbf{I} - \varphi(\mathbf{u})\mathbf{u}^T\right]\mathbf{W} \tag{34}$$

as proposed by Amari et al. [3] or the relative gradient, Cardoso and Laheld [10]. Here, $\mathbf{W}^T\mathbf{W}$ rescales the gradient, simplifies the learning rule in (32) and speeds convergence considerably. It has been shown that the general learning algorithm in (34) can be derived from several theoretical viewpoints, such as MLE [34], infomax [5], and negentropy maximization [18]. Lee et al. [21] review these techniques and show their relation to each other.

The parametric density estimate $p_i(u_i)$ plays an essential role in the success of the learning rule in (34). Local convergence is assured if $p_i(u_i)$ is an estimate of the true source density [36]. For example, the sigmoid function used in Bell and Sejnowski [5] learning algorithm is suited to separating super-Gaussian sources, i.e., p.d.f.s that have heavier tails than the Gaussian density.

A way of generalizing the learning rule to sources with either sub-Gaussian or super-Gaussian distributions is to derive a separate learning rule for sub-Gaussian and super-Gaussian components. A symmetric strictly sub-Gaussian density can be modeled using a symmetrical form of the Pearson mixture model [35] as follows: [17], [16].

$$p(u) = \frac{1}{2}\big(N(\mu, \sigma^2) + N(-\mu, \sigma^2)\big), \tag{35}$$

where $N(\mu, \sigma^2)$ is the normal density with mean μ and variance σ^2. For $\mu = 0$, $p(u)$ is a Gaussian model, otherwise $p(u)$ is bimodal. Setting $\mu = 1$ and $\sigma^2 = 1$, (33) reduces to [17]

$$\varphi(u) = u - \tanh(u). \tag{36}$$

In the case of unimodal super-Gaussian sources, we adopt the following density mode

$$p(u) \propto N(u)\mathrm{sech}^2(u), \tag{37}$$

where $N(u)$ is a zero-mean Gaussian density with unit variance. The nonlinearity $\varphi(u)$ is now

$$\varphi(u) = -\frac{\frac{\partial p(u)}{\partial u}}{p(u)} = u + \tanh(u). \tag{38}$$

The two equations can be combined as,

$$\begin{aligned} &\Delta\mathbf{W} \propto \left[\mathbf{I} - \mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T\right]\mathbf{W} \\ &\begin{cases} k_i = 1 & \text{super-Gaussian} \\ k_i = -1 & \text{sub-Gaussian,} \end{cases} \end{aligned} \tag{39}$$

where k_i are elements of the N-dimensional diagonal matrix $\mathbf{K}$. The k_is can be derived from the generic stability analysis [9] of separating solutions. This yields the choice of k_is used by Lee et al. [22],

$$k_i = \mathrm{sign}\Big(E\{\mathrm{sech}^2(s_{i,t})\}E\{s_{i,t}^2\} - E\{[\tanh(s_{i,t})]s_{i,t}\}\Big), \tag{40}$$

which ensures stability of the learning rule [22].

B.1 ICA Mixture Model Learning Rules

We can write (39) and (40) in terms of the basis functions for each class $\mathbf{A}_k$ in (19)

$$\Delta\mathbf{A}_k \propto p(C_k|\mathbf{x}_t, \Theta)\mathbf{A}_k\left[\mathbf{I} - \mathbf{K}\tanh(\mathbf{s}_k)\mathbf{s}_k^T - \mathbf{s}_k\mathbf{s}_k^T\right], \tag{41}$$

where

$$\mathbf{s}_k = \mathbf{A}_k^{-1}(\mathbf{x}_t - \mathbf{b}_k), \tag{42}$$

and

$$k_{k,i} = \mathrm{sign}\Big(E\{\mathrm{sech}^2(s_{k,i,t})\}E\{s_{k,i,t}^2\} - E\{[\tanh(s_{k,i,t})]s_{k,i,t}\}\Big). \tag{43}$$

The source distribution is super-Gaussian when $k_{k,i} = 1$ and sub-Gaussian when $k_{k,i} = -1$. The adaptation of the log prior $\log p(\mathbf{s}_k)$ can be approximated as follows:

$$\log p(\mathbf{s}_{k,t}) \propto -\sum_{i=1}^{N}\left(k_{k,i}\log(\cosh s_{k,i,t}) - \frac{s_{k,i,t}^2}{2}\right). \tag{44}$$

These are the equations used for unsupervised classification and automatic context switching in the examples given in the text.

For the learning of image codes, a Laplacian model was used to learn sparse representations. The simplified learning rule uses (19) and (45).

$$\Delta \mathbf{A}_k \propto p(C_k|\mathbf{x}_t, \Theta)\mathbf{A}_k\left[\mathbf{I} - \text{sign}(\mathbf{s}_k)\mathbf{s}_k^T\right]. \quad (45)$$

The log prior simplifies to

$$\log p(\mathbf{s}_k) \propto -\sum_i |s_{k,i}| \qquad \text{Laplacian prior.} \quad (46)$$

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their detailed comments and questions which improved the quality of the presentation of this paper. T.-W. Lee was supported by the Swartz Foundation. T.J. Sejnowski was supported by the Howard Hughes Medical Institute.

REFERENCES

[1] S. Amari, "Natural Gradient Works Efficiently in Learning," *Neural Computation,* vol. 10, no. 2, pp. 251-276, 1998.
[2] S. Amari and J.-F. Cardoso, "Blind Source Separation—Semiparametric Statistical Approach," *IEEE Trans. Signal Processing,* vol. 45, no. 11, pp. 2,692-2,700, 1997.
[3] S. Amari, A. Cichocki, and H. Yang, "A New Learning Algorithm for Blind Signal Separation," *Advances in Neural Information Processing Systems 8,* pp. 757-763, 1996.
[4] H. Attias, "Blind Separation of Noisy Mixtures: An EM Algorithm for Independent Factor Analysis," *Neural Computation,* vol. 11, pp. 803-851, 1999.
[5] A.J. Bell and T.J. Sejnowski, "An Information-Maximization Approach to Blind Separation and Blind Deconvolution," *Neural Computation,* vol. 7, pp. 1,129-1,159, 1995.
[6] A.J. Bell and T.J. Sejnowski, "The "Independent Components" of Natural Scenes are Edge Filters," *Vision Research,* vol. 37, no. 23, pp. 3,327-3,338, 1997.
[7] C. Bishop, "Mixture Density Networks," Technical Report NCRG/4,288, 1994.
[8] J.-F. Cardoso, "Infomax and Maximum Likelihood for Blind Source Separation," *IEEE Signal Processing Letters,* vol. 4, no. 4, pp. 112-114, 1997.
[9] J.-F. Cardoso, "Blind Signal Separation: Statistical Principles," *Proc. IEEE,* vol. 86, no. 10, pp. 2,009-2,025, 1998.
[10] J.-F. Cardoso and B. Laheld, "Equivariant Adaptive Source Separation," *IEEE Trans. Signal Processing,* vol. 45, no. 2, pp. 434-444, 1996.
[11] P. Comon, "Independent Component Analysis—A New Concept?," *Signal Processing,* vol. 36, no. 3 pp. 287-314, 1994.
[12] R. Duda and P. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.
[13] R. Fisher, "The Use of Multiple Measurements in Taxonomic Problem," *Ann. Eugenics,* vol. 7, part II, pp. 179-188. 1936.
[14] M. Gaeta and J.-L. Lacoume, "Source Separation without Prior Knowledge: The Maximum Likelihood Solution," *Proc. EUSIPO,* pp. 621-624, 1990.
[15] Z. Ghahramani and G.E. Hinton, "The EM Algorithm for Mixtures of Factor Analyzers," technical report, Dept. Computer Science, Univ. Toronto, 1997.
[16] M. Girolami, "Self-Organizing Artificial Neural Networks for Signal Separation," PhD thesis, Dept. Computing and Information Systems, Paisley Univ., Scotland, 1997.
[17] M. Girolami, "An Alternative Perspective on Adaptive Independent Component Analysis Algorithms," *Neural Computation,* vol. 10, no. 8, pp. 2,103-2,114, 1998.
[18] M. Girolami and C. Fyfe, "Generalised Independent Component Analysis through Unsupervised Learning with Emergent Bussgang Properties," *Proc. ICNN,* pp. 1,788-1,891, 1997.
[19] A. Hyvaerinen and E. Oja, "A Fast Fixed-Point Algorithm for Independent Component Analysis," *Neural Computation,* vol. 9, pp. 1,483-1,492, 1997.
[20] C. Jutten and J. Herault, "Blind Separation of Sources, Part I: An Adaptive Algorithm-Based on Neuromimetic Architecture," *Signal Processing,* vol. 24, pp. 1-10, 1991.
[21] T.-W. Lee, M. Girolami, A.J. Bell, and T.J. Sejnowski, "A Unifying Framework for Independent Component Analysis," *Computers and Math. with Applications,* vol. 39, no. 11, pp. 1-21, 2000.
[22] T.-W. Lee, M. Girolami, and T.J. Sejnowski, "Independent Component Analysis Using an Extended Infomax Algorithm for Mixed Sub-Gaussian and Super-Gaussian Sources," *Neural Computation,* vol. 11, no. 2, pp. 409-433, 1999.
[23] T.-W. Lee and M.S. Lewicki, "The Generalized Gaussian Mixture Model Using ICA," *Proc. Int'l Workshop ICA,* pp. 239-244, 2000.
[24] T.-W. Lee, M.S. Lewicki, and T.J. Sejnowski, "ICA Mixture Models for Unsupervised Classification and Automatic Context Switching," *Proc. Int'l Workshop ICA,* pp. 209-214, 1999.
[25] T.-W. Lee, M.S. Lewicki, and T.J. Sejnowski, "Unsupervised Classification with Non-Gaussian Mixture Models Using ICA," *Advances in Neural Information Processing Systems 11,* pp. 508-514, 1999.
[26] M. Lewicki and B. Olshausen, "Inferring Sparse, Overcomplete Image Codes Using an Efficient Coding Framework," *Advances in Neural Information Processing Systems 10,* pp. 556-562, 1998.
[27] M. Lewicki and T.J. Sejnowski, "Learning Nonlinear Overcomplete Represenations for Efficient Coding," *Advances in Neural Information Processing Systems 10,* pp. 815-821, 1998.
[28] M.S. Lewicki and B.A. Olshausen, "A Probabilistic Framework for the Adaptation and Comparison of Image Codes," *J. Opt. Soc. of Am. A: Optics, Image Science, and Vision,* vol. 6, no. 7, pp. 1,587-1,601, 1999.
[29] M.S. Lewicki and T.J. Sejnowski, "Learning Overcomplete Representations," *Neural Computation,* vol. 12, no. 2, pp. 337-365, 2000.
[30] D. MacKay, "Maximum Likelihood and Covariant Algorithms for Independent Component Analysis," report, Univ. Cambridge, Cavendish Lab, 1996.
[31] C. Merz and P. Murphy, "UCI Repository of Machine Learning Databases," 1998.
[32] B. Olshausen and D. Field, "Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images," *Nature,* vol. 381, pp. 607-609, 1996.
[33] *Probability and Statistics,* A. Papoulis, ed., vol. 1, N.J.: Prentice Hall, 1990.
[34] B. Pearlmutter and L. Parra, "A Context-Sensitive Generalization of ICA," *Proc. Int'l Conf. Neural Information Processing,* pp. 151-157, 1996.
[35] K. Pearson, "Contributions to the Mathematical Study of Evolution," *Phil. Trans. Royal Soc. A,* vol. 185, no. 71, 1894.
[36] D.-T. Pham and P. Garrat, "Blind Separation of Mixture of Independent Sources through a Quasi-Maximum Likelihood Approach," *IEEE Trans. Signal Processing,* vol. 45, no. 7, pp. 1,712-1,725, 1997.
[37] J. Stutz and P. Cheeseman, "Autoclass—A Bayesian Approach to Classification," *Maximum Entropy and Bayesian Methods,* Kluwer Academic Publishers,. 1994.

Te-Won Lee received his diploma degree in March 1995 and his PhD degree in October 1997 with the highest honor in electrical engineering from the University of Technology, Berlin. He was a visiting graduate student at the Institute Nationale Polytechnique de Grenoble, the University of California at Berkeley, and the Carnegie Mellon University. From 1995 to 1997, he was a Max-Planck Institute fellow. He is currently a research professor at the Institute for Neural Computation, University of California, San Diego and a research associate at The Salk Institute. His research interests are in the areas of unsupervised learning algorithms, artificial neural networks, and Bayesian probability theory with applications to biomedical signal processing and data mining. He is a member of the IEEE.

Michael S. Lewicki received the BS degree in mathematics and cognitive science in 1989 from Carnegie Mellon University. He received the PhD degree in computation and neural systems from the California Institute of Technology in 1996. From 1996 to 1998, he was a postdoctoral fellow in the Computational Neurobiology Laboratory at the Salk Institute. Dr. Lewicki is currently an assistant professor in the Computer Science Department at Carnegie Mellon University and at the CMU-University of Pittsburgh Center for the Neural Basis of Cognition. His research involves the study and development of computational approaches to the representation, processing, and learning of structure in natural patterns.

Terrence J. Sejnowski received the BS degree in physics from the Case-Western Reserve University, the MA degree in physics from Princeton University, and the PhD degree in physics from Princeton University in 1978. In 1982, he joined the faculty of the Department of Biophysics at the Johns Hopkins University and moved to San Diego in 1988. He is an investigator with the Howard Hughes Medical Institute and a professor at The Salk Institute for Biological Studies where he directs the Computational Neurobiology Laboratory. He is also a professor of biology and an adjunct professor in the Departments of Physics, Neurosciences, Psychology, Cognitive Science, and Computer Science and Engineering at the University of California, San Diego, where he is Director of the Institute for Neural Computation. He founded Neural Computation, published by the MIT Press, the leading journal in the area of neural networks and computational neuroscience. He is also the president of the Neural Information Processing Systems Foundation, a nonprofit organization that oversees the annual NIPS Conference. This interdisciplinary meeting brings together researchers from many disciplines, including biology, physics, mathematics, and engineering. The long-range goal Dr. Sejnowski's research is to build linking principles from the brain to behavior using computational models. He is a fellow of the IEEE.