Fast Algorithms for Mutual Information Based Independent Component Analysis

Dinh-Tuan Pham, *Member, IEEE*

***Abstract*—This paper provides fast algorithms to perform independent component analysis based on the mutual information criterion. The main ingredient is the binning technique and the use of cardinal splines, which allows the fast computation of the density estimator over a regular grid. Using a discretized form of the entropy, the criterion can be evaluated quickly together with its gradient, which can be expressed in terms of the score functions. Both offline and online separation algorithms have been developed. Our density, entropy, and score estimators also have their own interest.**

***Index Terms*—Binning, blind source separation, independence component analysis, cardinal spline, entropy estimation, kernel density estimation, mutual information, score function estimation.**

NOMENCLATURE

$\mathbf{X}(t) = [X_1(t) \cdots X_K(t)]^T$	Observation vector (^{T} denoting the transpose).
$\mathbf{S}(t) = [S_1(t) \cdots S_K(t)]^T$	Source vector.
$\mathbf{Y}(t) = [Y_1(t) \cdots Y_K(t)]^T$	Separated source vector.
$\mathbf{A}$	Mixing matrix.
$\mathbf{B}$	Separating matrix.
p_{Y_k}	Density of Y_k.
$p_{\mathbf{Y}}$	Joint density of $Y_1, \ldots Y_K$ or density of $\mathbf{Y}$.
$H(Y_k)$	entropy of Y_k.
$H(\mathbf{Y})$	Joint entropy of $Y_1, \ldots Y_K$ or entropy of $\mathbf{Y}$.
$\hat{p}_{Y_k}$	Density estimator for Y_k.
$\hat{H}(Y_k)$	Entropy estimator for Y_k.
κ	Kernel function (for density estimator).
h	Bandwidth (of density estimator).
b	Binwidth (in binning method).
δ	Ratio b/h.
$\mathbb{1}^{\star r}_{(0,1]}$	Cardinal spline of degree r.
$\hat{\pi}_Y(m - r/2)$	Estimated probability for Y in cell of size b centered at $[m - r/2]b$.
$\bar{Y}$	Sample mean of Y.
$\hat{\sigma}_Y$	Sample standard deviation of Y.
$\hat{\hat{\pi}}_Y(m - L/2 - r/2)$	Smoothed estimated probability for Y in cell of size b centered at $\bar{Y} + [m - (L + r)/2]b$.
$\hat{\psi}_Y$	Estimated score function of Y.
$Y\tilde{\psi}_Y$	Raw estimated score function of Y.
$\bar{\tilde{\psi}}_Y$	mean raw score of Y.
λ_Y	Correction for coefficient for score estimator.
$\widehat{\log}\,\pi_Y(j + (1/2))$	Smoothed estimate of the logarithm of the probability for Y in cell of size b centered at $\bar{Y} + (j + 1/2)b$.
$\hat{J}_Y$	Estimated Fisher information for Y.
$\tilde{J}_Y$	Raw estimate Fisher information for Y.

I. INTRODUCTION

INDEPENDENT component analysis (ICA) has its origin in the blind source separation (BSS) problem, which aims at separating sources from their mixtures, without relying on any specific knowledge of the sources. In its simplest form, one observes K sequences $X_1(t), \ldots, X_K(t)$, each being a linear combination of K independent sources $S_1(t), \ldots, S_K(t)$ so that $\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t)$, where $\mathbf{X}(t)$ and $\mathbf{S}(t)$ denote the vectors with components $X_1(t), \ldots, X_K(t)$, and $S_1(t), \ldots, S_K(t)$, respectively, and $\mathbf{A}$ is an invertible matrix. The goal is to recover the sources, as the components $Y_k(t)$ of the vector $\mathbf{Y}(t) = \mathbf{B}\mathbf{X}(t)$, where $\mathbf{B}$ is a matrix to be determined. Since no specific knowledge of the sources are available, the idea is to exploit the fundamental assumption that they are independent and thus determine $\mathbf{B}$ such that the random variables $Y_1(t), \ldots, Y_K(t)$ are as independent as it is possible. This is precisely the goal of ICA. Thus, ICA and BSS are often used interchangeably. The subtle difference is that ICA does not really postulate a mixture model.

To solve the above problem, mutual information (MI) and infomax criteria were first introduced by Bell and Sejnowski [1]. Similar approaches have been investigated by many other researchers [2]–[9]. To reduce the convergence time, Amari *et al.* [10]–[15] suggest using the natural gradient for minimizing the mutual information. This approach is also closely related to the maximum likelihood method (see, e.g., [8] and [16]). Assuming stationarity, we may ignore the time index and thus write Y_k in place of $Y_k(t)$. The mutual information between the random variables $Y_1, \ldots, Y_K$ is defined as

Manuscript received August 19, 2002; revised October 16, 2003. The associate editor coordinating the review of this paper and approving it for publication was Dr. Karim Abed-Meraim.

The author is with the Laboratory of Modeling and Computation, IMAG-C.N.R.S., Grenoble cedex 9, France (e-mail: Dinh-Tuan.Pham@imag.fr).

Digital Object Identifier 10.1109/TSP.2004.834398

[17] $\mathrm{E}\log[p_{\mathbf{Y}}(Y_1,\ldots,Y_K)/\prod_{k=1}^K p_{Y_k}(Y_k)]$, where p_{Y_i} and $p_{\mathbf{Y}}$ are the density of Y_i and of $\mathbf{Y}$ (or the joint density of $Y_1,\ldots,Y_K$), and E denotes the expectation operator. It can also be computed as $\sum_{k=1}^K H(Y_k) - H(Y_1,\ldots,Y_K)$, where $H(Y_k) = -\mathrm{E}\log p_{Y_k}(Y_k)$ denotes the Shannon entropy of Y_k, and $H(Y_1,\ldots,Y_K) = H(\mathbf{Y})$ denotes the joint entropy of $Y_1,\ldots,Y_K$ or the entropy of the vector $\mathbf{Y}$, which is defined similarly as $H(Y_k)$ but with Y_k replaced by $\mathbf{Y}$. Since $H(\mathbf{Y})$ equals $H(\mathbf{X}) + \log\det\mathbf{B}$ (see, e.g., [8]) and the term $H(\mathbf{X})$ does not depend on $\mathbf{B}$, minimizing the mutual information is the same as minimizing

$$C(\mathbf{B}) = \sum_{i=1}^K H(Y_k) - \log\det\mathbf{B}. \tag{1}$$

With respect to the mutual information, this criterion avoids the estimation of joint density, but it still requires the estimation of the density, which is *apparently* costly (not to mention the cost of minimization). Comon, in [2], has considered such criterion, but he found it too costly and went on proposing simplifications, leading to a kurtosis-based criterion. Pham [8] has proposed the same criterion but has not dwelled in detail on the computational aspect. In this work, we will derive algorithms that can compute and minimize (1) with a reasonable cost, which is comparable with that of other separation algorithms. It should be stressed that we do not aim to achieve an extremely fast algorithm but only to make the issue of computational cost no longer critical in the mutual information approach. The appeal of this approach lies elsewhere [on its relation with the maximum likelihood (hence, its statistical performance) and its flexibility] as the unknown density of the sources are themselves estimated from the data.

Note that most mutual information-based algorithms have ducked the computational issue by merely adopting a simple gradient algorithm and *replacing the unknown score functions in this gradient by some ad hoc function* (see, e.g., [18, p. 185]). Here, everything is estimated from the data; therefore, the method adapts itself fully to the particularity of the source distributions. Yet, we can keep the computational cost low through

1) the use of the binning technique for kernel density estimation;
2) the use of the cardinal spline as kernel;
3) a slight approximation of the entropy functional through discretization.

We also provide fast calculation of the gradient of the criterion and discuss the use of the quasi-Newton algorithm to speed up the minimization. Further, we develop an online version of our algorithm, with reasonable computational cost. Finally, two examples of simulation are given, illustrating the working of our algorithm. Our method for estimating the density, the entropy, and the score functions (in terms of which the gradient of the criterion is expressed) also have their own interest.

For ease of reading, technical materials are relegated to an Appendix. Further, a nomenclature of notations is given (as suggested by a reviewer).

II. Binned Kernel Density Estimator

A. Binning Method

Let Y be a random variable with density p_Y. The kernel estimator $\hat{p}_Y$ of p_Y, based on a sample $Y(1),\ldots,Y(N)$, is given by [19]

$$\hat{p}_Y(y) = \frac{1}{Nh}\sum_{n=1}^N \kappa\left[\frac{y - Y(n)}{h}\right] \tag{2}$$

where κ is the kernel, which is a density function, and h is a bandwidth (or smoothing) parameter to be adjusted according to the sample size and the data. Direct computation of this estimator is costly, however; it requires NM kernel evaluations to compute M values of the estimator. Therefore, a technique called binning [20]–[23], which can speed up the calculations considerably, has been introduced. The earliest form of binning, which we will refer to as simple binning, is no more than grouping the data into cells; letting the cell width be b, it amounts to replacing $Y(n)$ by $(\lfloor Y(n)/b\rfloor + 1/2)b$, where $\lfloor x\rfloor$ denotes the largest signed integer not exceeding x. Thus, the density estimate at $mb, m = \ldots,-1,0,1,\ldots$ would be $\sum_{l=-\infty}^{\infty} h^{-1}\kappa[(l + 1/2)b/h]\hat{\pi}_Y(m - l - 1/2)$, where $\hat{\pi}_Y(m - 1/2)$ is the fraction of data falling into the cell $[mb - b, mb)$. Later, linear binning has been introduced, which amounts to splitting the $Y(n)$ into two data points at $\lfloor Y(n)/b\rfloor$ and $\lfloor Y(n)/b\rfloor + 1$, with weighs $\lfloor Y(n)/b\rfloor + 1 - Y(n)/b$ and $Y(n)/b - \lfloor Y(n)/b\rfloor$, respectively. The density estimate at $mb, m = \ldots,-1,0,1,\ldots,$ would now be $\sum_{l=-\infty}^{\infty} h^{-1}\kappa(lb/h)\hat{\pi}_y(m-l)$, with $\hat{\pi}_Y(m)$ being the total weight contributed by all data to the point mb divided by N. There have been studies on the accuracy (in mean squares sense) of binning, which concludes that linear binning is better than simple binning [21], [23].

It has not been clearly recognized that binning is equivalent to just changing the kernel and computing the estimator only on a regularly spaced grid. The modified kernels are precisely

$$u \mapsto \sum_{l=-\infty}^{\infty} \kappa\left(\frac{lb + b/2}{h}\right) \mathbb{1}_{(0,1]}\left(\frac{uh}{b} - l\right)$$

and

$$u \mapsto \sum_{l=-\infty}^{\infty} \kappa\left(\frac{lb + b}{h}\right) \mathbb{1}_{(0,1]}^{\star 2}\left(\frac{uh}{b} - l\right)$$

for the simple and linear binning, where $\mathbb{1}_{[0,1)}$ is the indicator function of the interval $(0,1]$, and $\mathbb{1}_{(0,1]}^{\star r}$ denotes the r times convolution of $\mathbb{1}_{(0,1]}$ with itself (thus, $\mathbb{1}_{(0,1]}^{\star 2}(x) = 1 - |x-1|$ if $|x - 1| < 1, = 0$ otherwise). Recognizing this, it is easy to extend the idea of binning by replacing the function $\mathbb{1}_{[0,1)}$ or $\mathbb{1}_{(0,1]}^{\star 2}$ by a more complex function such as $\mathbb{1}_{(0,1]}^{\star r}, r > 2$. (Indeed, the use $r = 3$ and $r = 4$ has been introduced in [24].) However, the above modified kernel may depend on the data via h and b, but it is natural to take b proportional to h: $b = h\delta$, where δ is a constant (not deepening on the data, unlike h) since the higher the bandwidth parameter h, the smoother the density estimator, and hence, the larger the spacing b one can afford in computing it. Then, the kernel no longer depends on the data; it is just another kernel, and since the kernel shape is not very important (but the bandwidth is), one may very well consider a general kernel of the form

$$\kappa(u) = \sum_{l=-\infty}^{\infty} c_l \frac{1}{\delta} \mathbb{1}^{\star r}_{(0,1]} \left(\frac{u}{\delta} - l \right)$$

where $\{c_l\}$ is a sequence of positive coefficients summing to 1 (to ensure that κ is a density).

It is possible to generalize further by using some elementary kernel in place of $\mathbb{1}^{\star r}_{(0,1]}$. However, there is no advantage for this. The functions $\mathbb{1}^{\star r}_{(0,1]}$ are known as the cardinal splines and have been extensively studied [25] and possess nice properties; some of them are given in Appendix A. They are also easy to compute, except for large r, but there is no advantage to using r greater than 4. The main reason for considering $\mathbb{1}^{\star 3}_{(0,1]}$ and $\mathbb{1}^{\star 4}_{(0,1]}$ is that the first kernel has a continuous first derivative and only a finite number of jumps in the second derivative, and the second kernel has a continuous second derivative and only a finite number of jumps in the third derivative. Note that to ensure the symmetry of a kernel of the above form, the c_l must be symmetric around $-r/2$: $c_l = c_{-l-r}$ (as $\mathbb{1}^{\star r}_{[0,1)}$ is symmetric around $r/2$; see Appendix A). Since the sequence $\{c_l\}$ should contain only a finite number of nonzero terms (to reduce the calculation), it actually contains an odd or even number of such terms according to r even or odd. Thus, the above form of the kernel is not the most general as the parity of the number of nonzero term is imposed. For generality, we will shift the kernel by an appropriate integer or half integer multiple of the grid size and shift the index of c_l accordingly, which leads to a general class of kernels of the form

$$\kappa(u) = \sum_{l=0}^{L} c_l \frac{1}{\delta} \mathbb{1}^{\star r}_{(0,1]} \left(\frac{u}{\delta} + \frac{L+r}{2} - l \right) \tag{3}$$

with $c_0, \ldots, c_L$ being $L+1$ positive coefficients summing to 1 and symmetric around $L/2$: $c_l = c_{L-l}$.

With the kernel (3), the formula (2) for $\hat{p}_Y$ becomes

$$\hat{p}_Y(y) = \frac{1}{b} \sum_{l=0}^{L} c_l \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}^{\star r}_{(0,1]} \left[\frac{y - Y(n)}{b} + \frac{L+r}{2} - l \right].$$

Thus, the calculation of $\hat{p}_Y[(m - L/2 - r/2)b], m = \ldots, -1, 0, 1, \ldots$ can be done in two steps.

i) Compute $\hat{\pi}_Y(m - r/2) = (1/N) \sum_{n=1}^{N} \mathbb{1}^{\star r}_{(0,1]}[m - Y(n)/b], m = \ldots, -1, 0, 1, \ldots$
ii) Compute $\hat{p}_Y[(m - L/2 - r/2)b] = \sum_{l=0}^{L} c_l \hat{\pi}_Y(m - r/2 - l)/b, m = \ldots, -1, 0, 1, \ldots$

The above procedure computes the estimated density at integer multiples or integer plus half integer multiples of b, according to the parity of $L + r$, because it is convenient. However, one can compute it on any regular grid of *arbitrary* origin by a simple translation: $\hat{p}_Y(y + c) = \hat{p}_{Y-c}(y)$.

Step i) is fast, as can be seen from the case $r = 1$ and $r = 2$. For $r > 2$, the computation is more complex but is still fast if r is small. Later, we will detail this computation for the case $r = 3$ and $r = 4$.

Step ii) can be implemented as the output of a (smoothing) filter with impulse response $\{c_l\}$ and input $\{\hat{\pi}_Y(m - r/2)/b\}$. One may note that $\hat{\pi}_Y(m - r/2)/b$ is no more than the kernel density estimate at the point $(m - r/2)b$ using the kernel $\mathbb{1}^{\star r}_{[-1/2,1/2)}$ with bandwidth parameter b. Thus, $\hat{\pi}_Y(m - r/2)$ may be viewed as an estimate of the probability that Y belongs to a cell of size b centered at $(m - r/2)b$. As the kernel $\mathbb{1}^{\star r}_{[-1/2,1/2)}$ is already a good choice (see Appendix B), one may even take $L = 0$ and $c_0 = 1$, which would avoid Step ii). However, doing so would mean computing the estimated density over a grid that is equal in size to the bandwidth parameter.[1] In practice, one may need a grid size b much smaller than the "optimal" choice of h; therefore, forcing $h = b$ would result in an undersmoothed estimator, which is why Step ii) is needed to smooth it further. The amount of computations in this step is of the order LM, where M is the number of nonzero terms in the sequence $\{\hat{\pi}_Y(m - r/2)\}$, which is inversely proportional to δ, which in turn is inversely proportional to $L + 1$. Thus, the amount of calculations grows quadratically with L, but in practice, L can be taken quite low, and M is often much smaller than N. Thus, Step ii) can even be cheaper than Step i).

Note that the shape of the kernel (3) may change with L. Although this is a minor inconvenience, one would like to keep the kernel shape fixed, preferably a standard one with known good properties. This is possible if we adopt a cardinal spline as a kernel (which is a good kernel—see Appendix B) and take δ as a submultiple of 1. Indeed, from (16) in Appendix A, taking δ as the inverse of some integer m, $L = (m-1)r$ and $c_0, \ldots, c_L$ as the coefficients of the polynomial $(m^{-1} \sum_{l=0}^{m-1} z^l)^r$, the kernel (3) is identical to the centered cardinal spline of degree r. This shows that our binning technique permits the fast calculation of the cardinal spline kernel density estimator over any regular grid that is a submultiple of the bandwidth h in size.

B. Calculation of the Estimated Probability of a Cell

We now provide detailed calculation of $\hat{\pi}_Y(m - r/2)$. That this calculation is fast is due mainly to the fact that the kernel $\mathbb{1}^{\star r}_{[0,1)}$ has support a small *integer*, and the estimator is required on a regular grid with spacing b and only marginally on the simplicity of the kernel. Indeed, since $\mathbb{1}^{\star r}_{[0,1)}$ has support $(0, r]$, the term $\mathbb{1}^{\star r}_{[0,1)}[m - Y(n)/b]$ is nonzero if and only if $m = \lfloor Y(n)/b \rfloor + 1, \ldots, \lfloor Y(n)/b \rfloor + r$. Thus, the computation of $\hat{\pi}_Y(m - r/2), m \ldots, -1, 0, 1, \ldots$ can be done as follows:

i) Initialize $\hat{\pi}_Y(m - r/2) = 0, m = \ldots, -1, 0, 1, \ldots$
ii) For $n = 1, \ldots, N$, add $\mathbb{1}^{\star r}_{(0,1]}(\lfloor Y/b \rfloor + j - Y(n)/b]/N$ to $\hat{\pi}_Y(\lfloor Y/b \rfloor + j - r/2), j = 1, \ldots, r$.

In practice, one may compute first $N\hat{\pi}_Y(m - r/2), m = \ldots, -1, 0, 1, \ldots,$ and only divide them by N at the end.

In the case $r = 1$, the above procedure amounts to counting the fraction of data lying in the interval $(mb - b, mb]$. In the case $r = 2$, this amounts to spliting each data point $Y(n)$ into two neighboring grid points $b\lfloor Y(n)/b \rfloor$ and $b(\lfloor Y(n)/b \rfloor + 1)$, with weights, respectively, $\lfloor Y(n)/b \rfloor + 1 - Y(n)/b$ and $Y(n)/b - \lfloor Y(n)/b \rfloor$ and then summing up the weights at each grid point and dividing the result by N.

[1]If $\delta < 1$, the kernel $\mathbb{1}^{\star r}_{[-1/2,1/2)}$ is shrunk by δ, and hence, the bandwidth h must be inflated by the same amount to compensate it; thus, the effective bandwidth still equals b.

For $r = 3$, explicit calculations based on (15) in Appendix A yields

$$\mathbb{1}^{\star 3}_{(0,1]}(x) = \begin{cases} x^2/2, & 0 \le x \le 1 \\ 1/2 + (x-1)(2-x), & 1 \le x \le 2 \\ (3-x)^2/2, & 2 \le x \le 3 \\ 0, & \text{otherwise} \end{cases}$$

In this case, the above procedure consists of spliting each data point $Y(n)$ into three neighboring "center-grid" points $b(\lfloor Y(n)/b \rfloor - 1/2), b(\lfloor Y(n)/b \rfloor + 1/2)$, and $b(\lfloor Y(n)/b \rfloor + 3/2)$, with weights, respectively, $(1-r_n)^2/2, 1/2 + (1-r_n)r_n$, and $r_n^2/2$, where $r_n = Y(n)/b - \lfloor Y(n)/b \rfloor$, and then summing up the weights at each "center-grid" point and dividing the result by N.

For $r = 4$, explicit calculations based on (15) in Appendix A yields

$$\mathbb{1}^{\star 4}_{(0,1]}(x) = \begin{cases} (2 - |x-2|)^3/6, & 1 \le |x-2| \le 2 \\ 2/3 - (x-2)^2 + |x-2|^3/2, & |x-2| \le 1 \\ 0, & \text{otherwise} \end{cases}$$

In this case, the above procedure consists of spliting each data point $Y(n)$ into four neighboring grid points $b(\lfloor Y(n)/b \rfloor - 1), b\lfloor Y(n)/b \rfloor, b(\lfloor Y(n)/b \rfloor + 1)$ and $b(\lfloor Y(n)/b \rfloor + 2)$, with weights, respectively, $(1-r_n)^3/6, 2/3 - r_n^2 + r_n^3/2, 2/3 - (1-r_n)^2 + [1-r_n]^3/2$ and $(1-r_n)^3/6$, where $r_n = Y(n)/b - \lfloor Y(n)/b \rfloor$ and then summing up the weights at each grid point and dividing the result by N.

III. Estimation of the Entropy and Scores Function

A. *Estimation of Entropy*

A natural estimator of the entropy of a random variable Y is $-\int \hat{p}_Y(y) \log \hat{p}_Y(y)\, dy$, where $\hat{p}_Y$ is an estimator of the density p_Y of Y. However, this estimate would require integration (over the whole real line); therefore, we consider instead a discretized form of it:

$$-\sum_{m=-\infty}^{\infty} \hat{p}_Y(mb) \log \hat{p}_Y(mb) b \tag{4}$$

where b is a the discretization step. A better method would be to estimate $H(Y)$ by $N^{-1} \sum_{n=1}^{N} \log \hat{p}_Y[Y(n)]$ since it would not involve discretization, but it requires the calculation of the density estimator at arbitrary point, whereas (4) needs only such a calculation at regular grid points, and we already know how to do that with little cost. Note that the summation in (4) is actually a finite sum since our kernel *has finite support* so that $\hat{p}_Y(y) = 0$ for $|y|$ sufficiently large.

The estimator (4) suffers a minor problem in that it is not invariant with respect to translation. Adding a constant to the variable Y does not change its entropy; hence, one would prefer the estimator to be unchanged when all the data is added a same constant, but this is not true for the estimator (4). This problem is actually a consequence of the discretization since the "integral estimator" $-\int \hat{p}_Y(y) \log \hat{p}_Y(y)\, dy$ is translation invariant. For this reason, we will agree to center the data before applying the formula (4). Further, since the algorithm in Section II-A may compute the density not at integer multiples but at integer and half integer multiples of b, we will replace m in (4) by $m - (L+r)/2$. This leads to estimating $H(Y)$ by

$$\hat{H}(Y) = -\sum_{m=-\infty}^{\infty} \hat{p}_Y\left[\bar{Y} + \left(m - \frac{L+r}{2}\right) b\right] \times \log \hat{p}_Y \left[\bar{Y} + \left(m - \frac{L+r}{2}\right) b\right] b \tag{5}$$

where $\bar{Y} = (1/N) \sum_{n=1}^{N} Y(n)$ is the sample mean.

Another point is the equivariance with respect to scale change. The theoretical entropy has this property in the sense that multiplying the random variable Y by a constant c adds its entropy by $\log |c|$. Our estimator (5) also possesses the same property: It is added by $\log |c|$ if all the data is multiplied by c. To show this, we first note that $\sum_{m=-\infty}^{\infty} \hat{p}_Y[\bar{Y} + (m - L/2 - r/2)b]b = 1$ if the kernel is of the form (3). Indeed, since $\hat{p}_Y[\bar{Y} + (m - L/2 - r/2)b]b = \sum_{l=0}^{L} c_l \hat{\pi}_{Y-\bar{Y}}(m - r/2 - l)$, this equality clearly holds if the cell probabilities $\hat{\pi}_{Y-\bar{Y}}(m - r/2), m = \ldots, -1, 0, 1, \ldots$ sum to 1, but the last property is an easy consequence of the partition of unity property of the cardinal splines (see Appendix A). Therefore, one may rewrite (5) as

$$\hat{H}(Y) = -\sum_{m=-\infty}^{\infty} \hat{\hat{\pi}}_Y \left(m - \frac{L+r}{2}\right) \times \log \hat{\hat{\pi}} \left(m - \frac{L+r}{2}\right) + \log b \tag{6}$$

where $\hat{\hat{\pi}}_Y(m - L/2 - r/2) = \hat{p}_Y[\bar{Y} + (m - L/2 - r/2)b]b$. By taking, as we have said earlier, b to be a constant multiple of the bandwidth h and by requiring further, as it is natural, that h scales as the data, that is it is multiplied by $|c|$ if all the data are multiplied c, the $\hat{\hat{\pi}}_Y(m - L/2 - r/2)$ remain unchanged when one multiplies all the data by a positive constant and (since the kernel κ is symmetric) change to $\hat{\hat{\pi}}_Y(L/2 + r/2 - m)$ when one changes the sign of all the data. Therefore, when one multiplies all the data by a constant c, the first term on the right-hand side of (6) remains the same, whereas the second term is added by $\log |c|$. This yields the announced result.

The set $\{\hat{\hat{\pi}}_Y(m - L/2 - r/2)\}$ may be viewed as a discrete probability distribution providing probabilities of cells of size b centered at $\bar{Y} + (m - L/2 - r/2)b$. Thus, $\hat{H}(Y)$ appears as the entropy of a discrete distribution (which is a quantized version of the distribution of Y) plus $\log b$. This mirrors the relation between the entropy of a continuous distribution and the discrete entropy [17]. In practice, $\hat{\hat{\pi}}_Y(m - L/2 - r/2)$ will be computed directly as $\sum_{l=0}^{L-1} c_l \hat{\pi}_Y(m - r/2 - l)$, with the cell probabilities $\hat{\pi}_Y(m - r/2)$ being computed as in Section II-B.

B. *Estimation of the Score Function*

To minimize the criterion (1), we need to evaluate its relative gradient, which is defined through the linearization of $C(\mathbf{B} + \mathcal{E}\mathbf{B})$ around $\mathcal{E} = \mathbf{0}$ ($\mathcal{E}$ is a matrix) [8]. It is easy to see from (1) that this relative gradient is the matrix $\{\dot{H}(Y_i; Y_j)\}_{i,j=1}^{K} - \mathbf{I}$, where $\dot{H}(Y_i; Y_j) = \lim_{\epsilon \to 0} [H(Y_i + \epsilon Y_j) - H(Y_i)]/\epsilon$ is the gradient of $H(Y_i)$ along the "direction" Y_j. In [26], it was shown that $\dot{H}(Y; Z) = \mathrm{E}[\psi_Y(Y) Z]$, where ψ_Y is the derivative of

$-\log p_Y$. The function ψ_Y is known as the score function and can thus be viewed as the gradient of the entropy functional. Note that the relative gradient of the criterion (1) has been found in [8] to be the matrix $\{\mathrm{E}[\psi_{Y_i}(Y_i)Y_j]\}_{i,j=1}^{K} - \mathbf{I}$, which is consistent with the above result.

The above result suggests a general method for estimating the score function from an entropy estimator. Let $\hat{H}(Y) = \mathcal{H}(Y(1),\ldots,Y(n))$ be an entropy estimator of Y based on the data $Y(1),\ldots,Y(n)$, and let $Z(1),\ldots,Z(n)$ be the observations of another random variable Z. Then

$$\hat{H}(Y+\epsilon Z) = \hat{H}(Y) + \epsilon\frac{1}{N}\sum_{n=1}^{N}\hat{\psi}_Y[Y(n)]Z(n) + o(\epsilon) \quad (7)$$

where

$$\hat{\psi}_Y[Y(n)] = N\frac{\partial\mathcal{H}[Y(1),\ldots,Y(N)]}{\partial Y(n)}. \quad (8)$$

This suggests taking $\hat{\psi}_Y[Y(n)]$ as the estimated score function at $Y(n)$. Of course, (8) only defines $\hat{\psi}_Y$ at the data points, but it is often the case that this definition can be extended to cover the whole line.

The score estimator defined in this way possesses the properties

$$\frac{1}{N}\sum_{n=1}^{N}\hat{\psi}_Y[Y(n)] = 0, \quad \frac{1}{N}\sum_{n=1}^{N}\hat{\psi}_Y[Y(n)]Y(n) = 1. \quad (9)$$

These equalities come from the translation-invariance and scale-equivariance properties of the entropy estimator. Indeed, taking $Z = 1$ in (7), its left-hand side and the first term of its right-hand side must be the same, yielding the first equality. Further, take $Z = Y$ in (7) so that the last two terms in its right-hand side must sum to $\log(1+\epsilon) = \epsilon + o(\epsilon)$, yielding the second equality. Note that the theoretical score function also satisfies $\mathrm{E}\psi_Y(Y) = 0$ and $\mathrm{E}[\psi_Y(Y)Y] = 1$ (see, e.g., [8]).

C. Computation of the Score Estimator

Since our score estimator involves the derivative of the entropy estimator with respect to the data values and the last estimator depends on the bandwidth parameter h, we will need to consider the function

$$\varphi_Y[Y(n)] = N\partial\log h/\partial Y(n). \quad (10)$$

Then, a detailed calculation in Appendix C shows that (8), with the entropy estimator (6), yields

$$\hat{\psi}_Y[Y(n)] = \tilde{\psi}_Y[Y(n)] - \bar{\tilde{\psi}}_Y + (1-\lambda_Y)\varphi_Y[Y(n)] \quad (11)$$

where

$$\tilde{\psi}_Y(y) = \frac{\delta}{h}\sum_{m=-\infty}^{\infty}\kappa'\left[\left(m - \frac{L+r}{2}\right)\delta - \frac{y-\bar{Y}}{h}\right] \times \log\hat{\bar{\pi}}_Y\left(m - \frac{L+r}{2}\right) \quad (12)$$

where κ' is the derivative of κ defined in (3), and

$$\bar{\tilde{\psi}}_Y = \frac{1}{N}\sum_{l=1}^{N}\tilde{\psi}_Y[Y(n)]$$
$$\lambda_Y = \frac{1}{N}\sum_{l=1}^{N}\tilde{\psi}_Y[Y(l)][Y(l)-\bar{Y}]. \quad (13)$$

One can view $\tilde{\psi}_Y$ as a raw estimator of the score function. Indeed, viewing the sum in (12) as a Riemann sum, it is seen that $\tilde{\psi}_Y(y)$ equals approximately $\int\kappa'(u - y/h)\log\hat{p}_Y(hu)\,du/h$, which, by integration by parts and a change of integration variable, is the same as $\int\kappa[(u-y)/h](-\log\hat{p}_Y)'(u)\,du/h$. Thus, $\tilde{\psi}_Y$ is approximately a smoothed version of $(-\log\hat{p}_Y)'$, and the last function is clearly a score estimator. This estimator would, however, be undersmoothed if the bandwidth is adjusted to yield correct smoothing for the density since the score function involves a derivative that requires more smoothing. One can, of course, increase the bandwidth, but then, the denominator $\hat{p}_Y$ in $(-\log\hat{p}_Y)'$ would be oversmoothed. Our estimator $\tilde{\psi}_Y$ avoids this difficulty by introducing a second smoothing of $(-\log\hat{p}_Y)'$, but this estimator $\tilde{\psi}_Y$ does not satisfy (9). The extra terms in (11) are precisely correction terms, which make it satisfy these conditions, as φ_Y already satisfies (9) since $\log h$ is translation invariant and scale equivariant, just like $\hat{H}(Y)$. The correction terms would be small since $\tilde{\psi}_Y$, because it it a score estimator, should satisfy approximately $\bar{\tilde{\psi}}_Y = 0$ and $\lambda_Y = 1$. Note that the fact that $\bar{\tilde{\psi}}_Y$ is not exactly zero is due to the discretization that destroys the translation invariance of the entropy estimator. Indeed, its limit as $\delta \to 0$, which is $\int\kappa[(u-y)/h](-\log\hat{p}_Y)'(u)\,du/h$, satisfies the "centering" condition

$$\frac{1}{N}\sum_{n=1}^{N}\int\frac{1}{h}\kappa\left[\frac{u-Y(n)}{h}\right](-\log\hat{p}_Y)'(u)\,du = -\int\hat{p}_Y'(u)\,du = 0.$$

The choice of h (as function of the data) affects the second correction term in (11). We do not need a sophisticated choice since, as it has been seen, the score estimator does not need be very accurate. (The performance of the separation procedure depends only on this estimator and not directly on the entropy estimator since the gradient of the criterion is expressed in term of this estimator only.) Thus, we adopt the simple Silverman rule [19]. We take h to be the optimal bandwidth (as derived in Appendix B) for a Gaussian density. Direct calculation shows that it equals $1.06N^{-1/5}\sigma$ for a Gaussian kernel, where σ is the standard deviation [19]. For a cardinal spline kernel, the above bandwidth must be multiplied by the relative bandwidth, given in the last row of Table I in Appendix A. Taking h proportional to the sample standard deviation $\hat{\sigma}_Y = \{N^{-1}\sum_{n=1}^{N}[Y(n) - \bar{Y}]^2\}^{1/2}$ leads to $\varphi_Y(y) = (y-\bar{Y})/\hat{\sigma}_Y^2$, which is none other than the score function of a Gaussian variable.

For a kernel of the form (3), (12) reduces to

$$\tilde{\psi}_Y(y) = \frac{1}{b}\sum_{l=0}^{L}\sum_{m=-\infty}^{\infty} c_l \mathbb{1}_{(0,1]}^{\star r\,\prime}\left(m - \frac{y-\bar{Y}}{b}\right) \times \log\hat{\bar{\pi}}_Y\left(m + l - \frac{L+r}{2}\right)$$

where $\mathbb{1}_{(0,1]}^{\star r\,\prime}$ is the derivative of $\mathbb{1}_{(0,1]}^{\star r}$. By a change of order of summation, this yields

$$\tilde{\psi}_Y(y) = \frac{1}{b}\sum_{m=-\infty}^{\infty} \mathbb{1}_{(0,1]}^{\star r\,\prime}\left(m - \frac{y-\bar{Y}}{b}\right)\widehat{\log\pi}_Y\left(m - \frac{r}{2}\right)$$

where

$$\widehat{\log\pi}_Y\left(m - \frac{r}{2}\right) = \sum_{l=0}^{L} c_l \log\left[\sum_{\ell=0}^{L} c_\ell \hat{\pi}_Y\left(m - \frac{r}{2} + l - \ell\right)\right].$$

For $r = 3, \mathbb{1}_{(0,1]}^{\star r\,\prime}(u) = u$ if $0 \le u \le 1, 3 - 2u$ if $1 \le u \le 2$, and $u - 3$ if $2 \le u \le 3$. Therefore

$$\tilde{\psi}_Y[\bar{Y} + (j+u)b] = \frac{1}{b}\left\{(1-u)\widehat{\log\pi}_Y\left(j - \frac{1}{2}\right) + (2u-1)\widehat{\log\pi}_Y\left(j + \frac{1}{2}\right) - u\widehat{\log\pi}_Y\left(j + \frac{3}{2}\right)\right\}$$

for any signed integer j and $0 \le u < 1$; hence, $\tilde{\psi}_Y$ is the linear interpolation of the function taking the value $[\widehat{\log\pi}_Y(j - (1/2)) - \widehat{\log\pi}_Y(j + (1/2))]/b$ at the grid point $\bar{Y} + jb$.

We have noted that the score function does not need to be very accurately estimated. *This allows us to work with a coarse grid, corresponding to* $L = 0$. The main effect of using such a coarse grid is that $\tilde{\psi}_Y$ is linearly interpolated from a sparse set of points, entailing some loss of accuracy. However, in our experience, this loss is dominated by the bias and variability of the score estimator. Taking $L = 0$ avoids Step ii) in the algorithm of Section II-A and the computation of $\widehat{\log\pi}_Y$, which then just reduces to $\log\hat{\pi}_Y$.

IV. Separation Algorithms

A. Offline Algorithm

The minimization of the criterion (1) can be done through a gradient descent algorithm, but a much faster method is the Newton algorithm (which amounts to using the natural gradient [13]). Staring with a current estimator $\hat{\mathbf{B}}$, it consists of expanding $\hat{C}(\hat{\mathbf{B}} + \mathcal{E}\hat{\mathbf{B}})$ with respect to $\mathcal{E}$ up to second order and then minimizing the resulting quadratic form in $\mathcal{E}$ to obtain a new estimate. This method requires the Hessian[2] of $\hat{C}(\hat{\mathbf{B}} + \mathcal{E}\hat{\mathbf{B}})$ with respect to $\mathcal{E}$ (that is the relative Hessian of $\hat{C}$), which is quite involved. For this reason, we will approximate it by the relative Hessian of C, computed under the assumption that the Y_i are independent. (The method is then referred to as quasi-Newton.) It has been shown in [8] that this approximate relative Hessian is defined by the quadratic form

$$\mathcal{E} \mapsto \frac{1}{2}\sum_{1\le i\ne j\le K}\left[\mathrm{E}\psi_{Y_i}^2(Y_i)\,\mathrm{var}(Y_j)\mathcal{E}_{ij}^2 + \mathcal{E}_{ij}\mathcal{E}_{ji}\right]$$

The term $\mathrm{E}\psi_{Y_i}^2(Y_i)$ is none other than the Fisher information J_{Y_i} of Y_i, and since it is unknown, we replace it by $\hat{J}_{Y_i} = N^{-1}\sum_{n=1}^{N}\hat{\psi}_{Y_i}^2[Y_i(n)]$. Likewise, $\mathrm{var}(Y_j)$ is replaced by $\hat{\sigma}_{Y_j}^2$, which is the sample variance of Y_j. It is important to note that $\hat{J}_{Y_i}\hat{\sigma}_{Y_i}^2 \ge 1$ by (9) and the Schwartz inequality. This shows that the approximate Hessian is positive definite, which ensures the stability of the iterative algorithm. Explicitly, the iteration consists of computing

$$\begin{bmatrix}\mathcal{E}_{ij}\\ \mathcal{E}_{ji}\end{bmatrix} = \begin{bmatrix} J_{Y_j}\hat{\sigma}_{Y_i}^2 & 1\\ 1 & J_{Y_i}\hat{\sigma}_{Y_j}^2\end{bmatrix}^{-1}\frac{1}{N}\sum_{n=1}^{N}\begin{bmatrix}\hat{\psi}_{Y_i}[Y_i(n)]Y_j(n)\\ \hat{\psi}_{Y_j}[Y_j(n)]Y_i(n)\end{bmatrix}$$

($\mathcal{E}_{ii} = 0$) and then updating the estimator $\hat{\mathbf{B}}$ by subtracting $\mathcal{E}\hat{\mathbf{B}}$ from it.

Note that from (11) and with φ_Y being the Gaussian score function, one can rewrite $\hat{J}_Y$ as

$$\tilde{J}_Y + \frac{1-\lambda_Y}{\hat{\sigma}_Y^2}\left\{1 - \lambda_Y + \frac{2}{N}\sum_{n=1}^{N}\tilde{\psi}_Y[Y(n)][Y(n) - \bar{Y}]\right\}$$

where $\tilde{J}_Y = N^{-1}\sum_{n=1}^{N}\{\tilde{\psi}_Y[Y(n)] - \bar{\tilde{\psi}}_Y\}^2$. Therefore, from (13)

$$\hat{J}_Y = \tilde{J}_Y + \frac{1-\lambda_Y^2}{\hat{\sigma}_Y^2}. \quad (14)$$

The last formula will be useful later. It shows again that $\hat{J}_Y\hat{\sigma}_Y^2 \ge 1$ since $\hat{\sigma}_Y^2\tilde{J}_Y \ge \lambda_Y^2$ by (13) and the Schwartz inequality.

B. Online Algorithm

The above offline algorithm can be adapted to obtain an online version. The main ingredient of the algorithm is the calculation of the score function (8), which can be seen to be a linear combination of known functions with coefficients depending on the data. Thus, one only needs to update these coefficients, which amounts to essentially updating the probabilities of the grid cells. Note that in online processing, one often scales the reconstructed sources so that they have unit variance. The advantage is that the bandwidth parameter h (and hence the grid size) is now *constant in time*. The parameter h must be related to the learning step ρ since this coefficient determines the effective sample size, which is about $(2-\rho)/\rho$. Therefore, applying the simple Silverman rule for choosing the bandwidth, it should be $1.9898 \times 1.06[\rho/(2-\rho)]^{1/5} = 2.11[\rho/(2-\rho)]^{1/5}$ for the third cardinal spline kernel (see Table I in Appendix A).

In developing the online algorithm, it is important to ensure that the approximate Hessian is positive definite, since this is necessary for the stability of the algorithm. This can be achieved by making sure that $\hat{J}_Y\hat{\sigma}_Y^2 > 1$. Thus, the online analog of the formula $\hat{J}_Y = N^{-1}\sum_{n=1}^{N}\hat{\psi}_Y^2[Y(n)]$ cannot be used, since

[2]The Hessian of a function of several variables is the matrix of its partial derivatives.

then, the above condition cannot be guaranteed to hold as one can no longer apply the Schwartz inequality.[3] However, estimating the Fisher information through the online analog of (14) does lead to an estimator satisfying the above condition, provided that the online version of $\lambda_Y, \hat{\sigma}^2_Y$ and $\tilde{J}_Y$ are defined appropriately.

As the online algorithm is rather heavy, we limit ourselves to the simple case $r = 3$ and $L = 0$. We have already said that one can very well choose $L = 0$, and there is little difference between $r = 3$ and $r = 4$, as can be seen in Appendix B. The updating equations are given below, using the same notations as before, but adding an extra time index n.

1) Reconstructing the centered sources: $\mathbf{Y}(n) = \mathbf{B}_{n-1}[\mathbf{X}(n) - \bar{\mathbf{X}}_n]$, where $\bar{\mathbf{X}}_n = (1-\rho)\bar{\mathbf{X}}_{n-1} + \rho\mathbf{X}(n)$. Put $i_{k,n} = \lfloor Y_k(n)/b \rfloor, r_{k,n} = Y_k(n)/b - i_{k,n}$.

2) Updating the "cell probabilities"

$$\begin{aligned}\hat{\pi}_{Y_k,n}\left(i_{k,n} - \frac{1}{2}\right) &= (1-\rho)\hat{\pi}_{Y_k,n-1}\left(i_{k,n} - \frac{1}{2}\right) + \rho\frac{(1-r_{k,n})^2}{2}\\ \hat{\pi}_{Y_k,n}\left(i_{k,n} + \frac{1}{2}\right) &= (1-\rho)\hat{\pi}_{Y_k,n-1}\left(i_{k,n} + \frac{1}{2}\right) + \rho\left[\frac{1}{2} + (1-r_{k,n})r_{k,n}\right]\\ \hat{\pi}_{Y_k,n}\left(i_{k,n} + \frac{3}{2}\right) &= (1-\rho)\hat{\pi}_{Y_k,n-1}\left(i_{k,n} + \frac{3}{2}\right) + \frac{\rho r_{k,n}^2}{2}\\ \hat{\pi}_{Y_k,n}\left(i_{k,n} + \frac{1}{2} + j\right) &= (1-\rho)\hat{\pi}_{Y_k,n-1}\left(i_{k,n} + \frac{1}{2} + j\right) \quad |j| > 1.\end{aligned}$$

3) Computing the centered raw scores and the λ coefficients:

$$\begin{aligned}\tilde{\psi}_{Y_k}[Y_k(n)] &= \frac{1}{b}\Big[(1-r_{k,n})\log\hat{\pi}_{Y_k}\left(i_{k,n} - \frac{1}{2}\right) + (2r_{k,n} - 1)\log\hat{\pi}_{Y_k}\left(i_{k,n} + \frac{1}{2}\right) - r_{k,n}\log\pi_{Y_k}\left(i_{k,n} + \frac{3}{2}\right)\Big]\\ \bar{\tilde{\psi}}_{Y_k,n} &= (1-\rho)\bar{\tilde{\psi}}_{Y_k,n-1} + \rho\tilde{\psi}_{Y_k}[Y_k(n)]\\ \tilde{\psi}^c_{Y_k}[Y_k(n)] &\overset{\text{def}}{=} \tilde{\psi}_{Y_k}[Y_k(n)] - \bar{\tilde{\psi}}_{Y_k,n}\\ \lambda_{Y_k,n} &= (1-\rho)\lambda_{Y_k,n-1} + \rho\tilde{\psi}^c_{Y_k}[Y_k(n)]Y_k(n).\end{aligned}$$

4) Computing the Fisher information and the scores:

$$\begin{aligned}\tilde{J}_{Y_k,n} &= \left[1 - \rho + \rho Y_k^2(n)\right] \times \left\{(1-\rho)\tilde{J}_{Y_k,n-1} + \rho\tilde{\psi}^c_{Y_k}[Y_k(n)]^2\right\}\\ \hat{J}_{Y_k,n} &= \tilde{J}_{Y_k,n} + 1 - \lambda^2_{Y_k,n}.\\ \hat{\psi}_{Y_k}[Y_k(n)] &= \tilde{\psi}^c_{Y_k}[Y_k(n)] + (1-\lambda_{Y_k,n})\,Y_k(n)\end{aligned}$$

[3]In practice, we found that it often does not hold.

5) Updating the separation matrix: $\mathbf{B}_{n+1} = (\mathbf{I} - \mathcal{E})\mathbf{B}_n$, where $\mathcal{E}$ is the matrix with diagonal elements $\mathcal{E}_{ii} = \rho[Y_i^2(n) - 1]/2$ and off diagonal elements

$$\mathcal{E}_{ij} = \rho\frac{\hat{J}_{Y_j,n}\hat{\psi}_{Y_i}[Y_i(n)]Y_j(n) - \hat{\psi}_{Y_j}[Y_j(n)]Y_i(n)}{\hat{J}_{Y_i,n}\hat{J}_{Y_j,n} - 1}.$$

Some explanations and comments regarding the algorithm are now given.

- *Scaling and information matrix*: Assume that the variance estimate of Y_k based on its observations up to time $n-1$ is 1; then, the next estimate would be $s_k^2 = 1 - \rho + \rho Y_k^2(n)$. Thus, to keep the variance Y_k at the level 1, one has to scale down their observations by s_k. This has the effect of multiplying the Fisher information of Y_k by s_k^2, which explains the presence of the factor s_k^2 in the formula for $\tilde{J}_{Y_k,n}$ and the absence of the variance term in the formulas for $J_{Y_k,n}$ and $\mathcal{E}_{ij}$. Note that the rescaling is approximatively implement through the diagonal term $\mathcal{E}_{ii}$ of $\mathcal{E}$ since $1/s_k \approx 1 - \rho[Y_k^2(n) - 1]/2$. By taking correctly into account the scaling effect as above, one ensures that $\hat{J}_{Y_k,n} > 1$. Indeed, the coefficient $\lambda_{Y_k,n}$ can be viewed as the scalar product of the vectors $[1 \;\; Y_k(n)]$ and $[\lambda_{Y_k,n-1} \;\; \tilde{\psi}^c_{Y_k}[Y_k(n)]]$, with weights $1-\rho$ and ρ attached to their components; hence by the Schwartz inequality, $\lambda^2_{Y_k,n} \le s_k^2\{(1-\rho)\lambda^2_{Y_k,n-1} + \rho\tilde{\psi}^c_{Y_k}[Y_k(n)]^2\}$. Thus, $\lambda^2_{Y_k,n-1} < \tilde{J}_{Y_k,n-1}$ implies $\lambda^2_{Y_k,n} < \tilde{J}_{Y_k,n}$, implying that $\hat{J}_{Y_k,n} > 1$ if one starts the algorithm with $\tilde{J}_{Y_k,0} > \lambda^2_{Y_k,0}$. In practice, one takes $\tilde{J}_{Y_k,0} \gg \lambda^2_{Y_k,0} = 1$ to get a high value of $\hat{J}_{Y_k}$ at the beginning of the algorithm, which would make the approximate Hessian higher and closer to diagonal. As the estimate of the separation matrix is far from the truth at this stage, it makes sense not to use of the true Hessian, but rather a nearly diagonal matrix. Further, a higher Hessian matrix would make the step size smaller at this early stage.
- *Number of grid cells*: Unlike the offline algorithm, the number of grid cells that have nonzero probability would increase indefinitely in time, except for bounded sources, although only a finite number of cells would have non-negligible probabilities. We prefer to work with a fixed number of cells, $2M + 2$ say. This means that we have to deal with the case where a reconstructed source (which has unit variance) lies outside the interval $[-Mb, Mb]$ covered by the interior cells. Our solution is to pull it inside to Mb or $-Mb$, according to their sign. By taking M high enough, the above case can be made to happen with negligible probability. For example, for a standard Gaussian variable, the probability that a datum falls outside $[-3.1, 3.1]$ is only 0.002. For a long tail variable such as the bilateral exponential, the range must be double $[-6.2146, 6.2146]$ to achieve the same probability, but this is an extreme case. For short tail variables, the range would be shorter than that of the Gaussian. Thus, a choice of M for which $Mb \ge 6$ is a safe bet that most of the data fall in the interior cells.

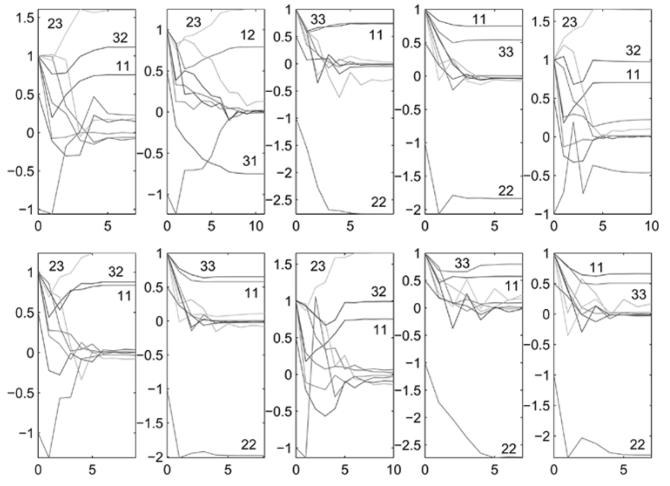
Fig. 1. Nine elements of the global matrix in the offline algorithm versus the iteration step, with indexes of the three nonzero elements shown; each subplot corresponds to a segment of length 200 of the data sequence.

- *Centering*: If the data are *known* to have zero mean, one can drop their centering. In addition, the average raw scores are often very close to 0; therefore, one may *drop the centering of these scores* as well.

V. Two Simulation Examples

To illustrate the performance of our algorithm, we generate three independent sequences of sources of length 2000 with distributions the uniform, the triangular, and the bilateral exponential distribution. The first two are bounded distributions with score function tending to infinity at the boundary of their support, whereas the last is a long tail distribution with bounded score function. The sources are scaled to have unit variance and mixed according to the mixing matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 0.5 & 1 & 1 \end{bmatrix}.$$

In a first simulation, we apply our offline algorithm to ten consecutive segments of length 200 of the data sequences. We take $r = 3$ and $L = 0$ so that the grid size equals the bandwidth. Fig. 1 plots the nine elements of the global matrix $\mathbf{BA}$ versus the iteration step. One can see that the algorithm converges in at most ten steps. Note that in applying the algorithms to consecutive segments, one could (and should) initialize the algorithm by the separating matrix just found in previous segment, and then, the algorithm would converge in one step or two. Here, we always initialize the algorithm by the identity matrix to see how fast it could converge. We also have applied our algorithm with $L = 1$: The results are very similar and are not shown. This suggests that we do not really needed $L > 0$, which justifies our decision to restrict ourselves to the choice $L = 0$ in the online algorithm (for resoasons of simplicity).

In a second simulation, we applied the online algorithm to the whole data sequence. Again, the separating matrix is initialized at the identity matrix. However, to initialize the scale and the probabilities, we make use of the first 50 samples, where the separation matrix is kept fixed and only scaled during the first

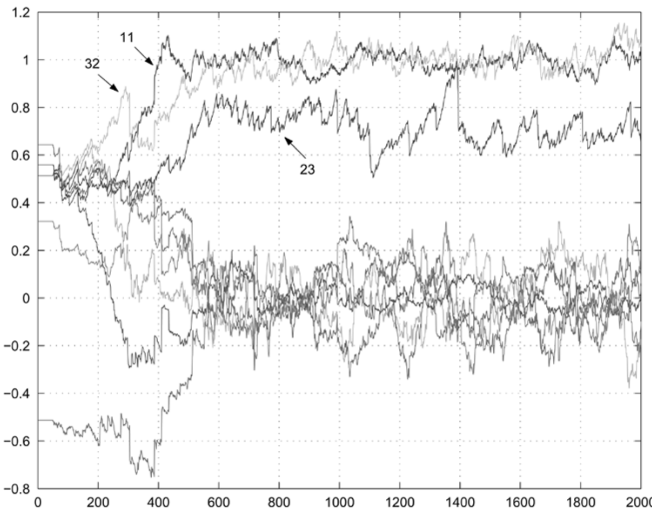

Fig. 2. Evolution of the nine elements of the global matrix in the online algorithm. The arrows indicate the indexes of the three nonzero ones.

50 samples. The full online algorithm only starts at sample 51. (Thus, this initialization is based on the mixtures of sources, since we have yet to separate them.) The leaning step ρ is set to 0.01. The evolution of the nine elements of the global matrix is plotted in Fig. 2.

As can be seen, both algorithms perform satisfactorily. Its convergence, in terms of the number of iterations, is quite fast. Our simulations, however, do not tell how costly the computation is, and a comparison of computation time with other existing algorithms could be of interest. However, this is not easy, since we do not have the codes of the other algorithms, and one needs to make comparison on the same machine, using the same programming language (with the same coding efficiently). Further, our algorithm computes both the criterion and its gradient and does not merely try to solve the estimating equation; it also involves the Hessian, which increases the cost for one iteration but reduces the number of iterations for convergence. Therefores we will choose to look at the complexity of the algorithm by counting the number of operations (multiplication or division) needed for each iteration.

In the case of the offline algorithm, one needs first to compute the probabilities of the cells, which requires $6N$ operations (N being the sample size) with the quadratic spline kernel $(r = 2)$. If $L > 0$, further LM operations are needed to smooth out these probabilities (M being the number of cells with nonzero probabilities), but this number is in general negligible with respect to N (as L is quite small). To compute the entropy, M operations and M evaluations of the logarithmic function will be needed. For the computation of the raw score function, we need $3N$ operations by taking $L = 0$. By taking $L > 0$, ML operations are needed to compute the quantities $\widehat{\log \pi_Y}(m - r/2)$ (not counting ML operations and M evaluations of the logarithmic function already required above). Thus, in total, one needs about $10N$ operations per source. For K sources, one should multiply this number by K, and one also needs $K(K-1)N$ operations to compute the elements of the gradient vector. The total number of operation is quite reasonable. A method based on the fourth-order cumulants, for example, can require a much smaller number of operations to compute the criterion, but one also needs to compute the gradient of the criterion that would involve cross cumulants, the number of which grows at least quadratically with K. In addition, we have not considered the computation of the Hessian, which is quite cheap in our algorithm, but not in other methods. This is because our (approximate) Hessian has the block diagonal form with 2×2 diagonal blocks; therefore, we need only to compute the estimated variance and the Fisher information for each source, which needs N operations each.

We now look at the complexity of the online algorithm. The costliest part of it is the update of the cell probabilities in step 2, which takes $(2M+1)+5$ operations (multiplication or division), $(2M+1)$ being the number of cells. With $M \geq 6/b$ and $b = 2.11[\rho/(2-\rho)]^{1/5}$, one gets $M \geq 9$ for $\rho = 0.01$ and $M \geq 13$ for $\rho = 0.001$. Thus, M is fairly small. The computation of the raw score in step 3 is quite quick. It needs three multiplications and three evaluations of the logarithm function. To compute the Fisher information and the score, Step 4) needs four further operations. We have yet to include centering, which needs some more operations, but as has been said, this can be dropped. Note that the above counts are per source; if there are K sources, one has to multiply them by K. Finally, we have not counted the number of operations to reconstruct the sources and update the separating matrix, since it is the same for all algorithms of the quasi-Newton type.

VI. Conclusion

We have provided low-cost algorithms to perform ICA through the minimization the mutual information criterion with the true unknown density of the source being nonparametrically estimated. Further, we introduce a new estimator of the score function (10)–(12), which allows the exact calculation of the (relative) gradient of the estimated criterion [obtained from (1) by replacing the entropies by their estimates]. If we just use *some* other score estimator in the formula for the gradient, then we get only an estimate of the relative gradient of the theoretical criterion C, which may not be the same. Having the exact gradient of the estimated criterion is necessary to minimize it correctly.

Appendix

A. Some Properties of the Cardinal Spline

The cardinal spline of degree r is the r times convolution of $\mathbb{1}_{(0,1]}$ (the indicator function of the interval $(0,1]$) with itself, which we denote by $\mathbb{1}_{(0,1]}^{\star r}$. A detailed treatment of these splines can be found in [25]. We give here only some of their useful properties.

- Cardinal splines are density functions
- The cardinal spline of degree r has support $[0,r]$ and is symmetric around it center of support: $\mathbb{1}_{(0,1]}^{\star r}(u) = \mathbb{1}_{(0,1]}^{\star r}(r-u)$
- Cardinal splines possess the property of partition of unity: $\sum_{m=-\infty}^{\infty} \mathbb{1}_{(0,1]}^{\star r}(u+m) = 1$ for all u.
- Cardinal splines satisfy the following recursion:

$$\mathbb{1}_{(0,1]}^{\star r}(u) = \frac{u}{r-1}\mathbb{1}_{(0,1]}^{\star (r-1)}(u) + \frac{r-u}{r-1}\mathbb{1}_{(0,1]}^{\star (r-1)}(u-1). \quad (15)$$

- For any positive integer m

$$\mathbb{1}_{(0,1]}^{\star r}(u) = \sum_{l=0}^{(m-1)r} c_l^{m,r} m \mathbb{1}_{(0,1]}^{\star r}(mu-l) \quad (16)$$

  where $c_l^{m,r}, l = 0, \ldots, (m-1)r$ are the coefficients of the polynomial $(m^{-1}\sum_{l=0}^{m-1} z^l)^r$. (This relation can be proved by noting that $\mathbb{1}_{(0,1]}(u) = m^{-1}\sum_{l=0}^{m-1} B^l(m\mathbb{1}_{(0,1]})(mu)$, where B is a backward shift operator $Bf(x) = f(x-1)$ and that the shift operator commutes with the convolution.)

B. Performance of the Cardinal Spline Kernel Density Estimator

We show here that this estimator has an efficiency close to optimal and approximates the Gaussian kernel density estimator quite closely. The last estimator has some appeal, as explained in [8], but has infinite support and, hence, cannot be used in our algorithms.

It is known [19] that the kernel density estimator (2) has approximate bias $(h^2/2)p_Y''(y)\int u^2\kappa(u)\,du$ and variance $(Nh)^{-1}p_Y(y)\int \kappa^2(u)\,du$. Thus, the integrated mean square error

$$\frac{h^4}{4}\int p_Y''^2(y)\,dy \left[\int u^2\kappa(u)\,du\right]^2 + \frac{1}{Nh}\int \kappa^2(u)\,du$$

is minimized when h equals

$$h_{\text{opt}} = \left\{\frac{\int \kappa^2(u)\,du}{\left[\int u^2\kappa(u)\,du\right]^2 \int p_Y''^2(y)\,dy}\right\}^{1/5} N^{-1/5}$$

and the attained minimum is

$$\frac{5}{4}\left[\int \kappa^2(u)\,du\right]^{4/5}\left[\int u^2\kappa(u)\,du\right]^{2/5} \times \left[\int p_Y''^2(y)\,dy\right]^{1/5} N^{-4/5}.$$

The above formulas, however, require the knowledge of the density p_Y, but they can still be used to compare kernels. The ratio of the optimal bandwidth h_{opt} of the kernel κ, to that of a reference kernel κ^*, which is denoted by h_{opt}^*, is

$$\frac{h_{\text{opt}}}{h_{\text{opt}}^*} = \left\{\frac{\int \kappa^2(u)\,du}{\left[\int u^2\kappa(u)\,du\right]^2}\frac{\left[\int u^2\kappa^*(u)\,du\right]^2}{\int \kappa^{*2}(u)\,du}\right\}^{1/5}$$

TABLE I
KERNEL CHARACTERISTICS AND THEIR EFFICIENCIES AND ASSOCIATED OPTIMAL BANDWIDTH RELATIVE TO THAT OF THE GAUSSIAN KERNEL

Kernel κ	$\mathbb{1}^2_{(-.5,.5)}$	$\mathbb{1}^3_{(-.5,.5)}$	$\mathbb{1}^4_{(-.5,.5)}$	Epane.	Gauss.
$\int u^2\kappa(u)du$	1/6	1/4	1/3	1/5	1
$\int \kappa^2(u)du$	2/3	11/20	151/315	3/5	$1/(2\sqrt{\pi})$
Efficiency	.9887	.9805	.9755	1	.9608
bandwidth	2.4320	1.9898	1.7255	2.2138	1

and the *inverse* of the ratio of the corresponding minimum integrated mean squares errors is

$$\left\{\frac{\left[\int \kappa^{*2}(u)\,du\right]^2 \int u^2\kappa^*(u)\,du}{\left[\int \kappa^2(u)\,du\right]^2 \int u^2\kappa(u)\,du}\right\}^{2/5}$$

and is called the efficiency of κ relative to κ^*.

The optimal choice of κ is the one that minimizes $[\int \kappa^2(u)\,du]^2[\int u^2\kappa(u)\,du]$, and it has been shown that it is the Epanechnikov kernel, which is given by $\kappa(u) = (3/4)(1-u^2)$ for $|u| \le 1, = 0$, otherwise. The relative efficiency of a kernel to this kernel is simply called efficiency.

Direct calculation yields Table I, which provides the efficiencies for the cardinal spline kernels and their associated optimal bandwidth relative to that of the Gaussian kernel. The efficiencies are quite close to 1; therefore, it makes little difference which one of the above kernels is used. There is, in fact, some other reason, which is explained in [8], to prefer the Gaussian kernel (which has lowest efficiency). The somewhat lower efficiency of the Gaussian kernel (compared with that of the cardinal splines) may be due to the fact that it is very smooth, admitting derivatives of all order. However, we do not really need that since the smoothness of $\mathbb{1}^3_{(-0.5,0.5]}$ or $\mathbb{1}^4_{(-0.5,0.5]}$ could be sufficient. The Gaussian kernel can be viewed as a limiting case of the cardinal spline kernel, since by the central limit theorem, $\mathbb{1}^{\star r}_{[-1/2,1/2)}$, appropriately scaled, converges as $r \to \infty$ to the Gaussian density. Fig. 3 shows that this limit is almost attained for the cardinal spline of order as low as 3 and 4, which further justifies their use.

C. Explicit Calculation of the Score Estimator

From (8) and using our entropy estimator (6), we get

$$\hat{\psi}_Y[Y(n)] = -N \sum_{m=-\infty}^{\infty} \frac{\partial \hat{\hat{\pi}}_Y(m - L/2 - r/2)}{\partial Y(n)} \times \left[\log \hat{\hat{\pi}}_Y\left(m - \frac{L+r}{2}\right) + 1\right] + N\frac{\partial \log b}{\partial Y(n)}.$$

We first note that in the above left-hand side, we can drop the constant 1 in $\log \hat{\hat{\pi}}_Y(m - L/2 - r/2) + 1$ since $\sum_{m=-\infty}^{\infty} \hat{\hat{\pi}}_Y(m - L/2 - r/2) = 1$, regardless of the data values, and hence, its partial derivative with respect to $Y(n)$ vanishes. Further, since $\hat{\hat{\pi}}_Y(m - L/2 - r/2) = \hat{p}_Y[\bar{Y} + (m - L/2 - r/2)\delta h]\delta h$, we get from (2)

$$N\frac{\partial \hat{\hat{\pi}}_Y(m - L/2 - r/2)}{\partial Y(n)} = \delta \sum_{l=1}^{N} \kappa'\left[\left(m - \frac{L+r}{2}\right)\delta - \frac{Y(l) - \bar{Y}}{h}\right] \frac{\partial[\bar{Y} - Y(l)]/h}{\partial Y(n)}$$

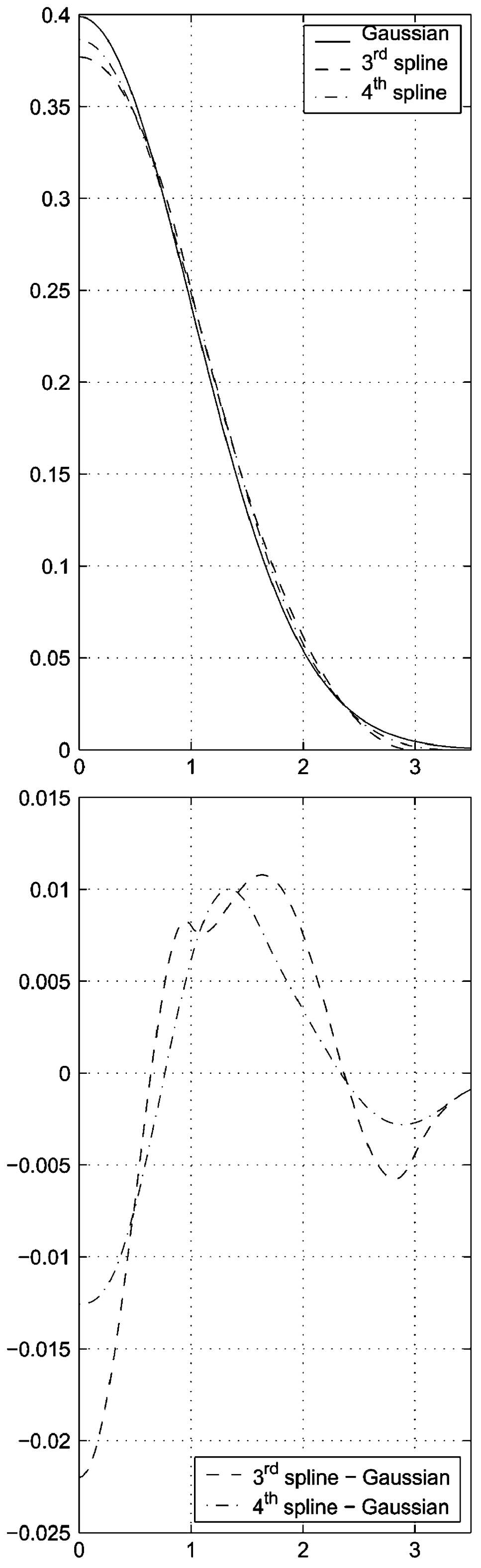

Fig. 3. Comparison between Gaussian kernel and the third and fourth scaled cardinal splines with the Gaussian kernel, scaled to have the same optimal bandwidth.

where κ' is the derivative of κ. However, using (10)

$$\frac{\partial[\bar{Y} - Y(l)]/h}{\partial Y(n)} = \frac{1}{h}\left\{\frac{1 + [Y(l) - \bar{Y}]\varphi_Y[Y(n)]}{N} - \delta_{l,n}\right\}$$

where $\delta_{l,n}$ denotes the Kronecker symbol. Therefore, combining the above results and (10)

$$\hat{\psi}_Y[Y(n)] = \tilde{\psi}_Y[Y(n)] - \sum_{l=1}^{N} \frac{1 + [Y(l) - \bar{Y}]\varphi_Y[Y(n)]}{N} \tilde{\psi}_Y[Y(l)] + \varphi_Y[Y(n)]$$

where $\tilde{\psi}_Y$ is given in (12). After some rearrangement, this yields (11).

References

[1] A. J. Bell and T. J. Sejnowski, "An Information-maximization approach to blind separation and blind deconvolution," *Neural Comput.*, vol. 7, no. 6, pp. 1129–1159, 1995.

[2] P. Comon, "Independent component analysis—A new concept?," *Signal Process.*, vol. 36, pp. 287–314, 1994.

[3] K. Torkkola, "Blind separation of convolved sources based on information maximization," in *Proc. IEEE Workshop Neural Networks Signal Process.*, Kyoto, Japan, 1996, pp. 423–432.

[4] H. H. Yang and S.-I. Amari, "Adaptive online learning algorithms for blind separation: Maximum entropy and minimum mutual information," *Neural Comput.*, vol. 9, no. 7, pp. 1457–1482, 1997.

[5] M. J. T. Alphay, D. I. Laurenson, and A. F. Murray, "Improvements in the online performance of information-maximization-based signal separation," in *Proc. ICA Conf.*, J. F. Cardoso, C. Jutten, and Ph. Loubaton, Eds., Aussois, France, Jan. 1999, pp. 49–54.

[6] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, "Separating convolutive mixtures by mutual information minimization," in *Proc. IWANN*, Granada, Spain, 2001.

[7] K. E. Hill, D. Erdogmus, and J. Príncipe, "Blind source separation using Renyi's mutual information," *IEEE Signal Processing Lett.*, vol. 8, pp. 174–176, June 2001.

[8] D. T. Pham, "Blind separation of instantaneous mixture of sources via an independent component analysis," *IEEE Trans. Signal Processing*, vol. 44, pp. 2768–2779, Nov. 1996.

[9] M. Babaie-Zadeh, C. Jutten, and K. Nayebi, Differential of mutual information function, in IEEE Signal Processing Letters, 2002, May 2002. Submitted in.

[10] S.-I. Amari, A. Cichocki, and H. H. Yang, "A new learning algorithm for blind source separation," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1996, vol. 8, pp. 757–763.

[11] S.-I. Amari, "Gradient learning in structure parameter space: Adaptive blind separation of signal sources," in *Proc. World Congr. Neural Networks*, 1996.

[12] ——, "Neural learning in structured parameter spaces: Natural Riemannian gradient," *Neural Inform. Process. Syst.—Natural Synthetic*, 1996.

[13] ——, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, no. 2, pp. 251–276, 1998.

[14] L. Zhang, A. Cichocki, and S. Amari, "Natural gradient approach for blind separation of over- and under-complete mixtures," in *Proc. ICA Conf.*, J. F. Cardoso, C. Jutten, and Ph. Loubaton, Eds., Aussois, France, Jan. 1999, pp. 455–460.

[15] ——, "Natural gradient algorithm for blind separation of overdetermined mixture with additive noise," *IEEE Signal Processing Lett.*, vol. 6, pp. 293–295, Nov. 1999.

[16] J.-F. Cardoso, "Blind signal separation: Statistical principles," *Proc. IEEE*, vol. 9, pp. 2009–2025, Oct. 1998.

[17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[18] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.

[19] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Chapman and Hall, 1982.

[20] D. W. Scott and S. J. Sheather, "Kernel density estimation with binned data," *Commun. Statist.—Theory Meth.*, vol. 14, pp. 1353–1359, 1985.

[21] M. C. Jones, "Discretized and interpolated kernel density estimates," *Amer. Statist. Assoc.*, vol. 84, pp. 733–741, 1989.

[22] J. Fan and J. S. Marron, "Fast implementation of non parametric curve estimators," *J. Comput. Graphical Statist.*, vol. 3, pp. 35–56, 1994.

[23] P. Hall and M. P. Wand, "On the accuracy of binned kernel density estimators," *J. Multivariate Anal.*, vol. 56, pp. 165–184, 1996.

[24] D. T. Pham, "On the discretization error in the computation of the empirical characteristic function," *J. Statist. Comput. Simulations*, vol. 53, pp. 129–141, 1995.

[25] C. K. Chui, *An Introduction to Wavelets*. San Diego, CA: Academic, 1992.

[26] D. T. Pham, "Mutual information approach to blind separation of stationary sources," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1935–1946, July 2002.

Dinh-Tuan Pham (M'88) was born in Hanoï, VietNam, on February 10, 1945. He graduated from the School of Applied Mathematics and Computer Sccience (ENSIMAG), Polytechnic Institute of Grenoble, Grenoble, France, in 1968. He received the Ph.D. degree in statistics from the University of Grenoble in 1975.

He was a Postdoctoral Fellow with the Department of Statistics, University of California at Berkeley, from 1977 to 1978 and a Visisting Professor with the Department of Mathematics, Indiana University, Bloomingtion, from 1979 to 1980. He is currently Director of Researche with the French Centre National de Recherche Scientifique, Grenoble. His research includes time series analysis, signal modeling, blind source separation, array processing, and biomedical signal processing.