CCIT Report #483 May 2004

Fast Kernel Entropy Estimation and Optimization

Sarit Shwartz, Michael Zibulevsky and Yoav Y. Schechner[*]
Department of Electrical Engineering,
Technion-Israel Institute of Technology,
Haifa 32000, Israel
psarit@tx.technion.ac.il, {mzib,yoav}@ee.technion.ac.il

Abstract

Differential entropy is a quantity used in many signal processing problems. Often we need to calculate not only the entropy itself, but also its gradient with respect to various parameters, for efficient optimization, sensitivity analysis, etc. Entropy estimation can be based on an estimate of the probability density function (PDF), which is computationally costly. For this reason, some of existing algorithms have assumed rough parametric models for the PDFs, which can lead to poor performance in some scenarios. To counter these obstacles, we consider non-parametric kernel entropy estimation, which is usually computationally costly, especially for the gradient evaluation. We present two different accelerated kernel algorithms. The first of them accelerates the entropy gradient calculation based on a *back propagation* technique, which allows the calculation of a gradient of a function with the same complexity of calculating the function itself. The second algorithm accelerates the estimation of both entropy and its gradient, exploiting fast convolution over a uniform grid. We apply both algorithms to blind source separation (BSS).

[*]This research has been supported by the HASSIP Research Network Program HPRN-CT-2002-00285, sponsored by the European Commission and by the Ollendorff Minerva Center. Minerva is funded through the BMBF. Yoav Schechner is a Landau Fellow-supported by the Taub Foundation, and an Alon Fellow

1 Introduction

Differential entropy is used as a quality criterion for solution of various signal processing problems such as segmentation, detection, source separation, image registration, channel equalization and estimating depth from focus [1, 2, 3, 4, 5, 6, 7, 8, 9]. Often there is a need to calculate not only the entropy itself but also its gradient for efficient optimization. Entropy estimation is based on an estimate of the probability density function (PDF) of signals, which is computationally costly. For this reason, some existing algorithms have assumed rough parametric models for the PDFs. There are scenarios in which assuming a parametric PDFs leads to poor performance. Therefore, the need arises for non-parametric PDF and entropy estimation, which are not prohibitively complex.

In this manuscript, we develop two different methods to bypass the computational load of non parametric entropy optimization. In Sec. 2, we derive a method for accelerating gradient calculations used in entropy optimization. Calculating the gradient of a function in an explicit way can lead to higher complexity than the calculation of the function itself. We use an approach in the spirit of the *back propagation* algorithm, which is used for neural network training (see for example [10]). It is known as *backward packing* in the community of automatic differentiation [11, 12, 13]. This method allows calculation of the entropy gradient with the same complexity needed to calculate the entropy itself.

In Sec. 3, we develop a non-parametric entropy estimator that has a complexity of $\mathcal{O}(N \log N)$ for both entropy and gradient computation, where N is the number of signal samples. It is based on an approximation of the kernel estimator, which is calculated using a fast convolution over a uniform grid. The errors caused by this approximation are reasonably small. Therefore, our method can be a practical tool for large problems.

Finally, we apply the algorithms to linear blind source separation (BSS) problems also called independent component analysis (ICA). The ICA problem is based on minimization of the mutual information (MI) criterion. MI is based on the entropy of signals. By applying our methods to this problem, we boost the performance of ICA as demonstrated in Sec. 4.

1.1 Non parametric entropy estimation

Let $\mathbf{s} = [s(1), \dots, s(N)]$ be an arbitrary signal. In general, $\mathbf{s}$ can be an arbitrary function of several measurements,

$$s(n) = f(n; \mathbf{y}; \mathbf{W}) \ , \tag{1}$$

where we denote the vector of measurements as $\mathbf{y}$ and a sample's index as n. Here, $\mathbf{W}$ is the vector of parameters of function f. For example, in the special case of calculating the entropy of some measurements $f(n; \mathbf{y}; \mathbf{W}) = y(n)$, thus $s(n) = y(n)$.

The Parzen-windows estimator for the PDF of $\mathbf{s}$ at point t is

$$\hat{p}(t|\mathbf{s}) \equiv \frac{1}{N} \sum_{n=1}^{N} \varphi[t - s(n)] \ , \tag{2}$$

Where $\varphi(t)$ is a smoothing kernel. There are several options for selecting the kernel [8, 14, 15]. We use a Gaussian with zero mean and variance σ^2,

$$\varphi(t) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{t^2}{2\sigma^2}\right) \ . \tag{3}$$

A discussion about a selection of σ is given in App. B. The Parzen-windows entropy estimator [6, 9] for a signal $\mathbf{s}$ is

$$\hat{\mathcal{H}}_{\mathbf{s}} = -\frac{1}{N} \sum_{l=1}^{N} \log \left\{ \frac{1}{N} \sum_{n=1}^{N} \varphi\left[s(l) - s(n)\right] \right\} \ . \tag{4}$$

Eq. (4) requires N^2 calculations of φ. Define $N_{\text{calc\_}f}$ as the number of operations needed to of calculate $f(n; \mathbf{y}; \mathbf{W})$ using Eq. (1). We need to calculate $f(n; \mathbf{y}; \mathbf{W})$ for all N samples of $\mathbf{s}$. Thus, the overall complexity of the Parzen-windows entropy estimator is

$$\mathcal{O}_{\text{entropy}}^{\text{explicit}} = \mathcal{O}(N^2 + N N_{\text{calc\_}f}) \tag{5}$$

Define the gradient of f with respect to $\mathbf{W}$ as

$$\mathbf{g}(n;\mathbf{y};\mathbf{W}) = \nabla_{\mathbf{W}} f(n;\mathbf{y};\mathbf{W}) \; . \tag{6}$$

Then, the gradient of the the Parzen-windows entropy with respect to $\mathbf{W}$ is

$$\begin{aligned}\nabla_{\mathbf{W}} \quad \hat{\mathcal{H}}_s &= -\frac{1}{N}\sum_{l=1}^{N} \frac{\sum_{n=1}^{N} \nabla_{\mathbf{W}}\varphi\left[\mathbf{s}(l)-\mathbf{s}(n)\right]}{\sum_{n=1}^{N} \varphi\left[\mathbf{s}_k(l)-\mathbf{s}_k(n)\right]} = \\ &= \tfrac{1}{N\sigma^2} \quad \sum_{l=1}^{N} \frac{\sum_{n=1}^{N} \varphi\left[\mathbf{s}(l)-\mathbf{s}(n)\right]\left[\mathbf{s}(l)-\mathbf{s}(n)\right]\left[g(l;\mathbf{y};\mathbf{W})-g(n;\mathbf{W};\mathbf{y})\right]}{\sum_{n=1}^{N} \varphi\left[\mathbf{s}_k(l)-\mathbf{s}_k(n)\right]} \; .\end{aligned} \tag{7}$$

Eq. (7) has two explicit nested summations, apparently indicating $\mathcal{O}(N^2)$ operations. However, we also need to calculate $[g(l;\mathbf{y};\mathbf{W}) - g(n;\mathbf{W};\mathbf{y})]$ for all the nested expressions. Define $N_{\text{calc\_}g}$ as the number of operations need to calculate $\mathbf{g}(n;\mathbf{y};\mathbf{W})$. Then, in order to calculate the two nested summation we need $\mathcal{O}(N_{\text{calc\_}g}N^2)$ operations. Therefore, the overall complexity of estimating the entropy gradient is

$$\mathcal{O}^{\text{explicit}}_{\text{gradient}} = \mathcal{O}(N_{\text{calc\_}g}N^2) \; , \tag{8}$$

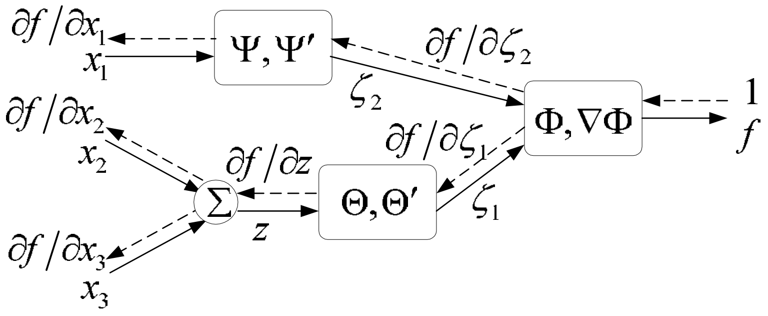
if this estimation is based explicitly on Eq. (7). Note that $\mathcal{O}^{\text{explicit}}_{\text{gradient}}$ (Eq. 8) is more expensive than $\mathcal{O}^{\text{explicit}}_{\text{entropy}}$ defined in Eq. (5).

2 Entropy gradient via back propagation

In this section we present a method for reducing the complexity of the gradient calculation (Eq. 8), so that the gradient of the entropy is calculated in a complexity which is not higher than Eq. (5). We start by presenting the general method in section 2.1. Then, we apply the method to the calculation of the entropy gradient in section 2.2.

2.1 Gradient calculation using back propagation

The back propagation technique [11, 12, 13] allows calculation of a gradient of a function with the same complexity of calculating the function itself. First,



$\partial f/\partial x_1$
x_1
Ψ, Ψ'
$\partial f/\partial \zeta_2$
ζ_2
$\Phi, \nabla\Phi$
1
f
$\partial f/\partial x_2$
x_2
$\partial f/\partial z$
$\partial f/\partial \zeta_1$
Σ
Θ, Θ'
ζ_1
z
$\partial f/\partial x_3$
x_3

Figure 1: Graphs of the forward [A] and back [B] propagation, corresponding to the function in the example of Sec. 2.1.

the function is described by slack variables and *atom functions*. We define atom functions as very simple operations into which the quality function can be factored. Each of the atom functions can have several inputs. As an example, the function

$$f(x_1, x_2, x_3) = \Phi[\Psi(x_1), \Theta(x_2 + x_3)] \tag{9}$$

is described using the slack variables

$$z \equiv x_1 + x_2, \quad \zeta_1 \equiv \Psi(x_1), \quad \zeta_2 \equiv \Theta(z) \ , \tag{10}$$

and the atom functions Φ, Ψ and Θ. The description of the function with slack variables is equivalent to describing the calculation as a directional graph. For example, the directional graph corresponding to the example of Eq. (9) is illustrated in Fig. 1A. The function is calculated by forward propagation through the graph, in which the slack variables represent inner graph layers.

During forward propagation, we calculate the value of each atom function, based on its direct slack variable inputs. While we calculate the output value of any atom function, we also calculate the gradient of this atom function with respect to its direct inputs. We save the values of these gradients for future use. For example, when calculating $\zeta_1 = \Psi(x_1)$, we also calculate and save $\frac{\mathrm{d}\Psi_{x_1}}{\mathrm{d}x_1}$, which is the coefficient of the differential $\mathrm{d}\zeta_1 = \frac{\mathrm{d}\Psi_{x_1}}{\mathrm{d}x_1}\mathrm{d}x_1$.

We now show that we can calculate the value of the gradient in a similar way to neural network training by the back propagation algorithm. Training a neural network starts by updating the network output layer. Then, the update is propagated backwards in the network. In a similar way, the process of back propagation of differentials is equivalent to replacing all the atom functions in the directional graph with multipliers. The multipliers' values are the gradients of the atom functions with respect to their direct inputs, which we had calculated during the forward propagation. In addition, we flip all the directional edges in the graph, and use 1 as an input to the inverted graph. The back propagation graph is illustrated in Fig. 1B. We use the *same* graph structure for calculating both the function value and the function derivative. Thus, we calculate the values of the gradient coefficients with the same complexity of calculating the value of the function itself.

2.2 Efficient calculation of the entropy gradient

In this section, we describe how to use back propagation in order to calculate the entropy gradient with the same complexity of calculating the entropy value. We start by defining slack variables and atom functions for the entropy equation (Eq. 4). The slack variables we use are the signals $\mathbf{s}$ and

$$p(l) \equiv \frac{1}{N}\sum_{n=1}^{N} \varphi[s(l)-s(n)] \ . \tag{11}$$

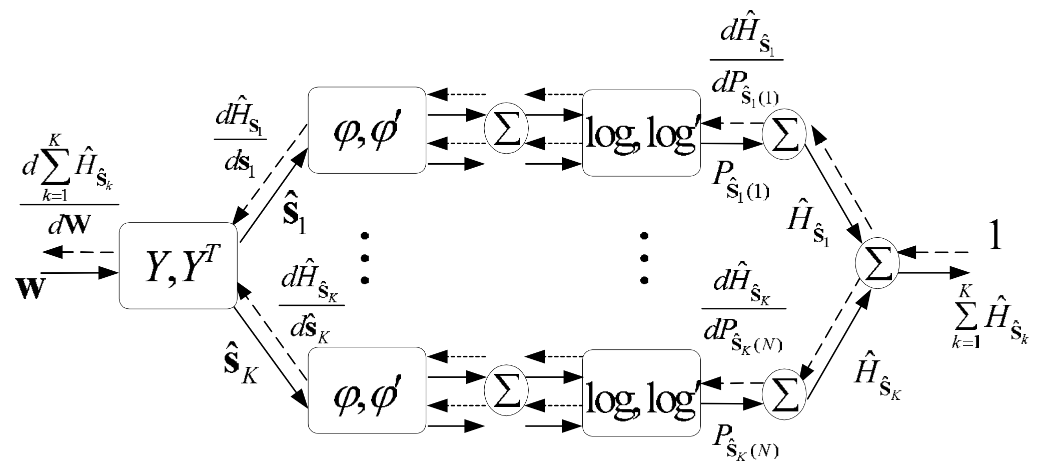
The atom functions we use are φ and L, where

$$L[p(l)] \equiv \log[p(l)] \ . \tag{12}$$

The directional graph describing the entropy calculation is illustrated in Fig. 2.

In order to calculate the entropy value, we use forward propagation. The forward propagation constitutes the following consecutive steps:

$$\text{(a)} \qquad \forall l \in \{1,\ldots,N\}, \quad s(l) = f(l;\mathbf{y};\mathbf{W}) \quad ; \quad \text{save} \ \ \mathbf{g}(l;\mathbf{W};\mathbf{y}) \ .$$



$\frac{d\sum_{k=1}^{K}\hat{H}_{\hat{\mathbf{s}}_k}}{d\mathbf{W}}$
$\mathbf{W}$
Y,Y^T
$\frac{d\hat{H}_{\hat{\mathbf{s}}_1}}{d\hat{\mathbf{s}}_1}$
$\hat{\mathbf{s}}_1$
$\frac{d\hat{H}_{\hat{\mathbf{s}}_K}}{d\hat{\mathbf{s}}_K}$
$\hat{\mathbf{s}}_K$
φ,φ'
Σ
$\log,\log'$
$\frac{d\hat{H}_{\hat{\mathbf{s}}_1}}{dP_{\hat{\mathbf{s}}_1(1)}}$
$P_{\hat{\mathbf{s}}_1(1)}$
$\hat{H}_{\hat{\mathbf{s}}_1}$
$\frac{d\hat{H}_{\hat{\mathbf{s}}_K}}{dP_{\hat{\mathbf{s}}_K(N)}}$
$P_{\hat{\mathbf{s}}_K(N)}$
$\hat{H}_{\hat{\mathbf{s}}_K}$
1
$\sum_{k=1}^{K}\hat{H}_{\hat{\mathbf{s}}_k}$

Figure 2: Graphs for forward [A] and back [B] propagation for estimating the entropy and its gradient.

$$\text{(b)} \qquad \forall l \in \{1,\ldots,N\}, \quad p(l) = \frac{1}{N}\sum_{n=1}^{N}\varphi[s(l)-s(n)]\ ; \qquad (13)$$
$$\text{save} \quad \varphi'[s(l)-s(n)]\ .$$
$$\text{(c)} \qquad H_{\mathbf{s}} = -\frac{1}{N}\sum_{l=1}^{N}L[p(l)] \quad ; \quad \text{save} \quad L'[p(l)] = [p(l)]^{-1}\ .$$

The complexity of step (a) is $\mathcal{O}[N(N_{\text{calc\_}g}+N_{\text{calc\_}f})]$. The complexity of step (b) is $\mathcal{O}(N^2)$. The complexity of step (c) is $\mathcal{O}(N)$. Therefore, the overall complexity of forward propagation is

$$\begin{aligned}\mathcal{O}_{\text{forward}} = \mathcal{O}[N(N_{\text{calc\_}g}+N_{\text{calc\_}f}) \quad &+ \quad N^2 + N] = \\ &= \quad \mathcal{O}[N(N_{\text{calc\_}g}+N_{\text{calc\_}f}) + N^2]\ .\end{aligned} \qquad (14)$$

The differentials of the entropy and all the slack variables used in Eq. (13) are

$$\text{(a)} \qquad \forall l \in \{1,\ldots,N\},\ \mathrm{d}s(l) = \langle g(l;\mathbf{W};\mathbf{y}), \mathrm{d}\mathbf{W}\rangle\ , \qquad (15)$$
$$\text{(b)} \qquad \forall l \in \{1,\ldots,N\},\ \mathrm{d}p(l) = \frac{1}{N}\sum_{n=1}^{N}\varphi'[s(l)-s(n)][\mathrm{d}s(l)-\mathrm{d}s(n)]$$

$$\text{(c)} \qquad \mathrm{d}H_{\mathbf{s}} = -\frac{1}{N}\sum_{l=1}^{N} L'[p(l)]\mathrm{d}p(l) \ ,$$

where φ' is the derivative of φ with respect to $[s(l) - s(n)]$, and L' is the derivative of L with respect to $p(l)$. Thus, whenever we calculate one of the slack variables in the forward propagation, we also calculate its gradient with respect to its direct inputs.

After we finish calculating the entropy value, we calculate the gradient by back propagation of the differentials. This is equivalent to substitution of Eq. (15) in reverse order

$$\text{(a)} \qquad \forall l \in \{1,\dots,N\}, \quad \frac{\mathrm{d}}{\mathrm{d}p(l)}H_{\mathbf{s}} = L'[p(l)] \ . \tag{16}$$

$$\text{(b)} \qquad \forall l \in \{1,\dots,N\}, \quad \frac{\mathrm{d}}{\mathrm{d}s(l)}H_{\mathbf{s}} = \frac{1}{N}\sum_{n=1}^{N}\varphi'[s(l)-s(n)]\left[\frac{\mathrm{d}}{\mathrm{d}p(l)}H_{\mathbf{s}}\right] \ .$$

$$\text{(c)} \qquad \forall n \in \{1,\dots,N\}, \quad \frac{\mathrm{d}}{\mathrm{d}s(n)}H_{\mathbf{s}} = \frac{-1}{N}\sum_{n=1}^{N}\varphi'[s(l)-s(n)]\left[\frac{\mathrm{d}}{\mathrm{d}p(l)}H_{\mathbf{s}}\right] \ .$$

$$\text{(d)} \qquad \frac{\mathrm{d}}{\mathrm{d}\mathbf{W}}H_{\mathbf{s}} = \sum_{l=1}^{N}\mathbf{g}(l;\mathbf{W};\mathbf{y})\left[\frac{\mathrm{d}}{\mathrm{d}s(l)}H_{\mathbf{s}}\right] \ .$$

The complexity of step (a) is $\mathcal{O}(N)$. The complexity of stages (b) and (c) is $\mathcal{O}(N^2)$. The complexity of stage (d) is $\mathcal{O}(NN_{\text{calc\_}g})$. Therefore, the overall complexity of back propagation is

$$\mathcal{O}_{\text{back}} = \mathcal{O}(N^2 + NN_{\text{calc\_}g} + N) = \mathcal{O}(N^2 + NN_{\text{calc\_}g}) \ . \tag{17}$$

The back propagation complexity is similar to the forward propagation complexity. The exception is the term $\mathcal{O}(N_{\text{calc\_}f})$, which is missing from the back propagation complexity. This term indicates the complexity of calculating $\mathbf{s}$ from $\mathbf{y}$ and is done only once during the forward propagation process. Combining all the algorithm stages yields estimation of both the entropy estimator and its gradient in a complexity of

$$\mathcal{O}^{\text{gradient}}_{\text{backpropagation}} = \mathcal{O}_{\text{forward}} + \mathcal{O}_{\text{back}} = \mathcal{O}[N^2 + N(N_{\text{calc\_}g} + N_{\text{calc\_}f})] \ . \tag{18}$$

This complexity is less expensive than the complexity of Eq. (8). We summarize in a pseudo-code the calculation of both the entropy and its gradient. The pseudo-code is given in Fig. 3

Input: $\mathbf{W}$, $\mathbf{s}$

Output: $H_{\mathbf{s}}$, $\frac{\mathrm{d}H_{\mathbf{s}}}{\mathrm{d}\mathbf{W}}$

Algorithm:

```
For $l = 1$ to $N$
    $s(l) = f(l; \mathbf{y}; \mathbf{W})$
    $\mathbf{g}(l; \mathbf{W}; \mathbf{y}) = \nabla_{\mathbf{W}} f(l; \mathbf{y}; \mathbf{W})$
end
For $l = 1$ to $N$
    For $n = 1$ to $N$
        $p(l) = p(l) + \varphi[s(l) - s(n)]/N$
        $p'(l) = \varphi'[s(l) - s(n)]/N$
    end
    $H_{\mathbf{s}} = H_{\mathbf{s}} - \log[p(l)]$
    $L'(l) = 1/p(l)$
    For $n = 1$ to $N$
        $\frac{\mathrm{d}}{\mathrm{d}s(l)} H_{\mathbf{s}} = \frac{\mathrm{d}}{\mathrm{d}s(l)} H_{\mathbf{s}} - p'(l)L'(l)/N$
        $\frac{\mathrm{d}}{\mathrm{d}s(n)} H_{\mathbf{s}} = \frac{\mathrm{d}}{\mathrm{d}s(n)} H_{\mathbf{s}} + p'(l)L'(l)/N$
    end
end
For $l = 1$ to $N$
    $\frac{\mathrm{d}}{\mathrm{d}\mathbf{W}} H_{\mathbf{s}} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{W}} H_{\mathbf{s}} + \mathbf{g}(l; \mathbf{W}; \mathbf{y}) \left[\frac{\mathrm{d}}{\mathrm{d}s(l)} H_{\mathbf{s}}\right]$
end
```

Figure 3: Pseudo-code for calculating the entropy gradient via back propagation

3 Entropy estimation with $N \log N$ complexity

In this section, we develop a non-parametric entropy estimator that has a complexity of $\mathcal{O}(N \log N)$. The kernel entropy is approximated by convolution of the signal histogram with the kernel. In addition, we develop an approximation to the entropy gradient, whose calculation has the same complexity.

3.1 Efficient calculation of the entropy estimator

3.1.1 Motivation

Calculating the differential entropy using Eq. (4) requires N^2 calculations of φ. However, consider the situation in which the signal values (range) are discrete and reside on a uniform grid, i.e., they are uniformly quantized. In such a situation, we can estimate the differential entropy with $\mathcal{O}(N \log N)$ operations. In this section we prove this claim. First, we show that for a signal having uniformly quantized values, the sum $\frac{1}{N}\sum \varphi[s(l) - s(n)]$ is equivalent to estimating the probability of $s(l)$ based on a smoothed histogram of $\mathbf{s}$.

The empirical probability of $s(l)$ estimated using N uniformly quantized samples of $\mathbf{s}$ is

$$\hat{P}[s(l)] = (1/N) \sum_{n=1}^{N} \delta[s(l) - s(n)] \;, \tag{19}$$

which is equivalent to the value of the histogram bin in which the value $s(l)$ falls. Let us sample the smoothing kernel φ on the histograms bin grid, leading to the discrete kernel $\varphi_{\text{sampled}}(j),\ j \in [1, N_{\text{kernel}}]$, where N_{kernel} is the support of φ_{sampled}. Then, we perform a discrete convolution of φ_{sampled} with the histogram, leading to

$$\hat{P}[s(l)] * \varphi_{\text{sampled}} \;=\; (1/N) \sum_{j=1}^{N_{\text{kernel}}} \sum_{n=1}^{N} \delta[s(l) - s(n) - j]\varphi_{\text{sampled}}(j) =$$

$$= \ (1/N)\sum_{n=1}^{N}\varphi[s(l)-s(n)] = \hat{p}[s(l)|\mathbf{s}] \ . \tag{20}$$

Where $\hat{p}$ is the PDF defined in Eq. (2). Hence, the inner sum in Eq. (4) can be calculated simultaneously for all l using a discrete convolution.

In order to calculate the histogram of a signal, we need to scan all of the signal samples, requiring $\mathcal{O}(N)$ operations. Afterwards, we convolve the histogram with φ_{sampled} using a fast discrete convolution[1] based on FFT. This step takes $\mathcal{O}(N_{\text{bins}}\log N_{\text{kernel}})$ operations, where N_{bins} is the number of histogram bins. Finally, the differential entropy estimation is calculated based on Eqs. (4) and (20)

$$\hat{\mathcal{H}}_{\mathbf{s}} = -(1/N)\sum_{l=1}^{N}\log\{\hat{p}[s(l)|\mathbf{s}]\} \ . \tag{21}$$

Eq. (21) requires $\mathcal{O}(N)$ operations, in addition to the operations needed for Eq. (20). Overall, we achieve a complexity of $\mathcal{O}(N + N_{\text{bins}}\log N_{\text{kernel}})$. Typically, N_{bins} is of the order of N or smaller. Therefore, the complexity we achieve is $\mathcal{O}(N\log N)$.

Clearly, quantization to a uniform grid leads to a high computational efficiency. Yet, we should ensure that we can perform this quantization without damaging the quality of differential entropy estimation.

3.1.2 The quantization procedure

In order to quantize a signal and create a histogram, we start by defining a vote function v on a uniform quantization grid. Each bin is δ_v wide. The vote function is an approximation of the histogram. We update the vote function in the following way. Let $m^{\#}$ be the index of the bin closest to the value of $s(n)$. It satisfies $m^{\#} \leq s(n)/\delta_v \leq m^{\#}+1$. Define the distance of the signal value $s(n)$ from the index $m^{\#}$ (normalized by δ_v) as:

$$\eta = \frac{s(n)}{\delta v} - m^{\#} \ . \tag{22}$$

[1]We used a Matlab code for fast convolution based on FFT, which had been written by Luigi Rosa, email: luigi.rosa@tiscali.it , http://utenti.lycos.it/matlab

Let $h(\eta)$ be a window function[2]that satisfies

$$h(1-\eta) = 1 - h(\eta) \quad 0 \leq \eta \leq 1 \; . \tag{23}$$

For each sample of the signal $s(n)$, $n = 1, 2, \ldots, N$ we update the voting by

$$v(m) \leftarrow \begin{cases} v(m) + h(\eta) & \text{for } \; m = m^{\#} \\ v(m) + 1 - h(\eta) & \text{for } \; m = m^{\#} + 1 \end{cases} , \tag{24}$$

where η is given by Eq. (22). After the voting is over, the quantization process is done, resulting in a vector $\mathbf{v}$. The voting process requires a scan of all the signal samples and therefore has a complexity of $\mathcal{O}_{\text{vote}} = \mathcal{O}(N)$.

Now, we associate the probability $\hat{P}(s)$ with $\mathbf{v}/N$. Following Eq. (20), we convolve $\mathbf{v}/N$ with φ_{sampled}:

$$\hat{p}_{\text{quant}} = (\mathbf{v}/N) * \varphi_{\text{sampled}} \; . \tag{25}$$

Since we use an efficient convolution, this stage has a complexity of $\mathcal{O}_{\text{convolve}} = \mathcal{O}(N \log N)$.

3.1.3 From histogram to differential entropy

Apparently, a natural method to estimate entropy from a histogram is to follow the discrete entropy definition and use

$$\tilde{\mathcal{H}}_{\mathbf{s}} = \sum_{m=1}^{N_{\text{bins}}} \hat{p}_{\text{quant}}(m) \log[\hat{p}_{\text{quant}}(m)] \; . \tag{26}$$

However, the use of discrete binning causes fluctuations in the entropy estimate as a function of $\mathbf{W}$. In addition, the entropy calculated by Eq. (4) is based on a PDF estimate at $s(l)$, rather than the discrete probability p_{quant}. We thus estimate a PDF $\hat{p}[s(l)|\mathbf{s}]$. We achieve this simply by interpolating the histogram given in Eq. (25)

$$\hat{p}[s(l)|\mathbf{s}] = h(\eta)\hat{p}_{\text{quant}}(m^{\#}) + [1 - h(\eta)]\hat{p}_{\text{quant}}(m^{\#} + 1) \; , \tag{27}$$

[2]We use a linear interpolation function $h(\eta) = 1 - \eta$.

where $m^{\#}$ and η are defined in Sec. 3.1.2. The interpolation requires a scan of all the signal samples $s(l)$, $l = 1, \ldots, N$. Therefore it has a complexity of $\mathcal{O}_{\text{interpolate}} = \mathcal{O}(N)$. Finally, the estimate of the differential entropy is calculated by

$$\hat{\mathcal{H}}_{\mathbf{s}} = -(1/N) \sum_{n=1}^{N} \log\{\hat{p}[s(l)|\mathbf{s}]\} \ . \tag{28}$$

Eq. (28) requires $\mathcal{O}(N)$ operations. Therefore, the overall complexity of calculating the signal entropy is

$$\mathcal{O}_{\text{approx}}^{\text{entropy}} = \mathcal{O}(N) + \mathcal{O}_{\text{interpolate}} + \mathcal{O}_{\text{convolve}} + \mathcal{O}_{\text{vote}} = \mathcal{O}(N \log N) \ . \tag{29}$$

This is significantly lower than the complexity $\mathcal{O}_{\text{entropy}}^{\text{explicit}}$ given in Eq. (5).

3.2 Efficient estimation of the entropy gradient

In order to compute the gradient, we could have differentiated the entropy approximation which we derived in Sec. 3.1. However, as we already mentioned, discrete binning causes fluctuations, which can stop optimization at local minima. We avoid this problem altogether by taking a different approach. Rather than differentiating an approximation based on quantization, we elect to approximate derivatives associated with continuous values. We thus apply the approximation process directly on the gradient of differential entropy (Eq. 7). We do so in a similar manner to the approximation of the differential entropy itself.

The entropy gradient Eq. (7) can be calculated in two stages, following a chain rule. First, we calculate the entropy gradient with respect to the signal samples. Then, we calculate the entropy gradient with respect to the desired parameters by

$$\nabla_{\mathbf{W}} \mathcal{H}_{\mathbf{s}} = \frac{1}{N} \sum_{l=1}^{N} \mathbf{g}(l; \mathbf{W}; \mathbf{y}) \left[\frac{\mathrm{d}\mathcal{H}_{\mathbf{s}}}{\mathrm{d}s(l)} \right] \ , \tag{30}$$

where $\mathbf{g}(l; \mathbf{W}; \mathbf{y})$ is given by Eq. (6). This equation has a complexity of

$\mathcal{O}(NN_{\text{calc\_}g})$. We define

$$\Phi'[s(l)|\mathbf{s}] \equiv (1/N)\sum_{n=1}^{N}\varphi'[s(l)-s(n)] \tag{31}$$

and

$$F'[s(l)] \equiv \frac{1}{N}\sum_{n=1}^{N}\frac{\varphi'[s(n)-s(l)]}{\hat{p}[s(n)|\mathbf{s}]} \ , \tag{32}$$

where $\hat{p}[s(l)|\mathbf{s}]$ is defined in Eq. (2). The derivatives of the entropy (Eq. 30) are given by

$$\frac{\mathrm{d}\mathcal{H}_{\mathbf{s}}}{\mathrm{d}s(l)} = \frac{1}{N}\frac{\Phi'[s(l)|\mathbf{s}]}{\hat{p}[s(l)|\mathbf{s}]} - F'[s(l)] \ . \tag{33}$$

Note that $\hat{p}[s(n)|\mathbf{s}]$ is known, since we calculate $\hat{p}[s(n)|\mathbf{s}]$ when we calculate the differential entropy itself, prior to the gradient calculation. Now, we will develop an efficient procedure for calculating Φ' and F'.

3.2.1 Approximating a quantized Φ'

We showed in Eq. (20) that when the values of a signal are uniformly quantized, $\hat{p}[s(l)|\mathbf{s}]$ is equivalent to a convolution of φ_{sampled} with the histogram of the signal. In the same manner, $\Phi'[s(l)|\mathbf{s}]$ is the convolution of the histogram with the sampled sequence of φ', which we term $\varphi'_{\text{sampled}}$. When the signal values are not uniformly quantized, we need to perform quantization and interpolation procedures similarly to Eq. (24) and Eq. (27). This yields the quantized Φ', which we denote as Φ'_{quant}. The complexity of these operations is $\quad \mathcal{O}_{\Phi'_{\text{quant}}} = \mathcal{O}_{\text{convolve}} + \mathcal{O}_{\text{vote}} = \mathcal{O}(N\log N)$.

3.2.2 Approximating a quantized F'

Now, we show that when the values of a signal are uniformly quantized, $F'[s(l)]$ is a weighted histogram convolved with $\varphi'_{\text{sampled}}$. Define a weighted histogram of the signal, in which every signal value is inversely weighted by its estimated probability,

$$F[s(l)] \equiv \frac{1}{N}\sum_{r=1}^{N}\frac{\delta[s(r)-s(l)]}{\hat{p}[s(r)|\mathbf{s}]} \ . \tag{34}$$

Then, convolve $F(s)$ with $\varphi'_{\text{sampled}}(-j)$, which is $\varphi'_{\text{sampled}}$ in reverse order:

$$\begin{aligned}
\sum_{j=1}^{N_{\text{kernel}}} F \quad & [s(r) - j]\varphi'_{\text{sampled}}(-j) = \\
&= \frac{1}{N}\sum_{j=1}^{N_{\text{kernel}}}\sum_{l=1}^{N}\frac{\delta[s(l)-s(r)+j]}{\hat{P}[s(l)]}\varphi'_{\text{sampled}}(-j) = \\
&= \frac{1}{N}\sum_{l=1}^{N}\frac{1}{\hat{p}[s(l)|\mathbf{s}]}\sum_{j=1}^{N_{\text{kernel}}}\delta[s(l)-s(r)+j]\varphi'_{\text{sampled}}(-j) = \\
&= \frac{1}{N}\sum_{l=1}^{N}\frac{\varphi'_{\text{sampled}}[s(l)-s(r)]}{\hat{p}[s(l)|\mathbf{s}]} = F'[s(l)], \qquad (35)
\end{aligned}$$

Hence, in order to calculate the second term of Eq. (33), we create a weighted histogram F and then convolve it with $\varphi'_{\text{sampled}}(-j)$.

In general, the signal values are not uniformly quantized. Hence, we estimate the weighted histogram in the manner we estimate the histogram itself (section 3.1.2). We start by defining a weighted vote function v_w. The weighted vote function is defined on the same uniformly quantized grid as v. Then, for each sample of the signal $s(n)$, $n = 1, 2, \ldots, N$ we update the voting by

$$v_w(m) \leftarrow \begin{cases} v_w(m) + h(\eta)/\hat{p}[s(n)|\mathbf{s}] & \text{for } m = m^{\#} \\ v_w(m) + [1 - h(\eta)]/\hat{p}[s(n)|\mathbf{s}] & \text{for } m = m^{\#} + 1 \end{cases} . \qquad (36)$$

After the voting, we associate $\mathbf{v}_w/N$ with the weighted histogram F. Finally, according to Eq. (35) we convolve $\mathbf{v}_w/N$ with $\varphi'_{\text{sampled}}$. This convolution yields the quantized F'. We denote the quantized F' as F'_{quant}. The complexity of calculating F'_{quant} is $\mathcal{O}_{F'_{\text{quant}}} = \mathcal{O}_{\text{convolve}} + \mathcal{O}_{\text{vote}} = \mathcal{O}(N \log N)$.

3.2.3 Combining all the approximations

In order to calculate the differential entropy gradient (Eq. 33), we need to estimate $\Phi'[s(l)|\mathbf{s}]$ and $F'[s(l)]$ from F'_{quant} and Φ'_{quant}. As was explained in section 3.1.3, we use interpolation. For each signal sample $s(l)$

$$\begin{aligned}
\Phi'[s(l)|\mathbf{s}] &= h(\eta)\Phi'_{\text{quant}}(m^{\#}) + [1-h(\eta)]\Phi'_{\text{quant}}(m^{\#}+1) \\
F'[s(l)|\mathbf{s}] &= h(\eta)F'_{\text{q}}(m^{\#}) + [1-h(\eta)]F'_{\text{quant}}(m^{\#}+1)\ .
\end{aligned} \tag{37}$$

This interpolation has a complexity of $\mathcal{O}_{\text{interpolate}}$.

Finally, the gradient of the signal entropy is calculated using Eq. (30) and Eq. (33) with a complexity of

$$\begin{aligned}
\mathcal{O}^{\text{gradient}}_{\text{approx}} = \mathcal{O}_{\Phi'_{\text{quant}}} + \mathcal{O}_{F'_{\text{quant}}} &+ \mathcal{O}_{\text{interpolate}} + NN_{\text{calc\_}g} = \\
&= \mathcal{O}(N\log N + NN_{\text{calc\_}g})\ .
\end{aligned} \tag{38}$$

This complexity is significantly smaller than the complexity $\mathcal{O}^{\text{explicit}}_{\text{gradient}}$ appearing in Eq. (8). A pseudo-code for the differential entropy estimator and its gradient is given in Fig. 4.

Input: $\mathbf{W}$, $\mathbf{y}$

Output: $H_{\mathbf{s}}, \frac{\mathrm{d}H_{\mathbf{s}}}{\mathrm{d}\mathbf{W}}$

Algorithm:

```
For $l = 1$ to $N$
    $s(l) = f(l; \mathbf{W}; \mathbf{y})$
end
For $l = 1$ to $N$
    find and save $m^{\#}, \eta$ for $s(l)$
    $v(m^{\#}) \leftarrow [v(m^{\#}) + h(\eta)]$
    $v(m^{\#}+1) \leftarrow [v(m^{\#}+1) + 1 - h(\eta)]$
end
$\hat{p}_{\text{quant}} = (v * \varphi_{\text{sampled}})/N$
$\Phi'_{\text{quant}} = (v * \varphi'_{\text{sampled}})/N$
For $l = 1$ to $N$
    load $m^{\#}, \eta$
    $\hat{p}[s(l)|\mathbf{s}] = h(\eta)\hat{p}_{\text{quant}}(m^{\#}) + [1 - h(\eta)]\hat{p}_{\text{quant}}(m^{\#}+1)$
    $\Phi'[s(l)|\mathbf{s}] = h(\eta)\Phi'_{\text{quant}}(m^{\#}) + [1 - h(\eta)]\Phi'_{\text{quant}}(m^{\#}+1)$
    $H_{\mathbf{s}} = H_{\mathbf{s}} - \log\{\hat{p}[s(l)|\mathbf{s}]\}/N$
    $v_w(m^{\#}) = v_w(m^{\#}) - h(\eta)/(\hat{p}[s(l)|\mathbf{s}]N)$
    $v_w(m^{\#}+1) = v_w(m^{\#}+1) - [1 - h(\eta)]/(\hat{p}[s(l)|\mathbf{s}]N)$
    $\frac{\mathrm{d}}{\mathrm{d}s(l)}\mathcal{H}_{\mathbf{s}} = \frac{\mathrm{d}}{\mathrm{d}s(l)}\mathcal{H}_{\mathbf{s}} - \Phi'[s(l)|\mathbf{s}]/(\hat{p}[s(l)|\mathbf{s}]N)$
end
$F'_{\text{quant}} = v_w * \varphi'_{\text{sampled}}$
For $l = 1$ to $N$
    load $m^{\#}, \eta$
    $F'[s(l)|\mathbf{s}] = h(\eta)F'_{\text{quant}}(m^{\#}) + [1 - h(\eta)]F'_{\text{quant}}(m^{\#}+1)$
    $\frac{\mathrm{d}}{\mathrm{d}s(l)}\mathcal{H}_{\mathbf{s}} = \frac{\mathrm{d}}{\mathrm{d}s(l)}\mathcal{H}_{\mathbf{s}} + F'[s(l)|\mathbf{s}]$
end
For $l = 1$ to $N$
    $\frac{\mathrm{d}}{\mathrm{d}\mathbf{W}}H_{\mathbf{s}} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{W}}H_{\mathbf{s}} + \mathbf{g}(l; \mathbf{W}; \mathbf{y})\left[\frac{\mathrm{d}}{\mathrm{d}s(l)}H_{\mathbf{s}}\right]$
end
```

Figure 4: Pseudo-code for calculating the entropy and its gradient via fast kernel convolutions.

4 ICA using fast kernel entropy estimation

MI of signals is a natural criterion for statistical dependency and therefore it is used in ICA algorithms. MI is based on estimates of entropies of signals. In order to avoid the computational complexity of entropy estimation, existing ICA algorithms have assumed rough models for the signals PDFs [16, 17, 18] or used high order cumulants instead of MI [19]. These approximations can sometimes lead to failure, as demonstrated in [6] as well as in section 4.3. In contrast, robust separation can be achieved with non-parametric kernel-based estimation of PDFs [6]. The drawback of current implementations of that approach is high computational complexity. For K sources, each of which having N samples, an existing non-parametric ICA algorithm [6] has a complexity of $\mathcal{O}(K^2N^2 + K^2N)$. Another existing algorithm given in [20] has a complexity of $\mathcal{O}(3^K N + K^2N)$, which may be tolerated for a small number of sources, but has exponential growth in K.

The complexity of non-parametric ICA algorithms such as [6] stems from the calculation of the MI gradient. By applying the back propagation technique, we reduce the complexity of calculating the MI gradient to be similar to the complexity of calculating the MI itself. The MI calculation requires $\mathcal{O}(KN^2 + K^2N)$ operations. A more significant acceleration is achieved by applying the entropy approximation based on discrete convolution. This method reduces the complexity of calculating both the MI and its gradient to $\mathcal{O}(KN \log N + K^2N)$.

4.1 ICA and mutual information

Let $\{\mathbf{s}_1, \mathbf{s}_2, ...\mathbf{s}_K\}$ be a set of independent sources. Each source is of the form $\mathbf{s}_k = [s_k(1), s_k(2), \ldots, s_k(N)]^T$. Let $\{\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_K\}$ be a set of measured signals, each of which being a linear mixture of the sources. Denote $\{\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, ...\hat{\mathbf{s}}_K\}$ as the set of the reconstructed sources and $\mathbf{W}$ the separation matrix. Then,

$$[\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, ..., \hat{\mathbf{s}}_K]^T = \mathbf{W}[\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_K]^T \ . \tag{39}$$

The mutual information of the K random variables $\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, ...\hat{\mathbf{s}}_K$ is (see for examples [21])

$$\mathcal{I}(\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, ...\hat{\mathbf{s}}_K) = \mathcal{H}_{\hat{\mathbf{s}}_1} + ... + \mathcal{H}_{\hat{\mathbf{s}}_K} - \log|\det(\mathbf{W})| - \mathcal{H}_{\text{measurements}} \ , \tag{40}$$

where $\mathcal{H}(\hat{\mathbf{s}}_k)$ is the entropy of $\hat{\mathbf{s}}_k$ and $\mathcal{H}_{\text{measurements}}$ is independent of $\mathbf{W}$ and is thus constant for a given sample set $\{\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_K\}$. For this reason we will ignore it.

The goal of ICA is to find the separation matrix $\mathbf{W}$ that leads to estimated sources that are independent, thus inverting the mixing process. The independence criterion is the MI of the estimated sources. The minimization problem that we solve is

$$\min_{\mathbf{W}} \left\{ \sum_{k=1}^{K} \mathcal{H}_{\hat{\mathbf{s}}_k} - \log|\det(\mathbf{W})| + \lambda E_{\text{normalization}} \right\} \ , \tag{41}$$

where

$$E_{\text{normalization}} \equiv \sum_{k=1}^{K} \left(\frac{\|\hat{\mathbf{s}}_k\|}{\sqrt{N}} - 1 \right)^2 \tag{42}$$

penalizes for un-normalized sources. This penalty, weighted by a constant λ resolves ambiguities arising from the scale invariance of MI.[3] The complexity of calculating a signal norm is N. Therefore, the complexity of Eq. (42) is $\mathcal{O}_{\text{penalty}} = \mathcal{O}(KN)$. The gradient of this term is:

$$\frac{2}{\sqrt{N}} \sum_{k=1}^{K} \left(\frac{1}{\sqrt{N}} - \frac{1}{\|\hat{\mathbf{s}}_k\|} \right) \left[\sum_{n=1}^{N} \hat{\mathbf{s}}_k(n) \Upsilon_{k,n} \right] \ , \tag{43}$$

where $\Upsilon_{k,n}$ is a $K \times K$ matrix, all of whose rows are zeros except the k'th row. That row equals $[y_1(n), \ldots, y_K(n)]$. Eq. (43) has two nested summation over matrices which have K non zero terms. Therefore the gradient of the penalty term has a complexity of $\mathcal{O}^{\text{penalty}}_{\text{gradient}} = \mathcal{O}(K^2N)$.

[3]This term does not affect the separation quality, but improves convergence of the optimization algorithm as explained in appendix A.

The gradient of $\log|\det(\mathbf{W})|$ is

$$\nabla_{\mathbf{W}}[\log|\det(\mathbf{W})|] = (\mathbf{W}^{-1})^T \tag{44}$$

(for example see [21]). The only terms in Eq. (40) that remains to be addressed in the optimization formulation are the entropies of the estimated sources $\mathcal{H}_{\hat{\mathbf{s}}_k}$. For a non-parametric estimate of these entropies, we use the Parzen-windows estimator.

4.2 Estimation of MI and its gradient

Substituting the Parzen-windows entropy estimator Eq. (4) into the MI equation Eq. (40) yields the MI estimator

$$\mathcal{I}(\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, ...\hat{\mathbf{s}}_K) = -\sum_{k=1}^{K} \frac{1}{N} \sum_{l=1}^{N} \log \left\{ \frac{1}{N} \sum_{n=1}^{N} \varphi\left[\hat{s}_k(l) - \hat{s}_k(n)\right] \right\} - \log|\det(\mathbf{W})| \tag{45}$$

where we ignore $\mathcal{H}_{\text{measurements}}$ and recalling that we already handled the normalization term in Eqs. (42,43). As in Eq. (1), denote f_k as the function creating the k'th signal $\hat{\mathbf{s}}_k$ based on the recorded signals $[\mathbf{y}_1, \ldots, \mathbf{y}_K]$. Then,

$$\hat{s}_k(n) = f_k(n; \mathbf{y}; \mathbf{W}) = [w_{k,1}, \ldots, w_{k,K}][y_1(n), \ldots, y_K(n)]^T \ , \tag{46}$$

where $[w_{k,1}, \ldots, w_{k,K}]$ is the k'th row of $\mathbf{W}$. The gradient of Eq. (46) is

$$\mathbf{g}_k(n; \mathbf{y}; \mathbf{W}) = [y_1(n), \ldots, y_K(n)]^T \ . \tag{47}$$

Calculating $f_k(n; \mathbf{y}; \mathbf{W})$ is done by K multiplications, while $\mathbf{g}_k(n; \mathbf{y}; \mathbf{W})$ has K terms. Thus, the complexity of calculating both terms is $N_{\text{calc\_}f} = N_{\text{calc\_}g} = K$.

Substituting Eq. (47) into the gradient of Eq. (45), yields (see [6, 7])

$$\begin{aligned} \nabla_{\mathbf{W}} \quad & \mathcal{I}(\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, ...\hat{\mathbf{s}}_K) = \\ = \quad & \sum_{k=1}^{K} \left[\frac{1}{N\sigma^2} \sum_{l=1}^{N} \frac{\sum_{n=1}^{N} \varphi\left[\hat{\mathbf{s}}_k(l) - \hat{\mathbf{s}}_k(n)\right] \left[\hat{\mathbf{s}}_k(l) - \hat{\mathbf{s}}_k(n)\right] \mathbf{Y}_{k,l,n}}{\sum_{n=1}^{N} \varphi\left[\hat{\mathbf{s}}_k(l) - \hat{\mathbf{s}}_k(n)\right]} \right] - (\mathbf{W}^{-1})^T \ . \end{aligned} \tag{48}$$

Where $\mathbf{Y}_{k,l,n} = \Upsilon_{k,l} - \Upsilon_{k,n}$.

The complexity of Eq. (45) is equal to K times the complexity of the entropy estimator, since we are calculating the entropy of K individual signals. Thus,

$$\mathcal{O}_{\mathrm{MI}}^{\mathrm{explicit}} = K\mathcal{O}_{\mathrm{entropy}}^{\mathrm{explicit}} = \mathcal{O}(KN^2 + KNN_{\mathrm{calc\_}f}) = \mathcal{O}(KN^2 + K^2N) \ . \quad (49)$$

For the same reason, the complexity of Eq. (48) is equal to K times the complexity of calculating the gradient of the entropy estimator,

$$\mathcal{O}_{\mathrm{MIgrad}}^{\mathrm{explicit}} = K\mathcal{O}_{\mathrm{gradient}}^{\mathrm{explicit}} = \mathcal{O}(KN_{\mathrm{calc\_}g}N^2) = \mathcal{O}(K^2N^2) \ . \quad (50)$$

By applying back propagation we achieve a complexity of $\mathcal{O}_{\mathrm{MIgrad}}^{\mathrm{backpropgation}} = \mathcal{O}(KN^2 + K^2N)$ for calculating both the MI and its gradient. However, by applying entropy approximation by discrete convolution we achieve a complexity of $\mathcal{O}_{\mathrm{MI}}^{\mathrm{approx}} = \mathcal{O}_{\mathrm{MIgrad}}^{\mathrm{approx}} = \mathcal{O}(KN\log N + K^2N)$. This complexity is significantly lower than Eqs. (49) and (50). This allows fast performance of ICA, while exploiting the advantages of non parametric methods in high dimensional problems.

4.3 Demonstrations

In order to evaluate our methods, we performed numerous separation simulations. The first set of simulations dealt with random sources. We simulated six sources: four of the sources were random i.i.d., while the other two were extracted as data vectors from the Lena and Trees standard pictures. The random i.i.d. sources had different PDFs (an exponential PDF[$\alpha = 2$], an exponential PDF[$\alpha = 0.6$], a normal PDF[0,1] and a Rayleigh PDF[$\beta = 1$]). The sample size of each signal was 3K. The sources were mixed using randomly generated full rank matrices (condition number$\leq$ 20).

The source separation was attempted using three parametric ICA algorithms. [19, 21, 22]: InfoMax, Jade and Fast ICA. In addition, separation was attempted using two non parametric ICA algorithms: the first is based on [6]. We implemented the algorithm described in [6] with the exception of using the method described in Sec. 2, in order to accelerate the gradient calculation. The second algorithm is the one we described in Sec. 3. The software for the prior algorithms was downloaded from the web-pages of the respective authors.

In order to limit the signals to the grid range we use, we first performed a rough normalization of the raw measurements. First, we subtracted the mean of each signal. Then, we divided each signal by its standard deviation. The InfoMax and FastICA algorithms are more efficient when the measured signals are sparse. We thus pre-filtered the inputs to these algorithms using the derivative operator $[-1\,0\,1]/2$. Our separation procedure was based on the BFGS Quasi-Newton algorithm as implemented in the MATLAB optimization toolbox (function FMINUNC).

The results of the simulations are presented in Table 1. The separation quality is given by the signal to interference ratio (SIR). The SIR is the energy of the signal divided by the energy of the interference:

$$\mathrm{SIR} = \min_k \left(\frac{\|\mathbf{s}_k\|^2}{\|\mathbf{s}_k - \hat{\mathbf{s}}_k\|^2} \right) \quad . \tag{51}$$

Note that Eq. (51) uses the signal k having the minimal ratio, i.e., having the worst separation quality.[4] After performing numerous simulations, we report the mean SIR and the standard deviation of the SIR. Clearly, Table 1 shows that practically no degradation of the separation quality is caused by our entropy approximation. On the other hand, the improvement in the run time is huge, compared to the competing non-parametric method. Our method does not compete with the parametric algorithms over run time, but it outperforms them in separation quality. We can separate signals that the parametric methods fail to handle.

In order to demonstrate the separation quality, we performed an additional set of separation simulations, this time based on 10 pictures. The

[4]As explained in App. A, the estimated $\hat{\mathbf{s}}_k$ is prone to permutation and scale ambiguities. Thus, Eq. (51) is applied to separation results which are compensated for these ambiguities.

Table 1: Simulation results: The accuracy of the separation is measured in terms of the signal to interference ratio (SIR).

Algorithm	SIR [dB]	Time
Non-parametric ICA with back propagation gradient computation	18 ± 4	**760** min
Non-parametric ICA with fast kernel convolution, using histograms of 1K bins	$\mathbf{22} \pm 3$	**1.2** min
Jade	$\mathbf{7} \pm 4$	0.2 sec
InfoMax	$\mathbf{1} \pm 0.5$	1.4 sec
InfoMax with pre-filtering	$\mathbf{8} \pm 4$	1.6 sec
Fast ICA	$\mathbf{4} \pm 4$	1.1 sec
Fast ICA with pre-filtering	$\mathbf{5} \pm 3$	1.9 sec

pictures where mixed using randomly generated full rank matrices (condition number$\leq$ 100). The separation results are presented in Fig. 5.

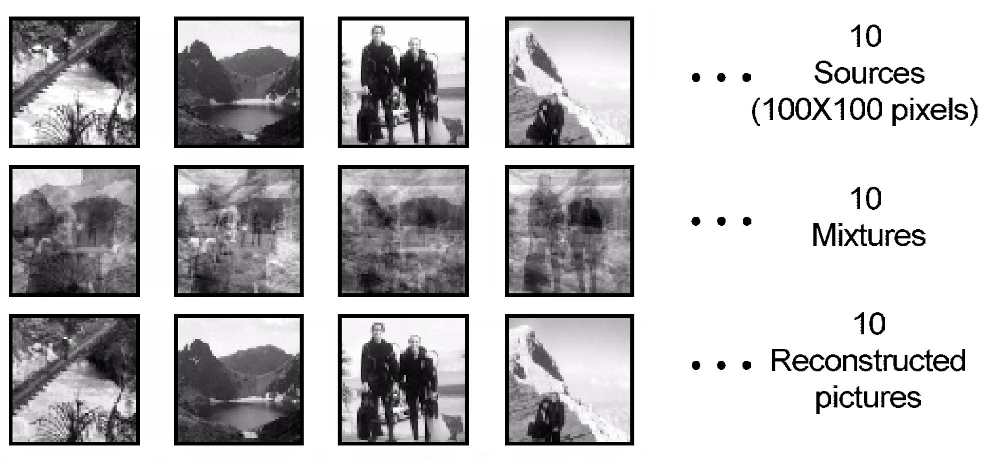


Figure 5: Four samples of a set of 10 pictures involved in the separation simulation. The mixed signals were pre-filtered by a derivative operator before the separation. The separation SIR is 20dB.

5 Conclusions

We have presented two techniques for accelerating the estimation of entropy its gradient using kernel methods. The first technique improves gradient computation using back propagation, and lowers the gradient complexity to be comparable to that of the entropy estimation itself. The second technique provides further acceleration using fast convolution, based on resampling (quantization) of signals to a uniform grid. This improves the complexity of estimating the entropy and its gradient from N^2 to $N \log N$.

The low computational cost of our algorithms makes non-parametric entropy estimation applicable to high dimensional problems and large sample sizes. We demonstrated this by applying our methods to the linear ICA problem, where both high separation performance and practical run times were achieved.

A Ambiguities in MI optimization

Optimization using MI as a cost function possesses three ambiguities: Permutation ambiguity, sign ambiguity, and scale ambiguity.

Permutation ambiguity

Let x, y be two signals, then

$$I_{x,y} = H_x + H_y - H_{x,y} = H_y + H_x - H_{y,x} = I_{y,x} \ . \tag{52}$$

Therefore, the reconstructed signals apears in no special order. This ambiguity does not concern us in this work.

Scale and sign ambiguity

Let x, y be two statistically independent signals. Then, their joint PDF is separable and equals

$$p_{x,y}(x, y) = p_x(x)p_y(y) \ . \tag{53}$$

Therefore, the MI of x, y is

$$I_{x,y} = \int_x \int_y p_{x,y}(x, y) \log \left[\frac{p_{x,y}(x, y)}{p_x(x)p_y(y)} \right] = \int_x \int_y p_{x,y}(x, y) \log(1) = 0 \ . \tag{54}$$

Denote $\bar{x} = \rho x$ and $\bar{y} = \tau y$. The joint PDF of $\bar{x}, \bar{y}$ is still separable and equals

$$p_{\bar{x},\bar{y}}(\bar{x}, \bar{y}) = p_{\bar{x}}(\bar{x}) p_{\bar{y}}(\bar{y}) \; . \tag{55}$$

Therefore, their MI is also zero. Assume that matrix $\mathbf{W}$ is a solution to the optimization problem, i.e it causes the MI of the reconstructed sources $\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_K$ to be zero. Denote $\mathbf{R}$ as a diagonal matrix whose diagonal terms are $r_{1,1}, \ldots, r_{K,K}$. Then, $\mathbf{RW}$ is also a solution to the optimization problem, causing the MI of $r_{1,1}\hat{\mathbf{s}}_1, \ldots, r_{K,K}\hat{\mathbf{s}}_K$ to be zero. Therefore the solution $\mathbf{W}$ is derived up to a scaling of each of its rows. The sign ambiguity is a special case of the scaling ambiguity, in which the scale is -1.

The scale ambiguity implies that we have infinitely many solutions to the separation problem. This ambiguity may cause the optimization algorithm to be unstable. In order to stabilize the algorithm, we add a penalty term that determines the scale of the estimated sources. We choose to force the norm of the estimates sources to be $\sqrt{N}$. This normalization solves only the scale ambiguity, but does not resolve the sign ambiguity. Nevertheless, the sign ambiguity leads to a finite number of solutions.

B Optimal window variance for Parzen-windows PDF estimation

Following [14], the optimal value for the effective width of the Parzen-window kernel is

$$\sigma_{\text{optimal}} = \kappa^{-2/5} \left\{ \int \varphi(t)^2 \, dt \right\}^{1/5} \left\{ p''(x)^2 \, dx \right\}^{-1/5} N^{-1/5} \; , \tag{56}$$

where p is the unknown PDF being estimated and

$$\kappa = \int t^2 \varphi(t) \, dt \; . \tag{57}$$

Unfortunately, Eq. (56) implies that the optimal kernel width depends on the PDF that we want to estimate. Therefore we cannot determine an optimal kernel width that will fit every arbitrary source.

The kernel width which we used in our simulations is $\sigma = 1.06 N^{-1/5}$. It is obtained for the special case where φ is a Gaussian window, and p is Gaussian PDF having a unit variance. In general, the optimal kernel width can be determined by a maximum likelihood optimization procedure (see [9]) prior to the separation optimization. As a rule of thumb, we recommend initialize this optimization from $\sigma = 1.06 N^{-1/5} \sigma_{\text{samples}}$, where σ_{samples} is the standard deviation of the signal samples.

References

[1] J. V. M. Bove, "Entropy-based depth from focus," *Journal of Optical Society of America A*, vol. 10, pp. 561–566, 1993.

[2] J. Principe and D. Erdogmus, "From adaptive linear to information filtering," in *Proc. IEEE Sympos. Adaptive Systems for Signal Processing, Communications and Control*, pp. 99–104, 2000.

[3] I. Santamaria, D. Erdogmus, and J. C. Principe, "Entropy minimization for supervised digital communications channel equalization," *IEEE Transactions on Signal Processing*, vol. 50, no. 5, pp. 1184–92, 2002.

[4] W. Schwartzkopf, B. Evans, and A. Bovik, "Entropy estimation for segmentation of multi-spectral chromosome images," in *Proc. IEEE Southwest Sympos. on Image Analysis and Interpretation.*, pp. 234–7, 2002.

[5] R. R. Wang, T. Huang, and Jialin-Zhong, "Generative and discriminative face modelling for detection," in *Proc. IEEE International Conf. Automatic Face Gesture Recognition.*, pp. 281–6, 2002.

[6] R. Boscolo, H. Pan, and V. P. Roychowdhury, "Non-parametric ICA," in *Proc. ICA2001*, pp. 13–18.

[7] Y. Lomnitz, *Efficient blind source separation using a semi-maximum likelihhod technique.* PhD thesis, Tel-Aviv University, Dept. Elect. Engineering-Systems, 2003.

[8] P. Thevenaz and M. Unser, "Optimization of mutual information for multiresolution image registration," *IEEE Transactions on Image Processing*, vol. 9, no. 12, pp. 2083–99, 2000.

[9] P. A. Viola, *Alignment by Maximization of mutual information.* PhD thesis, Massachusetts Institute of Technology - Artificial Intelligence Laboratory, 1995.

[10] R. O. Duda, P. E. Hart, and D. G. Stock, *Pattern classification.* NY: John Wiley and Sons, 2001.

[11] L. B. Rall, *Automatic differentiation: techniques and applications.* Springer-Verlag, 1981.

[12] A. Griewank, *Evaluating derivatives: principles and techniques of algorithmic differentiation.* Philadelphia: SIAM, 2000.

[13] B. A. Pearlmutter, "Fast exact multiplication by the hessian," *Neural Computation*, vol. 6, no. 1, pp. 147–160, 1994.

[14] B. Silverman, *Density estimation for statistics and data analysis.* NY: Chapman and Hall, 1986.

[15] J. I. de la Rosa and G. Fleury, "On the kernel selection for minimum-entropy estimation," in *Proc. IEEE Instrumentation and Measurement Technology Conference.*, vol. 2, pp. 1205–10, 2002.

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

[17] A. Hyvärinen, "The Fast-ICA MATLAB package," 1998. http://www.cis.hut.fi/˜aapo/.

[18] D. Pham and P. Garrat, "Blind separation of a mixture of independent sources through a quasi-maximum likelihood approach," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1712–1725, 1997.

[19] J.-F. Cardoso and A. Souloumiac, “Blind beamforming for non Gaussian signals,” *IEE Proc.-F*, vol. 140, pp. 362–370, dec 1993.

[20] D. T. Pham, “Fast algorithm for estimating mutual information, entropies and score functions,” in *Proc. ICA2003*, pp. 17–22.

[21] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent component analysis.* NY: John Wiley and Sons, 2001.

[22] S. Makeig, A. Bell, T.-P. Jung, and T. Sejnowski, “Independent component analysis of electroencephalographic data,” *Advances in Neural Information Processing Systems 8*, pp. 145–151, 1996.