Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Signal Processing 85 (2005) 1045–1058
SIGNAL PROCESSING
www.elsevier.com/locate/sigpro

Fast kernel entropy estimation and optimization

Sarit Shwartz*, Michael Zibulevsky, Yoav Y. Schechner

Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel

Received 30 April 2004; received in revised form 19 September 2004

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 variables, for efficient optimization, sensitivity analysis, etc. Entropy estimation can be based on an estimate of the probability density function, which is computationally costly if done naively. Some prior algorithms use computationally efficient non-parametric entropy estimators. However, differentiation of the previously proposed estimators is difficult and may even be undefined. To counter these obstacles, we consider non-parametric kernel entropy estimation that is differentiable. We present two different accelerated kernel algorithms. The first accelerates the entropy gradient calculation based on a back propagation principle. It allows calculating the differential entropy gradient in the same complexity as that of calculating the entropy itself. The second algorithm accelerates the estimation of both entropy and its gradient by using fast convolution over a uniform grid. As an example, we apply both algorithms to blind source separation.

Keywords: Non-parametric entropy estimation; Parzen windows; Automatic differentiation; Blind source separation; Independent component analysis; Mutual information

1. Introduction

Differential entropy (DE) is used as a quality criterion in various signal processing problems such as segmentation, detection, source separation, image registration, channel equalization, neural networks and estimating depth from focus [2,3,10,19,25,27,31,35,37,39]. Some algorithms estimate the entropy using rough parametric models for the probability density function (PDF). The use of a low-dimensional vector to parameterize the PDF reduces the complexity of calculating the DE. However, this may lead to inaccurate solutions. Therefore we concentrate here on non-parametric kernel estimators, also called *Parzen windows* estimators [33,37]. These estimators may have a large complexity if implemented naively. This complexity can be reduced if the estimation is implemented by on-line stochastic gradient

*Corresponding author.
E-mail addresses: psarit@tx.technion.ac.il (S. Shwartz), mzib@ee.technion.ac.il (M. Zibulevsky), yoav@ee.technion.ac.il (Y.Y. Schechner).

0165-1684/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2004.11.022

algorithms [11,30]. However, stochastic methods may have a larger asymptotic error than batch methods. Thus in this work we focus on batch methods, which rely on all the samples for which the entropy is estimated.

Often, for efficient optimization and sensitivity analysis it is beneficial to exploit the gradient rather than solely relying on global optimization methods [20]. Therefore, the differentiability of the cost function is very important. It is worth noting that there are efficient non-parametric methods for DE estimation with $\mathcal{O}(N \log N)$ complexity [13,14], where N is the sample size. However, those methods are based on building tree structures, which are discrete and depend on the input samples for which the DE is estimated. In some applications, the samples are *not* constant data but *variables*. As optimization progresses, the variable values are modified iteratively. This may change the tree structure abruptly during iterations. Hence formulation of derivatives of the DE estimator is hindered. Additional efficient entropy estimators are based on *order statistics* [18,36], however, they are based on sample sorting, which hinders differentiation as well.

To counter these problems, we present two methods to bypass the computational load of estimating the kernel entropy *and its gradient*. In Section 3, we describe a general method for accelerating the DE gradient calculation. We use an approach in the spirit of the *back propagation* algorithm, which is used for neural network training (see for example Ref. [9]). This principle is known as *backward packing* in the community of automatic differentiation [15,22,26]. Applying this principle here allows calculation of the entropy gradient with the same complexity needed to calculate the entropy itself.

In Sections 4 and 5, we propose a method which accomplishes our task in a complexity of $\mathcal{O}(N \log N)$. It is inspired by an approximation of the PDF [12,32,38]. We extend this approximation to the task of estimating the entropy and its gradient. We do so in a way that reduces numerical problems during optimization. The approximation of the kernel estimator 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 problems involving large sample sets.

Finally, we apply the algorithms to linear blind source separation (BSS) problems, also called independent component analysis (ICA) [5,6]. ICA can be 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 Section 6. Partial results were presented in Ref. [34].

2. Background

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

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

where we denote the vector of measurements as $\mathbf{y}$, while n is an index of a sample of $\mathbf{s}$. Here, $\mathbf{w}$ is the vector of parameters of the function f. This vector has a dimension of $N_{\dim}$. In the trivial case of calculating the entropy of some measurements, $f(n; \mathbf{y}; \mathbf{w}) = y(n)$, thus $s(n) = y(n)$ and $N_{\dim} = 0$.

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

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

where $\varphi(t)$ is a smoothing kernel. There are several options for selecting the kernel [8,33,35]. We use a Gaussian[1] 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 Ref. [33]. The Parzen-windows entropy estimator [2,37] for a signal $\mathbf{s}$ is

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

Define N_f as the number of operations needed to calculate $f(n; \mathbf{y}; \mathbf{w})$ using Eq. (1). We need to

[1] Our analysis is by no means limited to the Gaussian kernel, but can be applied to any differentiable kernel.

calculate $f(n;\mathbf{y};\mathbf{w})$ for all N samples of $\mathbf{s}$. In addition, Eq. (4) requires N^2 calculations of φ. Thus, the overall complexity of the Parzen-windows entropy estimator is

$$\mathcal{O}_{\text{entropy}}^{\text{explicit}} = \mathcal{O}(N^2 + N_f N). \tag{5}$$

Define the N_{dim}-dimensional 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 Parzen-windows entropy with respect to $\mathbf{w}$ is

$$\nabla_{\mathbf{w}} \hat{\mathscr{H}}_s = -\frac{1}{N}\sum_{l=1}^{N} \frac{\sum_{n=1}^{N} \varphi'(S_{l,n}) S_{l,n} \mathbf{G}(l,n;\mathbf{y};\mathbf{w})}{\sum_{n=1}^{N} \varphi[\mathbf{s}(l) - \mathbf{s}(n)]}, \tag{7}$$

where φ' is the derivative of φ. Here

$$S_{l,n} \equiv [s(l) - s(n)] \tag{8}$$

and

$$\mathbf{G}(l,n;\mathbf{y};\mathbf{w}) \equiv [\mathbf{g}(l;\mathbf{y};\mathbf{w}) - \mathbf{g}(n;\mathbf{w};\mathbf{y})]. \tag{9}$$

Eq. (7) has two explicit nested summations, for each of the N_{dim} components of $\mathbf{G}(l,n;\mathbf{y};\mathbf{w})$. In addition, let N_g be the number of operations needed for calculating $\mathbf{g}(n;\mathbf{y};\mathbf{w})$ per sample n. Thus the overall complexity of estimating the entropy gradient is

$$\mathcal{O}_{\text{gradient}}^{\text{explicit}} = \mathcal{O}(N_{\text{dim}} N^2 + N_g N), \tag{10}$$

if this estimation is based explicitly on Eq. (7). Often, a very large number of samples is used, while f and g are simple. We focus in this paper on such scenarios, i.e. where $N \gg N_f, N_g$. Thus, $\mathcal{O}_{\text{gradient}}^{\text{explicit}} \approx N_{\text{dim}} \mathcal{O}_{\text{entropy}}^{\text{explicit}}$.

3. Entropy gradient via back propagation

In Section 2 we showed that direct implementation of the entropy gradient is more computationally expensive than direct implementation of the DE itself. In this section we describe a method that allows calculation of the gradient as efficiently as calculating the DE. This method is applicable to differentiable functions. We start by presenting the general method in Section 3.1. Then, we apply the method to the calculation of the entropy gradient in Section 3.2.

3.1. Gradient calculation using back propagation

The back propagation technique [15,22,26] allows calculation of a gradient of a differentiable function with the same complexity of calculating the function itself. First, 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

$$f(x_1, x_2, x_3) = \Phi[\Psi(x_1), \Theta(x_2 + x_3)] \tag{11}$$

is described using the slack variables $z \equiv x_1 + x_2$, $\zeta_1 \equiv \Psi(x_1)$, $\zeta_2 \equiv \Theta(z)$, 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. (11) is illustrated in Fig. 1a. The function is calculated by forward propagation through the graph, where the slack variables represent inner graph layers.

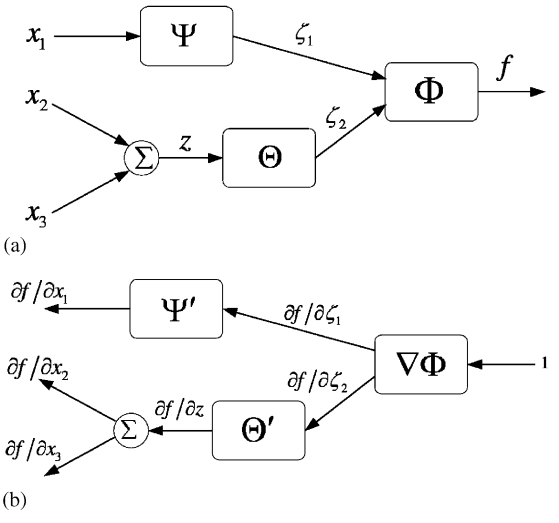

Fig. 1. Graphs of the forward (a) and back (b) propagation, corresponding to the function in the example of Section 3.1.

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 $\partial\Psi_{x_1}/\partial x_1$, which is the coefficient of the differential $\mathrm{d}\zeta_1 = (\partial\Psi_{x_1}/\partial x_1)\,\mathrm{d}x_1$.

We now show that we can calculate the value of the gradient by the back propagation algorithm, in a similar manner to neural network training. 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.

3.2. 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

$$p(l) \equiv (1/N)\sum_{n=1}^{N} \varphi[s(l) - s(n)]. \tag{12}$$

Define

$$L[p(l)] \equiv \log[p(l)]. \tag{13}$$

The atom functions we use are φ and L.

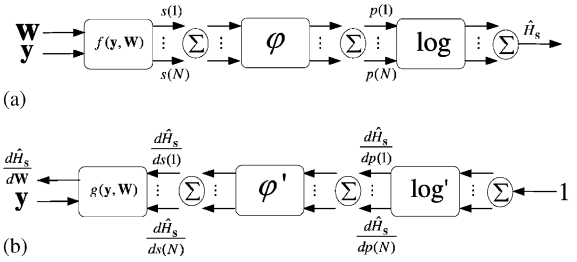

Fig. 2. Graphs for forward (a) and back (b) propagation for estimating the entropy and its gradient.

The directional graph describing the entropy calculation is illustrated in Fig. 2. In order to calculate the entropy value, we use forward propagation. It constitutes the following consecutive steps:

(a) $\forall l \in \{1,\ldots,N\}$,
$s(l) = f(l;\mathbf{y};\mathbf{w})$; save $\mathbf{g}(l;\mathbf{w};\mathbf{y})$,
(b) $\forall l \in \{1,\ldots,N\}$,

$$p(l) = \frac{1}{N}\sum_{n=1}^{N} \varphi(S_{l,n}); \text{ save } \varphi'(S_{l,n}),$$

(c) $$\mathscr{H}_{\mathbf{s}} = -\frac{1}{N}\sum_{l=1}^{N} L[p(l)];$$

$$\text{save } L'[p(l)] = [p(l)]^{-1}, \tag{14}$$

where L' is the derivative of L. Note that whenever we calculate one of the slack variables in the forward propagation, we also calculate its gradient with respect to its direct inputs. The complexity of step (a) is $\mathscr{O}[N(N_g + N_f)]$. The complexity of step (b) is $\mathscr{O}(N^2)$. The complexity of step (c) is $\mathscr{O}(N)$. Therefore, the overall complexity of forward propagation is

$$\begin{aligned}\mathscr{O}_{\text{forward}} &= \mathscr{O}[N(N_g + N_f) + N^2 + N] \\ &= \mathscr{O}[N(N_g + N_f) + N^2].\end{aligned} \tag{15}$$

After we finish calculating the entropy value, we calculate the gradient by back propagation of differentials. The differentials of the entropy and all the slack variables used in

Eq. (14) are

$$\text{(a) } \forall l \in \{1,\dots,N\},\ \mathrm{d}s(l) = \langle g(l;\mathbf{w};\mathbf{y}), \mathrm{d}\mathbf{w}\rangle,$$

$$\text{(b) } \forall l \in \{1,\dots,N\},$$

$$\mathrm{d}p(l) = \frac{1}{N}\sum_{n=1}^{N} \varphi'(S_{l,n})[\mathrm{d}s(l) - \mathrm{d}s(n)],$$

$$\text{(c) } \mathrm{d}\mathscr{H}_{\mathbf{s}} = -\frac{1}{N}\sum_{l=1}^{N} L'[p(l)]\mathrm{d}p(l). \tag{16}$$

The back propagation is equivalent to substitution of Eq. (16) in a reversed order

$$\text{(i) } \forall l \in \{1,\dots,N\},\ \frac{\partial}{\partial p(l)}\mathscr{H}_{\mathbf{s}} = L'[p(l)],$$

$$\text{(ii) } \forall l \in \{1,\dots,N\},$$

$$\frac{\partial}{\partial s(l)}\mathscr{H}_{\mathbf{s}} = \frac{1}{N}\sum_{n=1}^{N} \varphi'(S_{l,n})\left[\frac{\partial}{\partial p(l)}\mathscr{H}_{\mathbf{s}}\right],$$

$$\text{(iii) } \forall n \in \{1,\dots,N\},$$

$$\frac{\partial}{\partial s(n)}\mathscr{H}_{\mathbf{s}} = \frac{-1}{N}\sum_{n=1}^{N} \varphi'(S_{l,n})\left[\frac{\partial}{\partial p(l)}\mathscr{H}_{\mathbf{s}}\right],$$

$$\text{(iv) } \nabla_{\mathbf{w}}\mathscr{H}_{\mathbf{s}} = \sum_{l=1}^{N} \mathbf{g}(l;\mathbf{w};\mathbf{y})\left[\frac{\partial}{\partial s(l)}\mathscr{H}_{\mathbf{s}}\right]. \tag{17}$$

In Eq. (17), the complexity of step (i) is $\mathcal{O}(N)$. The complexity of steps (ii) and (iii) is $\mathcal{O}(N^2)$. The complexity of step (iv) is $\mathcal{O}(N_{\dim}N)$. Therefore, the overall complexity of back propagation is

$$\begin{aligned}\mathcal{O}_{\text{back}} &= \mathcal{O}(N^2 + N_{\dim}N + N)\\ &= \mathcal{O}(N^2 + N_{\dim}N).\end{aligned} \tag{18}$$

The complexity $\mathcal{O}_{\text{back}}$ is similar to $\mathcal{O}_{\text{forward}}$. Therefore, combining all the algorithm steps yields estimation of both the entropy and its gradient in a complexity of

$$\begin{aligned}&\mathcal{O}^{\text{gradient}}_{\text{backpropagation}}\\ &\quad= \mathcal{O}_{\text{forward}} + \mathcal{O}_{\text{back}}\\ &\quad= \mathcal{O}[N^2 + N(N_{\dim} + N_g + N_f)].\end{aligned} \tag{19}$$

This complexity is less expensive than the complexity of Eq. (10).

Note that the structure of the graphs in Figs. 1 and 2 does not depend on the optimization variables, e.g. **w** and **s**. Had it been otherwise, it would not have been possible to define differentials corresponding to edges of the graphs. Therefore back propagation is difficult to apply to methods like [13,14]: those methods do not maintain a constant graph structure during the optimization of the entropy of **s**.

Input: $\mathbf{w}$, $\mathbf{s}$
Output: $\mathcal{H}_{\mathbf{s}}$,$\nabla_{\mathbf{w}}\mathcal{H}_{\mathbf{s}}$
Algorithm:

```
For l = 1 to N
    s(l) = f(l; y; w)
    g(l; w; y) = ∇_w f(l; y; w)
end
For l = 1 to N
    For n = 1 to N
        p(l) = p(l) + φ[s(l) − s(n)]/N
        p'(l) = φ'[s(l) − s(n)]/N
    end
    H_s = H_s − log[p(l)]
    L'(l) = 1/p(l)
    For n = 1 to N
        ∂/∂s(l) H_s = ∂/∂s(l) H_s − p'(l)L'(l)/N
        ∂/∂s(n) H_s = ∂/∂s(n) H_s + p'(l)L'(l)/N
    end
end
For l = 1 to N
    ∇_w H_s = ∇_w H_s + g(l; w; y) [∂/∂s(l) H_s]
end
```

Fig. 3. Pseudo-code for calculating the entropy gradient via back propagation.

We summarize the algorithm in a pseudo-code (Fig. 3). This approach is general. Thus, a reduction of the complexity of the entropy estimator (as we describe in the next section), implies a similar reduction in the gradient complexity.

4. DE estimation in N log N

In this section, we develop a non-parametric kernel entropy estimator that has a complexity of $\mathcal{O}(N \log N)$. In contrast to other $\mathcal{O}(N \log N)$ estimators [13,14,36], this estimator is differentiable with respect to the input samples. We

approximate the entropy using a convolution on a resampled version of the signal, as had been done for PDF estimation in Refs. [12,32,38]. This formulation leads in Section 5 to an efficient approximation to the entropy gradient, which is both numerically stable and has $\mathcal{O}(N \log N)$ complexity.

Calculating the DE using Eq. (4) requires N^2 calculations of φ. Consider, however, the inner sum of Eq. (4), which is the PDF estimator (Eq. (2)). This sum can be seen as a convolution

$$\hat{p}(t|\mathbf{s}) = \hat{P} * \varphi, \tag{20}$$

where

$$\hat{P}(t) = (1/N) \sum_{n=1}^{N} \delta[t - s(n)]. \tag{21}$$

It is known that fast convolution can be performed in $\mathcal{O}(N \log N)$ operations if done over a *uniform grid*. Therefore, we resample (interpolate) $\hat{P}(t)$ to a uniform grid. Then we convolve it with a uniformly sampled version of φ which we denote φ_{sampled}. Finally, we interpolate the results back to the set of points $s(l)$. This process is illustrated in Fig. 4.

Resampling $\hat{P}$ starts by defining a vote function v on a uniform grid of length M, with a step size of

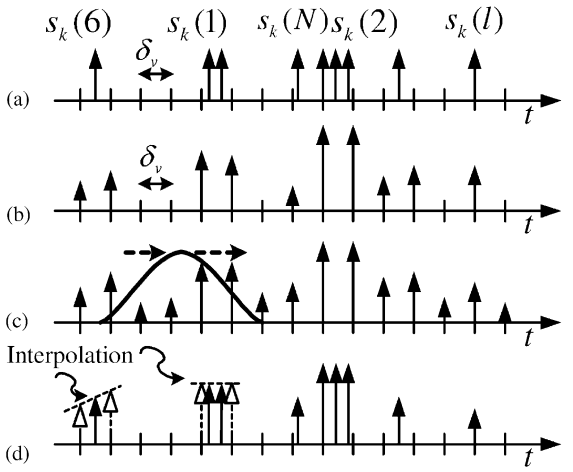

Fig. 4. Efficient estimation of a PDF: (a) the function $\hat{P}$; (b) resampling on a uniform grid; (c) discrete convolution with a sampled kernel; (d) interpolation to the original samples. The latter step makes the subsequent entropy estimation more accurate.

δ_v. Let $m^{\#}$ be the index of the grid node closest to the value of $s(n)$, that satisfies

$$m^{\#} \leqslant s(n)/\delta_v \leqslant m^{\#} + 1. \tag{22}$$

Define the distance of the signal value $s(n)$ from the index $m^{\#}$ (normalized by δ_v) as

$$\eta = \frac{s(n)}{\delta v} - m^{\#}. \tag{23}$$

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

$$h(1-\eta) = 1 - h(\eta) \quad 0 \leqslant \eta \leqslant 1. \tag{24}$$

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

$$v(m) \leftarrow \begin{cases} v(m) + h(\eta) & \text{for } m = m^{\#}, \\ v(m) + 1 - h(\eta) & \text{for } m = m^{\#} + 1, \end{cases} \tag{25}$$

where η is given by Eq. (23). After the voting is over, resulting in a vector $\mathbf{v}$, we associate the resampled $\hat{P}$ with $\mathbf{v}/N$. This transfers the function illustrated in Fig. 4a to the function illustrated in Fig. 4b. Then, following Eq. (20), we convolve[3] $\mathbf{v}/N$ with φ_{sampled} (Fig. 4b,c),

$$\hat{p}_{\text{quant}} = (\mathbf{v}/N) * \varphi_{\text{sampled}}. \tag{26}$$

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

$$\tilde{\mathcal{H}}_{\mathbf{s}} = \sum_{m=1}^{N_{\text{bins}}} \hat{p}_{\text{quant}}(m) \log[\hat{p}_{\text{quant}}(m)]. \tag{27}$$

However, the use of discrete binning creates 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 $\hat{p}[s(l)|\mathbf{s}]$ by interpolating p_{quant} onto the points $s(l)$, using the same interpolation function $h(\eta)$ as before (Fig. 4c,d)

$$\hat{p}[s(l)|\mathbf{s}] = h(\eta)\hat{p}_{\text{quant}}(m^{\#}) + [1 - h(\eta)]\hat{p}_{\text{quant}}(m^{\#} + 1). \tag{28}$$

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

[3]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.

Finally, the estimate of the DE is calculated by

$$\hat{\mathscr{H}}_{\mathbf{s}} = -(1/N)\sum_{n=1}^{N} \log\{\hat{p}[s(l)|\mathbf{s}]\}. \tag{29}$$

The voting, the interpolation and the entropy calculation (Eqs. (25), (28) and (29)) require $\mathcal{O}(N)$ operations. The convolution (Eq. (26)) requires $\mathcal{O}(M \log N_{\text{kernel}})$ operations,[4] where N_{kernel} is the length of φ_{sampled}. In addition, calculating the sources (Eq. (1)) requires $\mathcal{O}(NN_f)$ operations. Therefore, the overall complexity of calculating the DEs is

$$\mathcal{O}^{\text{entropy}}_{\text{approx}} = \mathcal{O}(N \log N + NN_f). \tag{30}$$

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

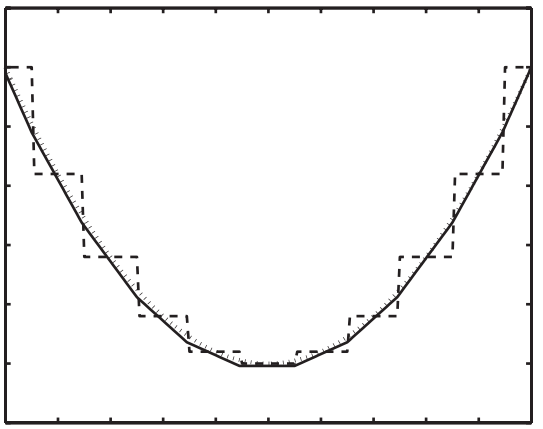

Fig. 5. Quantization of functions and derivatives. A continuous function (dotted). A quantized version of the function (dashed). A version of the function, based on a quantized derivative (solid).

5. Estimation of the entropy gradient

5.1. Drawbacks of an intuitive approach

We considered several approaches for estimating the gradient of the DE in $\mathcal{O}(N \log N)$. The most intuitive approach is to differentiate the entropy approximation that we derived in Section 4 and calculate it efficiently using back propagation. However, as we already mentioned, discrete binning causes fluctuations in the entropy value. Fig. 5 crudely illustrates the effect of quantization on optimization. Three curves are presented: the dotted curve depicts a continuous function with a single global minimum. The dashed curve depicts a quantized version of this function. Differentiating the quantized curve amplifies the quantization noise. This might hinder the convergence of gradient based optimization.

We avoid this problem by taking a different approach. Rather than differentiating an approximation based on quantization, we elect to approximate the DE derivatives directly. The solid curve in Fig. 5 depicts a version of the dotted curve, that has quantized values of the derivative. Clearly, quantization of the derivative itself does not corrupt the function as much. Thus optimization based on the quantized derivative suffers less from quantization noise amplification.

[4]Typically the number of bins M and the kernel support N_{kernel} are of the order of N or smaller. Therefore, the complexity needed is at most $\mathcal{O}(N \log N)$.

5.2. Distinct estimation of the gradient

The entropy gradient is calculated using a chain rule. First, we derive the entropy gradient with respect to the signal samples. Then, we calculate the entropy gradient with respect to the desired parameters by

$$\nabla_{\mathbf{w}}\mathscr{H}_{\mathbf{s}} = \frac{1}{N}\sum_{l=1}^{N} \mathbf{g}(l;\mathbf{w};\mathbf{y})\left[\frac{\partial\mathscr{H}_{\mathbf{s}}}{\partial s(l)}\right], \tag{31}$$

where $\mathbf{g}(l;\mathbf{w};\mathbf{y})$ is given in Eq. (6). The entropy derivatives in Eq. (31) are given by

$$\begin{aligned}\frac{\partial\mathscr{H}_{\mathbf{s}}}{\partial s(r)} &= -\frac{1}{N}\sum_{l=1}^{N}\frac{\frac{1}{N}\sum_{n=1}^{N}\varphi'(S_{l,n})(\delta_{l,r}-\delta_{n,r})}{\frac{1}{N}\sum_{n=1}^{N}\varphi(S_{l,n})}\\ &= -\frac{1}{N}\frac{\frac{1}{N}\sum_{n=1}^{N}\varphi'(S_{r,n})}{\hat{p}[s(r)|\mathbf{s}]}\\ &\quad +\frac{1}{N}\sum_{l=1}^{N}\frac{\frac{1}{N}\varphi'(S_{l,r})}{\hat{p}[s(l)|\mathbf{s}]},\end{aligned} \tag{32}$$

where $\delta_{l,r}$ is the Kroneker delta and $\hat{p}[s(l)|\mathbf{s}]$ is defined in Eq. (20).

Until this stage the gradient equations have been accurate analytical formulae. Now we will derive a fast approximation to these formulae using FFT. Define

$$\Phi'[s(l)|\mathbf{s}] \equiv (1/N)\sum_{n=1}^{N}\varphi'[s(l)-s(n)] \tag{33}$$

and

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

Eq. (32) can be written as

$$\frac{\partial \mathscr{H}_{\mathbf{s}}}{\partial s(l)} = \frac{1}{N}\frac{\Phi'[s(l)|\mathbf{s}]}{\hat{p}[s(l)|\mathbf{s}]} - F'[s(l)]. \tag{35}$$

Note that $\hat{p}[s(n)|\mathbf{s}]$ is known, since we had calculated it with the DE itself, prior to the gradient calculation.

Recall that Eq. (2) is represented by Eq. (20). In analogy, Eq. (33) is equivalent to

$$\Phi'(t|\hat{\mathbf{s}}_k) = \hat{P} * \varphi', \tag{36}$$

where $\hat{P}$ is given by Eq. (21), and $\varphi'^{\text{mirror}}(t) = \varphi'(-t)$. We compute a discrete approximation to Eq. (36) using the array $\mathbf{v}$ (which is $\hat{P}$, uniformly resampled by Eq. (25))

$$\hat{\Phi}'_{\text{quant}} = (\mathbf{v}/N) * \varphi'_{\text{sampled}}. \tag{37}$$

Here $\varphi'_{\text{sampled}}$ is a sampled version of φ'. Then, we interpolate $\hat{\Phi}'_{\text{quant}}$ to the set of points $s(l)$. We do so similarly to Eq. (28).

In a somewhat analogous manner, Eq. (34) is equivalent to

$$F'(t|\hat{\mathbf{s}}_k) = (\hat{P}/\hat{p}) * \varphi'^{\text{mirror}}. \tag{38}$$

It is approximated by uniformly resampling $\hat{P}/\hat{p}$: similarly to Eq. (25), we define a weighted vote function v_w on the uniform quantization grid. For each sample $s(l)$, we update the voting by

$$v_w(m) \leftarrow \begin{cases} v_w(m) + \dfrac{h(\eta)}{\hat{p}[s(n)|\mathbf{s}]} & \text{for } m = m^{\#}, \\ v_w(m) + \dfrac{1-h(\eta)}{\hat{p}[s(n)|\mathbf{s}]} & \text{for } m = m^{\#}+1, \end{cases} \tag{39}$$

where $\hat{p}[s(n)|\mathbf{s}]$ had been computed in Eq. (28), while $m^{\#}$ and η are defined in Section 4. We associate $\mathbf{v}_w/N$ with the resampled $\hat{P}/\hat{p}$. We then define a sampled version of φ'^{mirror}, termed $\varphi'^{\text{mirror}}_{\text{sampled}}$. We thus imitate Eq. (38) by

$$\hat{F}'_{\text{quant}} = (\mathbf{v}_w/N) * \varphi'^{\text{mirror}}_{\text{sampled}}. \tag{40}$$

Finally, we interpolate $\hat{F}'_{\text{quant}}$ to the set of points $s(l)$, similarly to Eq. (28).

Recall from Section 4 that the complexity of the voting and the interpolation is $\mathcal{O}(N)$, while the complexity of the discrete convolution is $\mathcal{O}(N \log N)$. Moreover, the complexity of Eq. (31) is $\mathcal{O}(N_{\text{dim}}N + N_g N)$, while the complexity of Eqs. (33)–(35) is $\mathcal{O}(N)$. Thus the overall complexity of calculating the DE gradient is

$$\mathcal{O}^{\text{gradient}}_{\text{approx}} = \mathcal{O}[N \log N + N(N_{\text{dim}} + N_g)]. \tag{41}$$

This complexity is significantly smaller than Eq. (10). Moreover, in the common case for which the number of degrees of freedom N_{dim} is much smaller than the sample size N, a significant benefit is achieved even relative to Eq. (19). A pseudo-code for the estimator and of the DE and its gradient is given in Fig. 6.

Note that the formulation up to Eq. (38) is made up of analytic formulae of the kernel entropy gradient. The quantization is done only *after* the differentiation. Therefore, there is no amplification of quantization noise. The benefit with respect to numerical performance is demonstrated in Fig. 7. It presents the norm of the gradient during optimization for two implementations run on an example. The solid curve presents the gradient calculated using the implementation considered in Section 5.1. The dashed curve presents the gradient calculated by the implementation described in Section 5.2. Both implementations used identical data and optimization parameters. The former approach did not converge to the desired minimum. The reached norm of the gradient was higher by two orders of magnitude than the expected quantization error. On the other hand, the optimization based on the approximation of the entropy gradient converged quickly to a lower gradient value, which is in the order of magnitude of the expected quantization error. Moreover, we

Input: $\mathbf{w}$, $\mathbf{y}$
Output: $\mathcal{H}_{\mathbf{s}}$, $\nabla_{\mathbf{w}}\mathcal{H}_{\mathbf{s}}$
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 $m^{\#}, \eta$
  $\hat{p}[s(l)|\mathbf{s}] = h(\eta)\hat{p}_{\text{quant}}(m^{\#}) + [1-h(\eta)]\hat{p}_{\text{quant}}(m^{\#}+1)$
  $\Phi'[s(l)|\mathbf{s}] = h(\eta)\Phi'_{\text{quant}}(m^{\#}) + [1-h(\eta)]\Phi'_{\text{quant}}(m^{\#}+1)$
  $\mathcal{H}_{\mathbf{s}} = \mathcal{H}_{\mathbf{s}} - \log\{\hat{p}[s(l)|\mathbf{s}]\}/N$
  $v_w(m^{\#}) = v_w(m^{\#}) - h(\eta)/(\hat{p}[s(l)|\mathbf{s}]N)$
  $v_w(m^{\#}+1) = v_w(m^{\#}+1) - [1-h(\eta)]/(\hat{p}[s(l)|\mathbf{s}]N)$
  $\frac{\partial}{\partial s(l)}\mathcal{H}_{\mathbf{s}} = \frac{\partial}{\partial s(l)}\mathcal{H}_{\mathbf{s}} - \Phi'[s(l)|\mathbf{s}]/(\hat{p}[s(l)|\mathbf{s}]N)$
end
$F'_{\text{quant}} = v_w * \varphi'_{\text{sampled}}$
For $l = 1$ to N
  load $m^{\#}, \eta$
  $F'[s(l)|\mathbf{s}] = h(\eta)F'_{\text{quant}}(m^{\#}) + [1-h(\eta)]F'_{\text{quant}}(m^{\#}+1)$
  $\frac{\partial}{\partial s(l)}\mathcal{H}_{\mathbf{s}} = \frac{\partial}{\partial s(l)}\mathcal{H}_{\mathbf{s}} + F'[s(l)|\mathbf{s}]$
end
For $l = 1$ to N
  $\nabla_{\mathbf{w}}\mathcal{H}_{\mathbf{s}} = \nabla_{\mathbf{w}}\mathcal{H}_{\mathbf{s}} + \mathbf{g}(l;\mathbf{w};\mathbf{y})\left[\frac{\partial}{\partial s(l)}\mathcal{H}_{\mathbf{s}}\right]$
end

Fig. 6. Pseudo-code for calculating the entropy and its gradient via fast kernel convolutions.

note that the final value of the cost function (beside the gradient of this function) was improved by the method described in Section 5.2.

6. ICA using fast kernel entropy optimization

MI of signals is a natural criterion for statistical dependency and is thus used in ICA algorithms (see for example in Refs. [2,17,23,28,29] and references therein). MI is based on estimates of entropies of signals, and therefore we use it as an application for our efficient optimization algorithm.

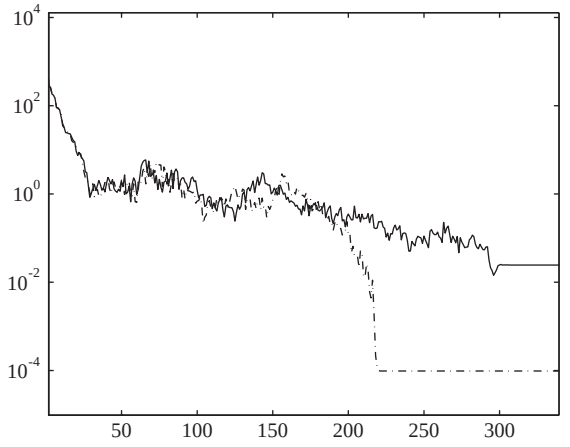

Fig. 7. The consequence of signal quantization (binning) on the optimization process of the entropy. The norm of the DE gradient is plotted as a function of the optimization iteration for: the gradient of the entropy approximation (solid). The approximation of the entropy gradient (dashed).

6.1. ICA and mutual information

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

$$[\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \ldots, \hat{\mathbf{s}}_K]^{\mathrm{T}} = \mathbf{W}[\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K]^{\mathrm{T}}. \tag{42}$$

The goal of ICA is to find the separation matrix $\mathbf{W}$ that yields estimated sources that are independent, thus inverting the mixing process. The independence criterion is the MI of the estimated sources. The MI of the K random variables $\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \ldots, \hat{\mathbf{s}}_K$ is (see for example Ref. [17])

$$\mathscr{I}(\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \ldots, \hat{\mathbf{s}}_K) = \mathscr{H}_{\hat{\mathbf{s}}_1} + \cdots + \mathscr{H}_{\hat{\mathbf{s}}_K} - \log|\det(\mathbf{W})| - \mathscr{H}_{\text{measurements}}, \tag{43}$$

where $\mathscr{H}(\hat{\mathbf{s}}_k)$ is the entropy of $\hat{\mathbf{s}}_k$. Here $\mathscr{H}_{\text{measurements}}$ is independent of $\mathbf{W}$ and is thus constant for a given measurements set $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$. For this reason, we ignore $\mathscr{H}_{\text{measurements}}$ in the optimization process. Thus,

the minimization problem that we solve is

$$\min_{\mathbf{w}}\left\{\sum_{k=1}^{K}\mathscr{H}_{\hat{\mathbf{s}}_k}-\log|\det(\mathbf{W})|+\lambda E_{\text{normalization}}\right\}, \tag{44}$$

where

$$E_{\text{normalization}}\equiv\sum_{k=1}^{K}\left(\frac{\|\hat{\mathbf{s}}_k\|}{\sqrt{N}}-1\right)^2 \tag{45}$$

penalizes for un-normalized sources. This penalty, weighted by a constant λ resolves ambiguities arising from the scale invariance of MI.[5] The complexity of calculating a signal norm is N. Therefore, the complexity of Eq. (45) is $\mathscr{O}_{\text{normalization}}=\mathscr{O}(KN)$.

The gradient of Eq. (45) is

$$\frac{2}{\sqrt{N}}\sum_{k=1}^{K}\left(\frac{1}{\sqrt{N}}-\frac{1}{\|\hat{\mathbf{s}}_k\|}\right)\left[\sum_{n=1}^{N}\hat{\mathbf{s}}_k(n)\Upsilon_{k,n}\right], \tag{46}$$

where $\Upsilon_{k,n}$ is a $K\times K$ matrix, all of whose rows are zeros except the kth row. That row equals $[y_1(n),\ldots,y_K(n)]$. Eq. (46) has two nested summations over matrices, each having K non-zero elements. Therefore, Eq. (46) has a complexity of $\mathscr{O}_{\text{gradient}}^{\text{normalization}}=\mathscr{O}(K^2N)$. The gradient of $\log|\det(\mathbf{W})|$ is (for example see Ref. [17]).

$$\nabla_{\mathbf{w}}[\log|\det(\mathbf{W})|]=(\mathbf{W}^{-1})^{\mathrm{T}}. \tag{47}$$

The only terms in Eq. (43) that remain to be addressed in the optimization formulation are the entropies of the estimated sources $\mathscr{H}_{\hat{\mathbf{s}}_k}$.

6.2. Optimization of MI

Many existing ICA algorithms have assumed rough models for the signals' PDFs (see Refs. [1,16,24]) or used high-order cumulants instead of MI (see Ref. [4]). These approximations can sometimes lead to failure, as demonstrated in Ref. [2], as well as in our subsequent example. In contrast, robust source separation can be achieved with non-parametric entropy estimation. An existing non-parametric ICA method is based on *order statistics* (Ref. [18]). It has an $\mathscr{O}[(K^2/2)N\log N]$ complexity, yet has two major drawbacks. First, this estimator is not differentiable. Therefore the optimization is done by exhaustive search, which is computationally costly in high dimensions. Second, in order to achieve this complexity, pre-whitening of the data was used with consequent estimation of an orthogonal separation matrix. This procedure may reduce the separation performance (see for example [5]).

An additional non-parametric ICA method is based on kernel estimation of PDFs (see Ref. [2]). Substituting Eq. (4) into Eq. (43) yields the MI estimator

$$\mathscr{I}(\hat{\mathbf{s}}_1,\hat{\mathbf{s}}_2,\ldots,\hat{\mathbf{s}}_K)=-\sum_{k=1}^{K}\frac{1}{N}\sum_{l=1}^{N}\log\left\{\frac{1}{N}\sum_{n=1}^{N}\varphi(S_{l,n})\right\}-\log|\det(\mathbf{W})|, \tag{48}$$

where we ignore $\mathscr{H}_{\text{measurements}}$ and recalling that we already handled the normalization term in Eqs. (45) and (46). As in Eq. (1), denote f_k as the function creating the kth signal $\hat{\mathbf{s}}_k$ based on the measurements $[\mathbf{y}_1,\ldots,\mathbf{y}_K]$. Then,

$$\begin{aligned}\hat{s}_k(n)&=f_k(n;\mathbf{y};\mathbf{w}_k)\\&=[w_{k,1},\ldots,w_{k,K}][y_1(n),\ldots,y_K(n)]^{\mathrm{T}},\end{aligned} \tag{49}$$

where $\mathbf{w}_k$ is the kth row of $\mathbf{W}$. The gradient of Eq. (49) is

$$\mathbf{g}_k(n;\mathbf{y};\mathbf{w}_k)=[y_1(n),\ldots,y_K(n)]^{\mathrm{T}}\quad\forall k. \tag{50}$$

Differentiating Eq. (48) and substituting Eq. (50) yields (see Refs. [2,19])

$$\nabla_{\mathbf{w}}\mathscr{I}(\hat{\mathbf{s}}_1,\hat{\mathbf{s}}_2,\ldots,\hat{\mathbf{s}}_K)=\sum_{k=1}^{K}\left[\frac{1}{N\sigma^2}\sum_{l=1}^{N}\frac{\sum_{n=1}^{N}\varphi(S_{l,n})S_{l,n}\mathbf{Y}_{k,l,n}}{\sum_{n=1}^{N}\varphi(S_{l,n})}\right]-(\mathbf{W}^{-1})^{\mathrm{T}}, \tag{51}$$

where $\mathbf{Y}_{k,l,n}\equiv\Upsilon_{k,l}-\Upsilon_{k,n}$.

For any single source $k\in[1,\ldots,K]$, $f_k(n;\mathbf{y};\mathbf{w}_k)$ involves K multiplications, i.e., $N_f=K$. We need to calculate the K sources and the DE for each one of them. Accounting, in addition, for the $\mathscr{O}(K^3)$ calculations needed for $\log|\det(\mathbf{W})|$, the

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

complexity of Eq. (48) is

$$\mathcal{C}_{\mathrm{MI}}^{\mathrm{explicit}} = \mathcal{O}(K^3) + K\mathcal{C}_{\mathrm{entropy}}^{\mathrm{explicit}} = \mathcal{O}(K^3 + KN^2 + K^2N). \quad (52)$$

As for the complexity of the MI gradient, note that $\mathbf{g}_k(n;\mathbf{y};\mathbf{w}_k)$ is equivalent to the non-zero elements of matrix $\Upsilon_{k,n}$, which is used in Eq. (46). For this reason, the matrix $\mathbf{Y}_{k,l,n}$ in Eq. (51), is equivalent to $\mathbf{G}(l,n,\mathbf{y},\mathbf{w}_k)$ used in Eq. (7). In this case $N_{\mathrm{dim}} = K$ while $N_g = 0$. This applies to each of the K sources. Accounting for all sources, the complexity of Eq. (51) is

$$\mathcal{C}_{\mathrm{MIgrad}}^{\mathrm{explicit}} = \mathcal{O}(K^3) + K\mathcal{C}_{\mathrm{gradient}}^{\mathrm{explicit}} = \mathcal{O}(K^2N^2 + K^3), \quad (53)$$

where we include the $\mathcal{O}(K^3)$ calculations needed for inverting $\mathbf{W}$. The complexity given in Eq. (53) is very high.

A prior algorithm based on non-parametric kernel estimation [23] 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. On the other hand, by applying back propagation (Section 3.2) we achieve a complexity of $\mathcal{C}_{\mathrm{MIgrad}}^{\mathrm{backpropagation}} = \mathcal{O}(KN^2 + K^2N + K^3)$ for calculating both the MI and its gradient. Moreover, by applying entropy approximation by discrete convolution (Sections 4 and 5) we reduce the complexity to $\mathcal{C}_{\mathrm{MI}}^{\mathrm{approx}} = \mathcal{C}_{\mathrm{MIgrad}}^{\mathrm{approx}} = \mathcal{O}(KN\log N + K^2N + K^3)$. This allows fast ICA, while exploiting the advantages of non-parametric methods in high-dimensional problems.

6.3. Demonstrations

We performed several separation demonstrations. A set of simulations dealt with random sources. We simulated $K = 6$ 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 $N = 3000$. The sources were mixed using randomly generated full rank matrices (condition number $\leqslant 20$).

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	18 ± 4	760 min
Non-parametric ICA with fast kernel convolution	22 ± 3	1.2 min
Jade	7 ± 4	0.2 s
InfoMax	1 ± 0.5	1.4 s
InfoMax with pre-filtering	8 ± 4	1.6 s
Fast ICA	4 ± 4	1.1 s
Fast ICA with pre-filtering	5 ± 3	1.9 s

Source separation was attempted using three parametric ICA algorithms (see Refs. [4,17,21]): InfoMax, Jade and Fast ICA.[6] The software for those algorithms was downloaded from the webpages of the respective authors. In addition, separation was attempted using two non-parametric[7] ICA algorithms: the first is based on Ref. [2]. We implemented the algorithm described in Ref. [2] with the exception of using the method described in Section 3, in order to accelerate the gradient calculation. We then implemented the algorithm[8] described in Sections 4 and 5.

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 source cross-talk:

$$\mathrm{SIR} = \min_k \left(\frac{\|\mathbf{s}_k\|^2}{\|\mathbf{s}_k - \hat{\mathbf{s}}_k\|^2} \right). \quad (54)$$

[6] 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 $[-1\ 0\ 1]/2$.

[7] Our separation procedure was based on the BFGS Quasi-Newton algorithm as implemented in the MATLAB optimization toolbox (function FMINUNC).

[8] We used a grid having $M = 1000$ bins. 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.

Fig. 8. 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 20 dB.

Note that Eq. (54) uses the signal k having the minimal ratio, i.e., having the worst separation quality.[9] 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 pictures were mixed using randomly generated full rank matrices (condition number $\leqslant 100$). The separation results are presented in Fig. 8.

7. Conclusions

We have presented two techniques for accelerating the estimation of entropy and 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. Moreover we presented an approach for gradient approximation, which is more numerically sound than the straightforward approach. The low computational cost of our algorithms makes non-parametric entropy estimation applicable to high-dimensional problems and large sample sizes. It will be interesting to explore extensions to this approach to other entropy estimators such as [13,14] and thus benefit from efficient gradient based optimization of high dimensional joint entropy estimators.[10]

Acknowledgements

This research has been supported in parts by the "Dvorah" Fund of the Technion and by the HASSIP Research Network Program HPRN-CT-2002-00285, sponsored by the European Commission. The research was carried out in 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.

[9] As explained in the appendix, the estimated $\hat{\mathbf{s}}_k$ is prone to permutation and scale ambiguities. Thus, Eq. (54) is applied to separation results that are compensated for these ambiguities.

[10] The Matlab code for our fast non-parametric ICA algorithm can be downloaded from http://www.ee.technion.ac.il/~yoav/NlogNoptim, http://iew3.technion.ac.il/~mcib/.

Appendix: Ambiguities in MI optimization

Optimization of MI possesses three ambiguities: Permutation, sign and scale.

Permutation ambiguity: Let s_1, s_2 be two signals. Then

$$\begin{aligned}\mathscr{I}_{s_1,s_2} &= \mathscr{H}_{s_1} + \mathscr{H}_{s_2} - \mathscr{H}_{s_1,s_2} \\ &= \mathscr{H}_{s_2} + \mathscr{H}_{s_1} - \mathscr{H}_{s_2,s_1} = \mathscr{I}_{s_2,s_1}.\end{aligned} \tag{A.1}$$

Therefore, the reconstructed signals appear in an arbitrary order. This ambiguity does not concern us in this work.

Scale and sign ambiguities: Let s_1, s_2 be two statistically independent signals. Their joint PDF is thus separable:

$$p_{s_1,s_2}(s_1, s_2) = p_{s_1}(s_1)p_{s_2}(s_2). \tag{A.2}$$

Therefore, the MI of s_1, s_2 is $\mathscr{I}_{s_1,s_2} = 0$ (see for example [7]). Denote $\bar{s}_1 = \rho_1 s_1$ and $\bar{s}_2 = \rho_2 s_2$. The joint PDF of $\bar{s}_1, \bar{s}_2$ is still separable and equals

$$p_{\bar{s}_1,\bar{s}_2}(\bar{s}_1, \bar{s}_2) = p_{\bar{s}_1}(\bar{s}_1)p_{\bar{s}_2}(\bar{s}_2). \tag{A.3}$$

Therefore, their MI is zero as well. Assume that the 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. Thus, if $\mathbf{R}$ is a diagonal matrix, then also $\mathbf{RW}$ is a solution to the optimization problem. Here the MI of $\rho_1\hat{\mathbf{s}}_1, \ldots, \rho_K\hat{\mathbf{s}}_K$ is zero, where $\rho_1, \ldots, \rho_K$ are the diagonal elements of $\mathbf{R}$. 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 scale ambiguity, for which $\rho = -1$.

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 estimated 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.

References

[1] A.J. Bell, T.J. Sejnowski, An information-maximization approach to blind separation and blind deconvolution, Neural Comput. 7 (6) (1995) 1129–1159.

[2] R. Boscolo, H. Pan, V. P. Roychowdhury, Non-parametric ICA, in: Proceedings of the ICA2001, pp. 13–18.

[3] V.M. Bove Jr., Entropy-based depth from focus, J. Opt. Soc. Amer. A 10 (1993) 561–566.

[4] J.-F. Cardoso, A. Souloumiac, Blind beamforming for non Gaussian signals, IEE Proc. F Radar Signal Process. 140 (6) (1993) 362–370.

[5] J.-F. Cardoso, Blind signal separation: statistical principles, Proc. IEEE 9 (10) (1998) 2009–2025.

[6] P. Comon, Independent component analysis: a new concept, Signal Process. 36 (1994) 287–314.

[7] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley, New York, 1991.

[8] J.I. de la Rosa, G. Fleury, On the kernel selection for minimum-entropy estimation, in: Proceedings of the IEEE Instrumentation and Measurement Technology, vol. 2, 2002, pp. 1205–1210.

[9] R.O. Duda, P.E. Hart, D.G. Stock, Pattern Classification, Wiley, New York, 2001.

[10] D. Erdogmus, J.C. Principe, An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems, IEEE Trans. Signal Process. 50 (7) (2002) 1780–1786.

[11] D. Erdogmus, J.C. Principe, K.E. Hild II, On-line entropy manipulation: Stochastic information gradient, IEEE Signal Process. Lett. 10 (8) (2003) 242–245.

[12] J. Fan, J. Marron, Fast implementation of nonparametric curve estimators, J. Comput. Graph. Statist. 3 (1994) 35–56.

[13] A. Gray, A. Moore, Nonparametric density estimation: toward computational tractability, SIAM Data Mining.

[14] A. Gray, A. Moore, Rapid evaluation of multiple density models, in: AI and Statistics, 2003.

[15] A. Griewank, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, SIAM, Philadelphia, 2000.

[16] A. Hyvärinen, The Fast-ICA MATLAB package http://www.cis.hut.fi/~aapo/.

[17] A. Hyvärinen, J. Karhunen, E. Oja, Independent Component Analysis, Wiley, New York, 2001.

[18] E. Learned-Miller, J. Fisher, ICA using spacings estimates of entropy, J. Mach. Learn. Res. 4 (2003) 1271–1295.

[19] Y. Lomnitz, A. Yeredor, A blind-ML scheme for blind source separation, in: Proceedings of the IEEE Workshop on Statistical Signal Processing, 2003.

[20] D.G. Luenberger, Linear and Non Linear Programming, Addison-Wesley, California, 1987.

[21] S. Makeig, A.J. Bell, T.-P. Jung, T.J. Sejnowski, Independent component analysis of electroencephalographic data, Adv. Neural Inform. Process. Systems 8 (1996) 145–151.

[22] B.A. Pearlmutter, Fast exact multiplication by the hessian, Neural Comput. 6 (1) (1994) 147–160.

[23] D.T. Pham, Fast algorithm for estimating mutual information, entropies and score functions, in: Proceedings of the ICA2003, pp. 17–22.
[24] D.T. Pham, P. Garrat, Blind separation of a mixture of independent sources through a quasi-maximum likelihood approach, IEEE Trans. Signal Process. 45 (7) (1997) 1712–1725.
[25] J.C. Principe, D. Erdogmus, From adaptive linear to information filtering, in: Proceedings of the IEEE Symposium on Adaptive Systems for Signal Processing, Communications and Control, 2000, pp. 99–104.
[26] L.B. Rall, Automatic Differentiation: Techniques and Applications, Springer, Berlin, 1981.
[27] I. Santamaria, D. Erdogmus, J.C. Principe, Entropy minimization for supervised digital communications channel equalization, IEEE Trans. Signal Process. 50 (5) (2002) 1184–1192.
[28] Y.Y. Schechner, N. Kiryati, R. Basri, Separation of transparent layers using focus, Int. J. Comput. Vision 89 (2000) 25–39.
[29] Y.Y. Schechner, J. Shamir, N. Kiryati, Polarization and statistical analysis of scenes containing a semi-reflector, J. Opt. Soc. Amer. A 17 (2000) 276–284.
[30] N.N. Schraudolph, Gradient-based manipulation of nonparametric entropy estimates, IEEE Trans. Neural Networks 15 (4) (2004) 828–837.
[31] W. Schwartzkopf, B.L. Evans, A.C. Bovik, Entropy estimation for segmentation of multi-spectral chromosome images, in: Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation, 2002, pp. 234–257.
[32] B.W. Silverman, Kernel density estimation using the fast Fourier transform, J. Roy. Statist. Soc. Ser. C: Appl. Statist. 31 (1) (1982) 93–99.
[33] B.W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman & Hall, New York, 1986.
[34] S. Shwartz, M. Zibulevsky, Y.Y. Schechner, ICA using kernel entropy estimation with NlogN complexity, in: Proceedings of the ICA2004.
[35] P. Thevenaz, M. Unser, Optimization of mutual information for multiresolution image registration, IEEE Trans. Image Process. 9 (12) (2000) 2083–2099.
[36] O. Vasicek, A test for normality based on sample entropy, J. Roy. Statist. Soc. Ser. B 38 (1) (1976) 54–59.
[37] P.A. Viola, Alignment by maximization of mutual information, Ph.D. Thesis, Massachusetts Institute of Technology—Artificial Intelligence Laboratory, 1995.
[38] M.P. Wand, Fast computation of multivariate kernel estimators, J. Comput. Graph. Statist. 3 (1994) 433–445.
[39] R.R. Wang, T. Huang, Jialin-Zhong, Generative and discriminative face modelling for detection, in: Proceedings of the IEEE International Conference Automatic Face Gesture Recognition, 2002, pp. 281–286.