ITE (Information Theoretical Estimators) Matlab/Octave Toolbox Release 0.12

Zoltán Szabó

October 27, 2012

Contents

List of Figures

List of Tables

List of Examples

List of Templates

1 Introduction

Since the pioneering work of Shannon [62], *entropy, mutual information, divergence* measures and their extensions have found a broad range of applications in many areas of machine learning. Entropies provide a natural notion to quantify the *uncertainty* of random variables, mutual information type indices measure the *dependence* among its arguments, divergences offer efficient tools to define the 'distance' of probability measures. Particularly, in the classical Shannon case, these three concepts form a gradually widening chain: entropy is equal to the self mutual information of a random variable, mutual information is identical to the divergence of the joint distribution and the product of the marginals [12]. Applications of Shannon entropy, -mutual information, -divergence and their generalizations cover, for example, (i) feature selection, (ii) clustering, (iii) independent component/subspace analysis, (iii) image registration, (iv) boosting, (v) optimal experiment design, (vi) causality detection, (vii) hypothesis testing, (viii) Bayesian active learning, (ix) structure learning in graphical models, (x) region-of-interest tracking, among many others. For an excellent review on the topic, the reader is referred to [6, 86, 84, 5, 47].

Independent component analysis (ICA) [31, 9, 10] a central problem of signal processing and its generalizations can be formulated as optimization problems of information theoretical objectives. One can think of ICA as a cocktail party problem: we have some speakers (sources) and some microphones (sensors), which measure the mixed signals emitted by the sources. The task is to estimate the original sources from the mixed observations only. Traditional ICA algorithms are one-dimensional in the sense that all sources are assumed to be *independent* real valued random variables. However, many important applications underpin the relevance of considering extensions of ICA, such as the independent subspace analysis (ISA) problem [8, 13]. In ISA, the independent sources can be multidimensional: we have a cocktail-party, where more than one *group* of musicians are playing at the party. Successful applications of ISA include (i) the processing of EEG-fMRI, ECG data and natural images, (ii) gene expression analysis, (iii) learning of face view-subspaces, (iv) motion segmentation, (v) single-channel source separation, (vi) texture classification, (vii) action recognition in movies.

One of the most relevant and fundamental hypotheses of the ICA research is the ISA separation principle [8]: the ISA task can be solved by ICA followed by clustering of the ICA elements. This principle (i) forms the basis of the state-of-the-art ISA algorithms, (ii) can be used to design algorithms that scale well and efficiently estimate the dimensions of the hidden sources, (iii) has been recently proved [74] and (iv) can be extended to different linear-, controlled-, post nonlinear-, complex valued-, partially observed systems, as well as to systems with nonparametric source dynamics. For a recent review on the topic, see [77].

Although there exist many exciting applications of information theoretical measures, to the best of our knowledge, available packages in this domain focus on (i) discrete variables, or (ii) quite specialized applications and information theoretical estimation methods. Our **goal** is to fill in this serious gap by coming up with a (i) highly modular, (ii) free and open source, (iii) multi-platform toolbox, the ITE (information theoretical estimators) package, which

1. is capable of estimating *many* different variants of entropy, mutual information and divergence measures:

   - Shannon-, Rényi entropy; generalized variance (GV), kernel canonical correlation analysis (KCCA), kernel generalized variance (KGV), Hilbert-Schmidt independence criterion (HSIC), Shannon-, L_2-, Rényi-, Tsallis mutual information, copula-based kernel dependency, multivariate version of Hoeffding's Φ, Schweizer-Wolff's σ and κ; complex variants of entropy and mutual information; L_2-, Rényi-, Tsallis divergence, maximum mean discrepancy (MMD), and J-distance based on
   - nonparametric methods[1]: k-nearest neighbors, generalized k-nearest neighbors, weighted k-nearest neighbors, minimum spanning trees, geodesic spanning forests, random projection, ensemble methods, kernel techniques.

2. offers a *simple and unified framework* to

   (a) easily construct new estimators from existing ones or from scratch, and
   (b) transparently use the obtained estimators in information theoretical optimization problems.

3. with a *prototype application* in ISA and its extensions including

   - 6 different ISA objectives,
   - 4 optimization methods: (i) handling known and unknown subspace dimensions as well, with (ii) further objective-specific accelerations,

---

[1]It is highly advantageous to apply nonparametric approaches to estimate information theoretical quantities. The bottleneck of the 'opposite' plug-in type methods, which estimate the underlying density and then plug it in into the appropriate integral formula, is that the unknown densities are nuisance parameters. As a result, plug-in type estimators scale poorly as the dimension is increasing.

- 5 extended problem directions: (i) different linear-, (ii) controlled-, (iii) post nonlinear-, (iii) complex valued-, (iv) partially observed models, (v) as well as systems with nonparametric source dynamics; which can be used in combinations as well.

The technical details of the ITE package are as follows:

- **Author**: Zoltán Szabó.
  - Homepage: `http://nipg.inf.elte.hu/szzoli`
  - Email: `szzoli@cs.elte.hu`
  - Affiliation: Eötvös Loránd University, Faculty of Informatics, Pázmány Péter sétány 1/C, Budapest, H-1117, Hungary.

- **Documentation of the source**: the source code of ITE has been enriched with numerous comments, examples, and pointers where the interested user can find further mathematical details about the embodied techniques.

- **License** (GNU GPLv3 or later): ITE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with ITE. If not, see <http://www.gnu.org/licenses/>.

- **Citing**: If you use the ITE toolbox in your work, please cite the papers [74, 77] (.bib in Appendix A).

- **Platforms**: The ITE package has been extensively tested on Windows and Linux. However, since it is made of standard Matlab/Octave and C++ files, it is expected to work on alternative platforms as well.

- **Environments**: Matlab[2], Octave[3].

- **Requirements**: The ITE package is self-contained, it only needs a Matlab or an Octave environment with standard toolboxes:
  - Matlab: Image Processing, Optimization, Statistics.
  - Octave[4]: Image Processing (image), Statistics (statistics), Input/Output (io, required by statistics), Ordinary Differential Equations (odepkg), Bindings to the GNU Scientific Library (gsl), ANN wrapper (ann).

- **Comments, feedbacks**: are welcome.

- **Homepage of the ITE toolbox**: `https://bitbucket.org/szzoli/ite/`

The remainder of this document is organized as follows. Section 2 is about the installation of the ITE package. Section 3 focuses on the estimation of information theoretical quantities (entropy, mutual information, divergence measures) and their realization in ITE. In Section 4, we present an application of Section 3 included in the ITE toolbox. The application considers the extension of independent subspace analysis (ISA, independent component analysis with multidimensional sources) to different linear-, controlled-, post nonlinear-, complex valued-, partially observed problems, as well as problems dealing with nonparametric source dynamics, i.e., the independent process analysis (IPA) problem family. Section 5 is about the organization of the directories of the ITE toolbox. Citing information of the ITE package is provided in Appendix A. Abbreviations of the paper are listed in Appendix B (Table 21). Functions with Octave-specific adaptations are summarized in Appendix C (Table 22).

2 Installation

This section is about (i) the installation of the ITE toolbox, and (ii) the external packages, dedicated solvers embedded in the ITE package. The purpose of this inclusion is twofold:

---

[2] `http://www.mathworks.com/products/matlab/`

[3] `http://www.gnu.org/software/octave/`

[4] See `http://octave.sourceforge.net/packages.php`.

- to further increase the efficiency of certain subtasks to be solved (e.g., k-nearest neighbor search, finding minimum spanning trees, some subtasks revived by the IPA separation principles (see Section 4.1)),

- to provide both purely Matlab/Octave implementations, and specialized (often faster) non-Matlab/-Octave solutions that can be called from Matlab/Octave.

The core of the ITE toolbox has been written in Matlab, as far it was possible in an Octave compatible way. The particularities of Octave has been taken into account by adapting the code to the *actual* environment (Matlab/Octave). The working environment can be queried (e.g., in case of extending the package it is also useful) by the `working_environment_Matlab.m` function included in ITE. Adaptations has been carried out in the functions listed in Appendix C (Table 22). The functionalities extended by the external packages are also available in both environments (Table 1).

Here, a short description of the embedded/downloaded packages (directory 'shared/embedded', 'shared/downloaded') is given:

1. **fastICA** (directory 'shared/embedded/FastICA'; version 2.5):
   - **URL**: `http://research.ics.tkk.fi/ica/fastica/`
   - **License**: GNU GPLv2 or later.
   - **Solver**: ICA (independent component analysis).
   - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
   - **Environment**: Matlab, Octave.
   - **Note**: By commenting out the `g_FastICA_interrupt` variable in `fpica.m`, the `fastica.m` function can be used in Octave, too. The provided fastICA code in the ITE toolbox contains this modification.

2. **Complex fastICA** (directory 'shared/embedded/CFastICA')
   - **URL**: `http://www.cs.helsinki.fi/u/ebingham/software.html`, `http://users.ics.aalto.fi/ella/publications/cfastica_public.m`
   - **License**: GNU GPLv2 or later.
   - **Solver**: complex ICA.
   - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
   - **Environment**: Matlab, Octave.

3. **ANN (approximate nearest neighbor) Matlab wrapper** (directory 'shared/embedded/ann_wrapperM'; version 'Mar2012'):
   - **URL**: `http://www.wisdom.weizmann.ac.il/~bagon/matlab.html`, `http://www.wisdom.weizmann.ac.il/~bagon/matlab_code/ann_wrapper_Mar2012.tar.gz`
   - **License**: GNU LGPLv3.
   - **Solver**: approximate nearest neighbor computation.
   - **Installation**: Follow the instructions in the ANN wrapper package (README.txt: INSTALLATION) till 'ann_class_compile'. Note: If you use a more recent C++ compiler (e.g., g++ on Linux), you have to include the following 2 lines into the original code to be able to compile the source:
     (a) '#include <cstdlib>' to 'ANNx.h'
     (b) '#include <cstring>' to 'kd_tree.h'
     The provided ANN code in the ITE package contains these modifications.
   - **Environment**: Matlab, Octave[5].
   - **Note**: fast nearest neighbor alternative of `knnsearch` ∈ Matlab: Statistics Toolbox.

4. **MatlabBGL** (directory 'shared/embedded/MatlabBGL', version 4.0)
   - **URL**: `https://github.com/dgleich/matlab-bgl`, `http://www.mathworks.com/matlabcentral/fileexchange/10922`
   - **License**: 2-clause BSD, and GNU GPLv2 or later.
   - **Solver**: minimum spanning trees: Prim and Kruskal algorithm.
   - **Installation**: Add it with subfolders to your Matlab/Octave PATH. Note:

---

[5] At the time of writing this paper, the Octave ANN wrapper (`http://octave.sourceforge.net/ann/index.html`, version 1.0.2) supports $2.9.12 \leq$ Octave $< 3.4.0$. According to our experiences, however the ann wrapper can also be used for higher versions of Octave provided that (i) a new swig package (`www.swig.org/`) is used ($>$=2.0.5), (ii) a new 'SWIG=swig' line is inserted in src/ann/bindings/Makefile (the ITE package contains the modified makefile), and (iii) the row containing 'typedef OCTAVE_IDX_TYPE octave_idx_type;' (in '.../octave/config.h') is commented out for the time of 'make'-ing.

– The package includes precompiled MEX files for Windows (32-bit and 64-bit), and Linux (32-bit and 64-bit for Matlab 2006b+), and MacOSX (32-bit Intel and 32-bit PPC).
– The package includes source code to compile on other platforms as well.
- **Environment**: Matlab, Octave[6].
- **Note**: alternative of '14) = pmtk3' in finding minimum spanning trees.

5. **FastKICA** (directory 'shared/embedded/FastKICA', version 1.0):
   - **URL**: `http://people.kyb.tuebingen.mpg.de/arthur/fastkica.htm`
   - **License**: GNU GPL v2 or later.
   - **Solver**: HSIC (Hilbert-Schmidt independence criterion) mutual information estimator.
   - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
   - **Environment**: Matlab, Octave.
   - **Note**: one can extend the implementation of HSIC to measure the dependence of d_m-dimensional variables, too. The ITE toolbox contains this modification.

6. **NCut** (Normalized Cut, directory 'shared/embedded/NCut'; version 9):
   - **URL**: `http://www.seas.upenn.edu/~timothee/software/ncut/ncut.html`, `http://www.seas.upenn.edu/~timothee/software/ncut/Ncut_9.zip`
   - **License**: GNU GPLv3.
   - **Solver**: spectral clustering, fixed number of groups.
   - **Installation**: Run `compileDir_simple.m` from Matlab to the provided directory of functions.
   - **Environment**: Matlab.
   - **Note**: the package is a fast alternative of '11) = spectral clustering'.

7. **sqdistance** (directory 'shared/embedded/sqdistance')
   - **URL**: `http://www.mathworks.com/matlabcentral/fileexchange/24599-pairwise-distance-matrix/`, `http://www.mathworks.com/matlabcentral/fileexchange/24599-pairwise-distance-matrix?download=true`
   - **License**: 2-clause BSD.
   - **Solver**: fast pairwise distance computation.
   - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
   - **Environment**: Matlab, Octave.
   - **Note**: compares favourably to the Matlab/Octave function `pdist`.

8. **TCA** (directory 'shared/embedded/TCA'; version 1.0):
   - **URL**: `http://www.di.ens.fr/~fbach/tca/index.htm`, `http://www.di.ens.fr/~fbach/tca/tca1_0.tar.gz`
   - **License**: GNU GPLv2 or later.
   - **Solver**: KCCA (kernel canonical correlation analysis) / KGV (kernel generalized variance) estimator, incomplete Cholesky decomposition.
   - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
   - **Environment**: Matlab, Octave.
   - **Note**: Incomplete Cholesky factorization can be carried out by the Matlab/Octave function `chol_gauss.m`. One can also compile the included `chol_gauss.c` to attain improved performance. Functions provided in the ITE toolbox contain extensions of the KCCA and KGV indices to measure the dependence of d_m-dimensional variables. The computations have also been accelerated in ITE by '7) = sqdistance'.

9. **Weighted kNN** (kNN: k-nearest neighbor; directory 'shared/embedded/weightedkNN' and the core of `HRenyi_weightedkNN_estimation.m`):
   - **URL**: `http://www-personal.umich.edu/~kksreddy/`
   - **License**: GNU GPLv3 or later.
   - **Solver**: Rényi entropy estimator based on the weighted k-nearest neighbor method.
   - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
   - **Environment**: Matlab, Octave.

---

[6]With some trick, the MatlabBGL works on Octave, see `https://answers.launchpad.net/matlab-bgl/+question/48686`.

- **Note**: in the weighted kNN technique the weights are optimized. Since Matlab and Octave rely on different optimization engines, one has to adapt the weight estimation procedure to Octave. The `calculateweight.m` function in ITE contains this modification.

10. **E4** (directory 'shared/embedded/E4'):
    - **URL**: `http://www.ucm.es/info/icae/e4`, `http://www.ucm.es/info/icae/e4/downfiles/E4.zip`
    - **License**: GNU GPLv2 or later.
    - **Solver**: AR (autoregressive) fit.
    - **Installation**: Add it with subfolders to your Matlab/Octave PATH[7].
    - **Environment**: Matlab, Octave.
    - **Note**: alternative of '13) = ARfit' in AR identification.

11. **spectral clustering** (directory 'shared/embedded/sp_clustering'):
    - **URL**: `http://www.mathworks.com/matlabcentral/fileexchange/34412-fast-and-efficient-spectral-clustering`
    - **License**: 2-clause BSD.
    - **Solver**: spectral clustering.
    - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
    - **Environment**: Matlab, Octave.
    - **Note**: the package is a purely Matlab/Octave alternative of '6)=NCut'. It is advisable to alter the eigensystem computation in the `SpectralClustering.m` function to work stably in Octave; the modification is included in the ITE toolbox and is activated in case of Octave environment.

12. **clinep** (directory 'shared/embedded/clinep'):
    - **URL**: `http://www.mathworks.com/matlabcentral/fileexchange/8597-plot-3d-color-line/content/clinep.m`
    - **License**: 2-clause BSD.
    - **Solver**: Plots a 3D line with color encoding along the length using the patch function.
    - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
    - **Environment**: Matlab, Octave.
    - **Note**: (i) calling of the cylinder function (in `clinep.m`) has to modified somewhat to work in Octave, and (ii) since 'gnuplot (as of v4.2) only supports 3D filled triangular patches' one has to use the fltk graphics toolkit in Octave for drawing. The included `cline.m` code in the ITE package contains these modifications.

13. **ARfit** (directory 'shared/downloaded/ARfit', version 'March 20, 2011')
    - **URL**: `http://www.gps.caltech.edu/~tapio/arfit/`, `http://www.gps.caltech.edu/~tapio/arfit/arfit.zip`.
    - **License**: ACM.
    - **Solver**: AR identification.
    - **Installation**: Download, extract and add it with subfolders to your Matlab/Octave PATH.
    - **Environment**: Matlab, Octave.
    - **Note**: alternative of '10) = E4' in AR identification.

14. **pmtk3** (directory 'shared/embedded/pmtk3', version 'Jan 2012')
    - **URL**: `http://code.google.com/p/pmtk3`, `http://code.google.com/p/pmtk3/downloads/detail?name=pmtk3-3jan11.zip&can=2&q=`.
    - **License**: MIT.
    - **Solver**: minimum spanning trees: Prim algorithm.
    - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
    - **Environment**: Matlab, Octave.
    - **Note**: purely Matlab/Octave alternative of '4) = MatlabBGL' in finding minimum spanning trees.

15. **knn** (directory 'shared/embedded/knn', version 'Nov 02, 2010')

---

[7]In Octave, this step results in a 'warning: function `.../shared/embedded/E4/vech.m` shadows a core library function'; it is OK, the two functions compute the same quantity.

- **URL**: http://www.mathworks.com/matlabcentral/fileexchange/28897-k-nearest-neighbor-search, http://www.mathworks.com/matlabcentral/fileexchange/28897-k-nearest-neighbor-search?download=true
- **License**: 2-clause BSD.
- **Solver**: kNN search.
- **Installation**: Run the included `build` command to compile the partial sorting function `top.cpp`. Add it with subfolders to your Matlab/Octave PATH.
- **Environment**: Matlab, Octave.
- **Note**: Alternative of '3)=ANN' in finding k-nearest neighbors.

16. **SWICA** (directory 'shared/embedded/SWICA')
    - **URL**: http://www.stat.purdue.edu/~skirshne/SWICA, http://www.stat.purdue.edu/~skirshne/SWICA/swica.tar.gz
    - **License**: 3-clause BSD.
    - **Solver**: Schweizer-Wolff's σ and κ estimation.
    - **Installation**: Add it with subfolders to your Matlab/Octave PATH.
    - **Environment**: Matlab, Octave.
    - **Note**: one can also compile the included `SW_kappa.cpp` and `SW_sigma.cpp` functions to further accelerate computations (see 'build_SWICA.m').

A short summary of the packages can be found in Table 1. To ease installation, the ITE package contains an installation script, `ITE_install.m`. A typical usage is to cd to the directory 'code' and call `ITE_install(pwd)`. Running the script from Matlab/Octave, it (i) adds the main ITE directory with subfolders to the Matlab/Octave PATH, (ii) downloads and extracts the ARfit package, and (iii) compiles the embedded ANN, NCut, TCA, SWICA, knn packages and a .cpp acceleration of the Hoeffding's Φ [see Eq. (16)] computation[8]. The `ITE_install.m` script automatically detects the working environment (Matlab/Octave) and performs the installation accordingly, for example, it deletes the ann wrapper not suitable for the current working environment. The output of a successful installation in Matlab is given below (the Octave output is similar):

Example 1 (ITE installation (output; with compilation))

```
>> ITE_install(pwd); %after cd-ing to the code directory
Installation: started.
We are working in Matlab environment. => ann_wrapper for Octave: deleted.
ARfit package: downloading, extraction: started.
ARfit package: downloading, extraction: ready.
ITE directory: added with subfolders to the Matlab PATH.
ANN compilation: started.
ANN compilation: ready.
NCut compilation: started.
NCut compilation: ready.
TCA (chol_gauss.c) compilation: started.
TCA (chol_gauss.c) compilation: ready.
SWICA (SW_kappa.cpp, SW_sigma.cpp) compilation: started.
SWICA (SW_kappa.cpp, SW_sigma.cpp) compilation: ready.
Hoeffding_term1.cpp compilation: started.
Hoeffding_term1.cpp compilation: ready.
knn (top.cpp) compilation: started.
knn (top.cpp) compilation: ready.
-------------------
Installation tests:
ANN quick test: successful.
NCut quick test: successful.
ARfit quick test: successful.
knn quick test: successful.
```

---

[8]The ITE package also offers a purely Matlab/Octave implementation for the computation of Hoeffding's Φ. Without compilation this Matlab/Octave implementation is evoked.

Task	Package	Written in	Environment	Directory
ICA	fastICA	Matlab	Matlab, Octave	shared/embedded/FastICA
complex ICA	complex fastICA	Matlab	Matlab, Octave	shared/embedded/CFastICA
kNN search	ANN	C++	Matlab	shared/embedded/ann_wrapperM^a
kNN search	ANN	C++	Octave^b	shared/embedded/ann_wrapperO^a
Prim-, Kruskal algorithm	MatlabBGL	C++	Matlab, Octave^c	shared/embedded/MatlabBGL
HSIC estimation	FastKICA	Matlab	Matlab, Octave	shared/embedded/FastKICA
spectral clustering	NCut	C++	Matlab	shared/embedded/NCut
fast pairwise distance computation	sqdistance	Matlab	Matlab, Octave	shared/embedded/sqdistance
KCCA, KGV	TCA	Matlab, C	Matlab, Octave	shared/embedded/TCA
Rényi entropy via weighted kNNs	weighted kNN	Matlab	Matlab, Octave	shared/embedded/weightedkNN
AR fit	E4	Matlab	Matlab, Octave	shared/embedded/E4
spectral clustering	spectral clustering	Matlab	Matlab, Octave	shared/embedded/sp_clustering
trajectory plot	clinep	Matlab	Matlab, Octave	shared/embedded/clinep
AR fit	ARfit	Matlab	Matlab, Octave	shared/downloaded/ARfit
Prim algorithm	pmtk3	Matlab	Matlab, Octave	shared/embedded/pmtk3
kNN search	knn	Matlab, C++	Matlab, Octave	shared/embedded/knn
Schweizer-Wolff's σ and κ	SWICA	Matlab, C++	Matlab, Octave	shared/embedded/SWICA

Table 1: External, dedicated packages increasing the efficiency of ITE.

^{a}In 'ann_wrapperM' 'M' stands for Matlab, in 'ann_wrapperO' 'O' denotes Octave.
^{b}See footnote 5.
^{c}See footnote 6.

3 Estimation of Information Theoretical Quantities

In this section we focus on the estimation of information theoretical quantities. Particularly, in the sequel, the underlying idea how the estimators are implemented in ITE are detailed, accompanied with definitions, numerous examples and extension possibilities/instructions.

The ITE package supports the estimation of many different variants of entropy, mutual information and divergence measures:

1. From construction point of view, we distinguish two types of estimators in ITE: *base* (Section 3.1) and *meta* (Section 3.2) ones. Meta estimators are *derived* from existing base/meta ones by taking into account information theoretical identities. For example, by considering the well-known

$$I\left(\mathbf{y}^1,\ldots,\mathbf{y}^M\right) = \sum_{m=1}^{M} H\left(\mathbf{y}^m\right) - H\left(\left[\mathbf{y}^1;\ldots;\mathbf{y}^M\right]\right) \tag{1}$$

   relation [12], one can estimate mutual information (I) by making use of existing entropy estimators (H).

2. From calling point of view, base and meta estimations follow exactly the same syntax (Section 3.3).

This modular implementation of the ITE package, makes it possible to

1. construct new estimators from existing ones, and

2. transparently use *any* of these estimators in information theoretical optimization problems (see Section 4) – provided that they follow a simple template described in Section 3.3.

3.1 Base Estimators

This section is about the *base* information theoretical estimators of the ITE package. Entropy estimation is in the focus of Section 3.1.1; in Section 3.1.2 and Section 3.1.3 we consider mutual information and divergence estimation, respectively.

3.1.1 Entropy Estimators

Let us start with a simple example: our goal is to estimate the Shannon entropy [62]

$$H(\mathbf{y}) = -\int_{\mathbb{R}^d} f(\mathbf{y}) \log f(\mathbf{y}) \mathrm{d}\mathbf{y} \tag{2}$$

of a random variable $\mathbf{y} \in \mathbb{R}^d$ from which we have i.i.d. (independent identically distributed) samples $\{\mathbf{y}_t\}_{t=1}^T$, and f denotes the density function of $\mathbf{y}$. The estimation of Shannon entropy can be carried out, e.g., by k-nearest neighbor techniques. Let us also assume that multiplicative contants are also important for us – in many applications, it is completely irrelevant whether we estimate, for example, $H(\mathbf{y})$ or $cH(\mathbf{y})$, where $c = c(d)$ is a constant depending only on the *dimension* of $\mathbf{y}$ (d), but *not on the distribution* of $\mathbf{y}$. By using the ITE package, the estimation can be carried out as simply as follows:

Example 2 (Entropy estimation (base-1: usage))

```
>Y = rand(5,1000);                       %generate the data of interest (d=5, T=1000)
>mult = 1;                               %multiplicative constant is important
>co = HShannon_kNN_k_initialization(mult); %initialize the entropy ('H') estimator
                                         %('Shannon_kNN_k'), including the value of k
>H = HShannon_kNN_k_estimation(Y,co);    %perform entropy estimation
```

An alternative entropy measure of interest is the Rényi entropy [59] defined as

$$H_\alpha(\mathbf{y}) = \frac{1}{1-\alpha} \log \int_{\mathbb{R}^d} f^\alpha(\mathbf{y}) \mathrm{d}\mathbf{y}, \tag{3}$$

where the random variable $\mathbf{y} \in \mathbb{R}^d$ have density function f. In fact, the Shannon entropy [Eq. (2)] is a special case of the Rényi entropy family by the

$$\lim_{\alpha \to 1} H_\alpha = H \tag{4}$$

limit relation. In the ITE toolbox, H_α can be estimated similarly to the Shannon entropy H (see Example 2):

Example 3 (Entropy estimation (base-2: usage))

```
>Y = rand(5,1000);                       %generate the data of interest (d=5, T=1000)
>mult = 1;                               %multiplicative constant is important
>co = HRenyi_kNN_k_initialization(mult); %initialize the entropy ('H') estimator ('Renyi_kNN_k'),
                                         %including the value of k and α
>H = HRenyi_kNN_k_estimation(Y,co);      %perform entropy estimation
```

Beyond k-nearest neighbor based H (see [35] ($S = \{1\}$), [64, 19] $S = \{k\}$) and H_α estimation methods [88, 39] ($S = \{k\}$), the ITE package also provide functions for the estimation of $H_\alpha(\mathbf{y})$ ($\mathbf{y} \in \mathbb{R}^d$) using (i) k-nearest neighbors ($S = \{1, \ldots, k\}$) [53], (ii) generalized nearest neighbor graphs ($S \subseteq \{1, \ldots, k\}$) [51], (iii) weighted k-nearest neighbors [65], (iv) minimum spanning trees [88, 52], and (v) geodesic spanning forests [52]. The base entropy estimators are summarized in Table 2; the calling syntax of these methods is the same as in Example 2 and Example 3, one only has to change 'Shannon_kNN_k' (see Example 2) and 'Renyi_kNN_k' (see Example 3) to the `cost_name` given in the last column of the table.

3.1.2 Mutual Information Estimators

In our next example, we consider the estimation of the mutual information of the d_m-dimensional components of the random variable $\mathbf{y} = [\mathbf{y}^1, \ldots, \mathbf{y}^M] \in \mathbb{R}^D$ ($D = \sum_{m=1}^M d_m$):

$$I\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right) = \int_{\mathbb{R}^{d_1}} \cdots \int_{\mathbb{R}^{d_M}} f\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right) \log \left[\frac{f\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right)}{\prod_{m=1}^M f_m(\mathbf{y}^m)}\right] \mathrm{d}\mathbf{y}^1 \cdots \mathrm{d}\mathbf{y}^M \tag{5}$$

using an i.i.d. sample set $\{\mathbf{y}_t\}_{t=1}^T$ from $\mathbf{y}$, where f is the joint density function of $\mathbf{y}$ and f_m is its mth marginal density, the density function of $\mathbf{y}^m$. As it is known, $I\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right)$ is non-negative and is zero, if and only if the $\{\mathbf{y}^m\}_{m=1}^M$ variables are jointly independent [12]. Mutual information can be efficiently estimated, e.g., on the basis of entropy [Eq. (1)] or Kullback-Leibler divergence; we will return to these *derived* approaches while presenting *meta* estimators in Section 3.2.

There also exist other mutual information-like quantities measuring the independence of $\mathbf{y}^m$s:

Estimated quantity	Principle	Name (`cost_name`)
Shannon entropy (H)	k-nearest neighbors ($S = \{k\}$)	'Shannon_kNN_k'
Rényi entropy (H_α)	k-nearest neighbors ($S = \{k\}$)	'Renyi_kNN_k'
Rényi entropy (H_α)	k-nearest neighbors ($S = \{1, \ldots, k\}$)	'Renyi_kNN_1tok'
Rényi entropy (H_α)	generalized nearest neighbor graphs ($S \subseteq \{1, \ldots, k\}$)	'Renyi_kNN_S'
Rényi entropy (H_α)	weighted k-nearest neighbors	'Renyi_weightedkNN'
Rényi entropy (H_α)	minimum spanning trees	'Renyi_MST'
Rényi entropy (H_α)	geodesic spanning forests	'Renyi_GSF'

Table 2: Entropy estimators (base).

1. The kernel canoncial correlation analysis (KCCA) is defined as

$$I_{\text{KCCA}}(\mathbf{y}^1, \mathbf{y}^2) = \sup_{f_1 \in \mathcal{F}^1, f_2 \in \mathcal{F}^2} \frac{\text{cov}[f_1(\mathbf{y}^1), f_2(\mathbf{y}^2)]}{\sqrt{\text{var}\,[f_1(\mathbf{y}^1)] + \kappa \, \|f_1\|_{\mathcal{F}^1}^2} \sqrt{\text{var}\,[f_2(\mathbf{y}^2)] + \kappa \, \|f_2\|_{\mathcal{F}^2}^2}}, \quad (\kappa > 0) \tag{6}$$

for $M = 2$ components, where 'cov' denotes covariance and 'var' stands for variance. In words, I_{KCCA} is the regularized form of the supremum correlation of $\mathbf{y}^1 \in \mathbb{R}^{d_1}$ and $\mathbf{y}^2 \in \mathbb{R}^{d_2}$ over two 'rich enough' reproducing kernel Hilbert spaces (RKHSs), $\mathcal{F}^1$ and $\mathcal{F}^2$. The computation of I_{KCCA} can be reduced to a generalized eigenvalue problem and the measure can be extended to $M \geq 2$ components to measure pairwise independence [4, 74].

2. Let $\mathbf{y} = \left[\mathbf{y}^1; \ldots, \mathbf{y}^M\right]$ be a multidimensional Gaussian random variable with covariance matrix $\mathbf{C}$ and let $\mathbf{C}^{i,j} \in \mathbb{R}^{d_i \times d_j}$ denote the cross-covariance between components of $\mathbf{y}^m \in \mathbb{R}^{d_m}$. In the Gaussian case, the mutual information between components $\mathbf{y}^1, \ldots, \mathbf{y}^M$ is [12]:

$$I\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right) = -\frac{1}{2} \log\left(\frac{\det \mathbf{C}}{\prod_{m=1}^M \det \mathbf{C}^{m,m}}\right). \tag{7}$$

If $\mathbf{y}$ is *not normal* then one can transform $\mathbf{y}^m$s using feature mapping $\boldsymbol{\varphi}$ associated with an RKHS and apply Gaussian approximation to obtain

$$I_{\text{KGV}}\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right) = -\frac{1}{2} \log\left[\frac{\det(\boldsymbol{\mathcal{K}})}{\prod_{m=1}^M \det(\boldsymbol{\mathcal{K}}^{m,m})}\right], \tag{8}$$

where $\boldsymbol{\phi}(\mathbf{y}) := [\boldsymbol{\varphi}(\mathbf{y}^1); \ldots; \boldsymbol{\varphi}(\mathbf{y}^M)]$, $\boldsymbol{\mathcal{K}} := cov[\boldsymbol{\phi}(\mathbf{y})]$, and the sub-matrices are $\boldsymbol{\mathcal{K}}^{i,j} = cov[\boldsymbol{\varphi}(\mathbf{y}^i), \boldsymbol{\varphi}(\mathbf{y}^j)]$. For further details on the kernel generalized variance (KGV) method, see [4, 74].

3. Let us given two RKHSs $\mathcal{F}^1$ and $\mathcal{F}^2$ with associated feature maps $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$. Let the corresponding cross-covariance operator be

$$\mathbf{C}_{\mathbf{y}_1, \mathbf{y}_2} = E\left([\boldsymbol{\varphi}_1(\mathbf{y}_1) - \boldsymbol{\mu}_1] \otimes [\boldsymbol{\varphi}_2(\mathbf{y}_2) - \boldsymbol{\mu}_2]\right), \tag{9}$$

where $\otimes$ denotes tensor product, E is the expectation and the mean embeddings are

$$\boldsymbol{\mu}_m = E[\boldsymbol{\varphi}_m(\mathbf{y}^m)] \quad (m = 1, 2). \tag{10}$$

The Hilbert-Schmidt independence criterion (HSIC) [21] is defined as the Hilbert-Schmidt norm of the cross-covariance operator

$$I_{\text{HSIC}}\left(\mathbf{y}^1, \mathbf{y}^2\right) = \|\mathbf{C}_{\mathbf{y}_1, \mathbf{y}_2}\|_{\text{HS}}^2. \tag{11}$$

The HSIC measure can also be extended to the $M \geq 2$ case to measure pairwise independence.

4. The generalized variance (GV) measure [79] considers the decorrelation of two one-dimensional random variables $y^1 \in \mathbb{R}$ and $y^2 \in \mathbb{R}$ ($M = 2$) over a finite function set $\mathcal{F}$:

$$I_{\text{GV}}\left(y^1, y^2\right) = \sum_{f \in \mathcal{F}} \left(corr\left[f\left(y^1\right), f\left(y^2\right)\right]\right)^2. \tag{12}$$

5. Let C be the copula of the random variable $\mathbf{y} = \left[y^1; \ldots; y^M\right] \in \mathbb{R}^M$. One may think of C as the distribution function on $[0,1]^M$, which links the joint distribution function (F) and the marginals $(F_m, i = 1, \ldots, M)$:

$$F(\mathbf{y}) = C\left(F_1\left(y^1\right), \ldots, F_M\left(y^M\right)\right). \tag{13}$$

It can be shown that the $y^i \in \mathbb{R}$ variables are independent if and only if C, the copula of $\mathbf{y}$ equals to the product copula Π defined as

$$\Pi(u_1, \ldots, u_M) = \prod_{m=1}^{M} u_m. \tag{14}$$

Using this result, the independence of y^is can be measured by the (normalized) L^p distance of C and Π:

$$\left(h_p(d) \int_{[0,1]^d} |C(\mathbf{u}) - \Pi(\mathbf{u})|^p \, d\mathbf{u}\right)^{\frac{1}{p}}, \tag{15}$$

where (i) $1 \leq p \leq \infty$, (ii) by an appropriate choice of the normalization constant $h_p(d)$, the value of (15) belongs to $\in [0,1]$ for any C.

- For $p = 2$, the special

$$I_{\text{Hoeffding}}\left(y^1, \ldots, y^d\right) = \left(h_2(d) \int_{[0,1]^d} [C(\mathbf{u}) - \Pi(\mathbf{u})]^2 d\mathbf{u}\right)^{\frac{1}{2}} \tag{16}$$

  quantity
  - is a generalization of Hoeffding's Φ defined for $d = 2$ [24],
  - can be analytically computated [18].
- For $p = 1$ and $p = \infty$, we obtain the Schweizer-Wolff's σ and κ [61], respectively. In this case no explicit expressions for the integrals are available. For small dimensional problems, however, the quantities can be efficiently estimated numerically. ITE contains methods for the $M = 2$ case:

$$I_{\text{SW1}}\left(y^1, y^2\right) = \sigma = 12 \int_{[0,1]^2} |C(\mathbf{u}) - \Pi(\mathbf{u})| d\mathbf{u}, \tag{17}$$

$$I_{\text{SWinf}}\left(y^1, y^2\right) = \kappa = 4 \sup_{\mathbf{u} \in [0,1]^2} |C(\mathbf{u}) - \Pi(\mathbf{u})|, \tag{18}$$

  using the SWICA package [34].

For an excellent introduction on copulas, see [43].

The estimation of these quantities can be carried out easily in the ITE package. Let us take the KCCA measure as an example:

Example 4 (Mutual information estimation (base: usage))

```
>ds = [2;3;4]; Y=rand(sum(ds),5000); %generate the data of interest (ds(m)=dim(y^m), T=5000)
>mult = 1;                            %multiplicative constant is important
>co = IKCCA_initialization(mult);     %initialize the mutual information ('I') estimator ('KCCA')
>I = IKCCA_estimation(Y,ds,co);       %perform mutual information estimation
```

The calling syntax of the mutual information estimators, are completely the same; one only has to change 'KCCA' to the `cost_name` given in the last column of the Table 3. The table summarizes the base mutual information estimators in ITE.

Estimated quantity	Principle	Name (`cost_name`)
generalized variance (I_{GV})	f-covariance/-correlation ($f \in \mathcal{F}$, $|\mathcal{F}| < \infty$)	`'GV'`
Hilbert-Schmidt (HS) independence criterion (I_{HSIC})	HS norm of the cross-covariance operator	`'HSIC'`
kernel canonical correlation (I_{KCCA})	sup correlation over RKHSs	`'KCCA'`
kernel generalized variance (I_{KGV})	Gaussian mutual information of the features	`'KGV'`
multivariate version of Hoeffding's Φ ($I_{\mathrm{Hoeffding}}$)	L^2 distance of the copula and the product copula	`'Hoeffding'`
Schweizer-Wolff's σ (I_{Sw1})	L^1 distance of the copula and the product copula	`'SW1'`
Schweizer-Wolff's κ (I_{SWinf})	L^∞ distance of the copula and the product copula	`'SWinf'`

Table 3: Mutual information estimators (base).

3.1.3 Divergence Estimators

Divergences measure the 'distance' between two probability densities, $f_1 : \mathbb{R}^d \mapsto \mathbb{R}$ and $f_2 : \mathbb{R}^d \mapsto \mathbb{R}$. One of the most well-known such index is the Kullback-Leibler divergence [36]:

$$D(f_1, f_2) = \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} f_1\left(\mathbf{y}^1\right) \log \left[\frac{f_1\left(\mathbf{y}^1\right)}{f_2\left(\mathbf{y}^2\right)} \right] \mathrm{d}\mathbf{y}^1 \mathrm{d}\mathbf{y}^2. \tag{19}$$

In practise, one has independent, i.i.d. samples from f_1 and f_2, $\{\mathbf{y}_t^1\}_{t=1}^{T_1}$ and $\{\mathbf{y}_t^2\}_{t=1}^{T_2}$, respectively. The goal is to estimate divergence D using these samples. Of course, there exist many variants/extensions of the traditional Kullback-Leibler divergence [84, 5]; depending on the application addressed, different divergences can be advantageous. The ITE package is capable of estimating the following divergences:

1. L_2 divergence:

$$D_{\mathrm{L}}(f_1, f_2) = \sqrt{\int_{\mathbb{R}^d} \left[f_1(\mathbf{y}) - f_2(\mathbf{y})\right]^2 \mathrm{d}\mathbf{y}}. \tag{20}$$

2. Tsallis divergence:

$$D_{\mathrm{T},\alpha}(f_1, f_2) = \frac{1}{\alpha - 1} \left(\int_{\mathbb{R}^d} f_1^\alpha(\mathbf{y}) f_2^{1-\alpha}(\mathbf{y}) \mathrm{d}\mathbf{y} - 1 \right) \quad (\alpha \in \mathbb{R} \setminus \{1\}). \tag{21}$$

3. Rényi divergence:

$$D_{\mathrm{R},\alpha}(f_1, f_2) = \frac{1}{\alpha - 1} \log \int_{\mathbb{R}^d} f_1^\alpha(\mathbf{y}) f_2^{1-\alpha}(\mathbf{y}) \mathrm{d}\mathbf{y} \quad (\alpha \in \mathbb{R} \setminus \{1\}). \tag{22}$$

4. MMD (maximum mean discrepancy) [20]:

$$D_{\mathrm{MMD}}(f_1, f_2) = \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_{\mathcal{F}}^2, \tag{23}$$

where $\boldsymbol{\mu}_m$ is the mean embedding of f_m ($m = 1, 2$) and $\mathcal{F} = \mathcal{F}^1 = \mathcal{F}^2$, see the definition of HSIC [Eq. (10)].

The Kullback-Leibler divergence [Eq. (19)] is a special of Tsallis' and Rényi's in limit sense:

$$\lim_{\alpha \to 1} D_{\mathrm{T},\alpha} = D, \qquad\qquad \lim_{\alpha \to 1} D_{\mathrm{R},\alpha} = D. \tag{24}$$

The calling syntax of the divergence estimators in the ITE package are again uniform. In the following example, the estimation of the Rényi divergence is illustrated using the k-nearest neighbor method:

Example 5 (Divergence estimation (base: usage))

```
>Y1 = randn(3,2000); Y2=randn(3,3000);   %generate the data of interest (d=3, T_1=2000, T_2=3000)
>mult = 1;                                %multiplicative constant is important
>co = DRenyi_kNN_k_initialization(mult); %initialize the divergence ('D') estimator ('Renyi_kNN_k')
>D = DRenyi_kNN_k_estimation(Y1,Y2,co);   %perform divergence estimation
```

13

Estimated quantity	Principle	Name (`cost_name`)
L_2 divergence (D_{L})	k-nearest neighbors ($S = \{k\}$)	`'L2_kNN_k'`
Tsallis divergence ($D_{\mathrm{T},\alpha}$)	k-nearest neighbors ($S = \{k\}$)	`'Tsallis_kNN_k'`
Rényi divergence ($D_{\mathrm{R},\alpha}$)	k-nearest neighbors ($S = \{k\}$)	`'Renyi_kNN_k'`
maximum mean discrepancy (D_{MMD})	norm of the difference of mean embeddings, online	`'MMD_online'`

Table 4: Divergence estimators (base).

Beyond the Rényi divergence $D_{\mathrm{R},\alpha}$ [55, 54, 56], the k-nearest neighbor technique can also be used to estimate the L_2-(D_{L}) [55, 54, 56] and the Tsallis ($D_{\mathrm{T},\alpha}$) divergence [55, 54]. For the MMD measure, a linearly scaling, online method [20] has been implemented in ITE. Table 4 contains the base divergence estimators of the ITE package. The estimations can be carried out by changing the name `'Renyi_kNN_k'` in Example 5 to the `cost_name` in the last column of the table. The samples numbers (T_1 and T_2) in the divergence estimators can be different, except for, of course the online MMD technique, where $T_1 = T_2$.

3.2 Meta Estimators

Here, we present how one can easily derive in the ITE package new information theoretical estimators from existing ones on the basis of relations between entropy, mutual information, divergence. These *meta* estimators are included in ITE. The additional goal of this section is to provide examples for meta estimator construction so that users could simply create novel ones. In Section 3.2.1, Section 3.2.2 and Section 3.2.3 we focus on entropy, mutual information and divergence estimators, respectively.

3.2.1 Entropy Estimators

Here, we present the idea of the meta construction in entropy estimation through examples:

1. The first example considers estimation via the ensemble approach. As it has been recently demonstrated the computational load of entropy estimation can be heavily decreased by (i) dividing the available samples into groups and then (ii) computing the averages of the group estimates [37]. Formally, let the samples be denoted by $\{\mathbf{y}_t\}_{t=1}^{T}$ ($\mathbf{y}_t \in \mathbb{R}^d$) and let us partition them into N groups of size g ($gN = T$), $\{1,\ldots,T\} = \cup_{n=1}^{N} I_n$ ($I_i \cap I_j = \emptyset$, $i \neq j$) and average the estimations based on the groups

$$H_{\mathrm{ensemble}}(\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \hat{H}\left(\{\mathbf{y}_t\}_{t \in I_n}\right). \tag{25}$$

As a prototype example for meta entropy estimation the implementation of the ensemble method [Eq. (25)] is provided below (see Example 6 and Example 7). In the example, the individual estimators in the ensemble are based on k-nearest neighbors (`'Shannon_kNN_k'`). However, the flexibility of the ITE package allows to change the H estimator [r.h.s of (25)] to *any* other entropy technique (base/meta, see Table 2 and Table 5).

Example 6 (Entropy estimation (meta: initialization))

```
function [co] = Hensemble_initialization(mult)
co.name = 'ensemble';              %name of the estimator: 'ensemble'
co.mul = mult;                     %set whether multiplicative constant is important
co.group_size = 500;               %group size (g=500)
co.member_name = 'Shannon_kNN_k';  %estimator used in the ensemble ('Shannon_kNN_k')
co.member_co = H_initialization(co.member_name,mult);%initialize the member in the ensemble,
                                   %the value of 'mult' is passed
```

The estimation part is carried out in accordance with (25):

Example 7 (Entropy estimation (meta: estimation))

```
function [H] = Hensemble_estimation(Y,co)
g = co.group_size;                      %initialize group size (g)
num_of_samples = size(Y,2);             %initialize number of samples (T)
num_of_groups = floor(num_of_samples/g); %initialize number of groups (N)

H = 0;
for k = 1 : num_of_groups               %compute the average over the ensemble
  H = H + H_estimation(Y(:,(k-1)*g+1:k*g),co.member_co); %add the estimation
                                               %of the initialized member
end
H = H / num_of_groups;
```

The usage of the defined method follows the syntax of base entropy estimators (Example 2, Example 3):

Example 8 (Entropy estimation (meta: usage))

```
>Y = rand(5,1000);          %generate the data of interest (d=5, T=1000)
>mult = 1;                  %multiplicative constant is important
>co = Hensemble_initialization(mult); %initialize the entropy ('H') estimator ('ensemble'),
>H = Hensemble_estimation(Y,co);      %perform entropy estimation
```

2. Since (i) entropy can be estimated consistently using pairwise distances of sample points[9], and (ii) random projection (RP) techniques realize approximate isometric embeddings [30, 16, 28, 1, 40, 3, 42], one can construct efficient estimation methods by the integration of the ensemble and the RP technique.

Formally, the definition of the estimation is identical to that of the ensemble approach [Eq. (25)], except for random projections $\mathbf{R}_n \in \mathbb{R}^{d_{RP} \times d}$ ($n = 1, \ldots, N$). The final estimation is

$$H_{\mathrm{RPensemble}}(\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \hat{H}\left(\{\mathbf{R}_n \mathbf{y}_t\}_{t \in I_n}\right). \tag{26}$$

The approach shows exciting potentials with serious computational speed-ups in independent subspace analysis [70] and image registration [71]. The technique has been implemented in the ITE toolbox under the name 'RPensemble' (see Table 5, `HRPensemble_initialization.m`, `HRPensemble_estimation.m`).

3. Information theoretical quantities can be defined over the complex domain via the Hilbert transformation [14]

$$\varphi_v : \mathbb{C}^d \ni \mathbf{v} \mapsto \mathbf{v} \otimes \left[\begin{array}{c} \Re(\cdot) \\ \Im(\cdot) \end{array} \right] \in \mathbb{R}^{2d}, \tag{27}$$

as the entropy of the mapped 2d-dimensional real variable

$$H_{\mathbb{C}}(\mathbf{y}) := H(\varphi_v(\mathbf{y})). \tag{28}$$

Relation (28) can be transformed to a meta entropy estimator, the method is available under the name 'complex' (see Table 5, `Hcomplex_initialization.m`, `Hcomplex_estimation.m`).

The meta entropy estimator methods in ITE are summarized in Table 5. The calling syntax of the estimators is identical to Example 8, one only has to change the name 'ensemble' to the `cost_name` of the target estimators, see the last column of the table.

3.2.2 Mutual Information Estimators

In this section we are dealing with meta mutual information estimators:

1. As it has been seen in (1), mutual information can be expressed via entropy terms. The corresponding method is available in the ITE package under the name 'Shannon_HShannon' (see Table 6, `IShannon_HShannon_initialization.m`, `IShannon_HShannon_estimation.m`). As a prototype example for meta mutual information estimator the implementation is provided below:

---

[9]The construction holds for other information theoretical quantities like mutual information and divergence.

Estimated quantity	Principle	Name (`cost_name`)
complex entropy ($H_\mathbb{C}$)	entropy of a real random vector variable	`'complex'`
Shannon entropy (H)	average the entropy over an ensemble	`'ensemble'`
Shannon entropy (H)	average the entropy over a random projected ensemble	`'RPensemble'`

Table 5: Entropy estimators (meta).

Example 9 (Mutual information estimator (meta: initialization))

```
function [co] = IShannon_HShannon_initialization(mult)
co.name = 'Shannon_HShannon';       %name of the estimator: 'Shannon_HShannon'
co.mul = mult;                      %set the importance of multiplicative factors
co.member_name = 'Shannon_kNN_k';   %method used for entropy estimation: 'Shannon_kNN_k'
co.member_co = H_initialization(co.member_name,1);%initialize entropy estimation member, mult=1
```

Example 10 (Mutual information estimator (meta: estimation))

```
function [I] = IShannon_HShannon_estimation(Y,ds,co) %samples(Y), component dimensions(ds),
                                                     %initialized estimator (co)
num_of_comps = length(ds);                  %number of components, M
cum_ds = cumsum([1;ds(1:end-1)]);           %starting indices of the components
I = -H_estimation(Y,co.member_co);          %minus the joint entropy, H([y^1;...;y^M]) using the
                                            %initialized H estimator
for k = 1 : num_of_comps                    %add the entropy of the y^m components, H(y^m)
  idx = [cum_ds(k) : cum_ds(k)+ds(k)-1];
  I = I + H_estimation(Y(idx,:),co.member_co);%use the initialized H estimator
end
```

The usage of the meta mutual information estimators follow the syntax of base mutual information estimators (see Example 4):

Example 11 (Mutual information estimator (meta: usage))

```
>ds = [1;2]; Y=rand(sum(ds),5000);         %generate the data of interest
                                           %(ds(m)=dim(y^m), T=5000)
>mult = 1;                                 %multiplicative constant is important
>co = IShannon_HShannon_initialization(mult); %initialize the mutual information ('I') estimator
                                           %('Shannon_HShannon')
>I = IShannon_HShannon_estimation(Y,ds,co); %perform mutual information estimation
```

2. The mutual information of complex random variables ($\mathbf{y} \in \mathbb{C}^{d_m}$) can be defined via the Hilbert transformation [Eq. (27)]:

$$I_\mathbb{C}\left(\mathbf{y}^1,\ldots,\mathbf{y}^M\right) = I\left(\varphi_v\left(\mathbf{y}^1\right),\ldots,\varphi_v\left(\mathbf{y}^M\right)\right). \tag{29}$$

The relation is realized in ITE by the `'complex'` meta estimator (see Table 6, `Icomplex_initialization.m`, `Icomplex_estimation.m`).

3. The Shannon-, L_2-, Tsallis- and Rényi mutual information can be expressed in terms of the corresponding divergence of the joint (f) and the product of marginals ($\prod_{m=1}^M f_m$)[10]:

$$I\left(\mathbf{y}^1,\ldots,\mathbf{y}^M\right) = D\left(f,\prod_{m=1}^M f_m\right), \qquad I_L\left(\mathbf{y}^1,\ldots,\mathbf{y}^M\right) = D_\mathrm{L}\left(f,\prod_{m=1}^M f_m\right), \tag{30}$$

$$I_{\mathrm{T},\alpha}\left(\mathbf{y}^1,\ldots,\mathbf{y}^M\right) = D_{\mathrm{T},\alpha}\left(f,\prod_{m=1}^M f_m\right), \qquad I_{\mathrm{R},\alpha}\left(\mathbf{y}^1,\ldots,\mathbf{y}^M\right) = D_{\mathrm{R},\alpha}\left(f,\prod_{m=1}^M f_m\right). \tag{31}$$

---

[10]For the definitions of f and f_ms, see Eq. (5). The divergence definitions can be found in Eqs. (19), (20), (21) and (22).

Estimated quantity	Principle	Name (`cost_name`)
complex mutual information ($I_\mathbb{C}$)	mutual information of a real random vector variable	`'complex'`
L_2 mutual information (I_L)	L_2-divergence of the joint and the product of marginals	`'L2_DL2'`
Rényi mutual information ($I_{\text{R},\alpha}$)	Rényi divergence of the joint and the product of marginals	`'Renyi_DRenyi'`
copula-based kernel dependency (I_c)	MMD div. of the joint copula and the uniform distribution	`'MMD_DMMD'`
Rényi mutual information ($I_{\text{R},\alpha}$)	minus the Rényi entropy of the joint copula	`'Renyi_HRenyi'`
(Shannon) mutual information (I)	entropy sum of the components minus the joint entropy	`'Shannon_HShannon'`
Tsallis mutual information ($I_{\text{T},\alpha}$)	L_2-divergence of the joint and the product of marginals	`'Tsallis_DTsallis'`

Table 6: Mutual information estimators (meta).

Shannon mutual information is a special case of Rényi's and Tsallis' in limit sense:

$$I_{\text{R},\alpha} \xrightarrow{\alpha \to 1} I, \qquad\qquad I_{\text{T},\alpha} \xrightarrow{\alpha \to 1} I. \tag{32}$$

The associated Rényi-, L_2- and Tsallis meta mutual information estimators are available in ITE using the names `'Renyi_DRenyi'`, `'L2_DL2'` and `'Tsallis_DTsallis'` (see Table 6, `IRenyi_DRenyi_initialization.m`, `IRenyi_DRenyi_estimation.m`, `IL2_DL2_initialization.m`, `IL2_DL2_estimation.m`, `ITsallis_DTsallis_initialization.m`, `ITsallis_DTsallis_estimation.m`).

4. [47] has recently defined a novel, robust, copula-based mutual information measure of the random variable $y^m \in \mathbb{R}$ ($m = 1, \ldots, M$) as the MMD divergence [Eq. (23)] of the joint copula and the M-dimensional uniform distribution on $[0,1]^M$:

$$I_c\left(y^1, \ldots, y^M\right) = D_{\text{MMD}}(P_\mathbf{Z}, P_\mathbf{U}), \tag{33}$$

where $\mathbf{Z} = \left[F_1\left(y^1\right); \ldots; F_M\left(y^M\right)\right] \in \mathbb{R}^M$ is the joint copula, F_m is the cumulative density function of y^m and P denotes the distribution. The associated meta estimator available has the name `'MMD_DMMD'` (see Table 6, `IMMD_DMMD_initialization.m`, `IMMD_DMMD_estimation.m`) in ITE.

The calling syntax of the meta mutual information are identical (and the same as that of the base estimators, see Section 3.1.2), the possible methods are summarized in Table 6. The techniques are identified by their `'cost_name'`, see the last column of the table.

3.2.3 Divergence Estimators

In this section we focus on meta divergence estimators (Table 7). Our prototype example is the estimation of the symmetrised Kullback-Leibler divergence, the so-called J-distance (also called the Jensen-Shannon divergence):

$$D_{\text{J}}(f_1, f_2) = D(f_1, f_2) + D(f_2, f_1). \tag{34}$$

The definition of meta divergence estimators follows the idea of meta entropy and mutual information estimators (see Example 6, 7, 9 and 10). Initialization and estimation of the meta J-distance estimator can be carried out as follows:

Example 12 (Divergence estimator (meta: initialization))

```
function [co] = DJdistance_initialization(mult)
co.name = 'Jdistance';              %name of the estimator: 'Jdistance'
co.mult = mult;                     %set whether multiplicative constant is important
co.member_name = 'Renyi_kNN_k';     %method used for Kullback-Leibler divergence estimation
co.member_co = D_initialization(co.member_name,mult); %initialize the Kullback-Leibler divergence
                                                      %estimator
```

Example 13 (Divergence estimator (meta: estimation))

```
function [D_J] = DJdistance_estimation(X,Y,co)
D_J = D_estimation(X,Y,co.member_co) + D_estimation(Y,X,co.member_co); %definition of J-distance
```

Having defined the J-distance estimator, the calling syntax is completely analogous to base estimators (see Example 5).

Estimated quantity	Principle	Name (`cost_name`)
J-distance (D_{J})	symmetrised Kullback-Leibler divergence	`'Jdistance'`

Table 7: Divergence estimators (meta).

Example 14 (Divergence estimator (meta: usage))

```
>Y1 = rand(3,1000); Y2 = rand(3,2000); %generate the data of interest (d=3, T₁=1000, T₂=2000)
>mult = 1;                             %multiplicative constant is important
>co = DJdistance_initialization(mult); %initialize the divergence ('D') estimator ('Jdistance')
>D = DJdistance_estimation(Y1,Y2,co);  %perform divergence estimation
```

3.3   Uniform Syntax of the Estimators

The modularity of the ITE package in terms of (i) the definition and usage of the base/meta entropy, mutual information and divergence estimators, and the possibility to (ii) simple embed novel estimators can be assured by following the templates:

1. Initialization:

   Template 1 (Entropy estimator: initialization)

   ```
   function [co] = H<cost_name>_initialization(mult)
   co.name = <cost_name>;
   co.mult = mult;
   ...
   ```

   Template 2 (Mutual information estimator: initialization)

   ```
   function [co] = I<cost_name>_initialization(mult)
   co.name = <cost_name>
   co.mult = mult;
   ...
   ```

   Template 3 (Divergence estimator: initialization)

   ```
   function [co] = D<cost_name>_initialization(mult)
   co.name = <cost_name>
   co.mult = mult;
   ...
   ```

2. Estimation:

   Template 4 (Entropy estimator: estimation)

   ```
   function [H] = H<cost_name>_estimation(Y,co)
   ...
   ```

   Template 5 (Mutual information estimator: estimation)

   ```
   function [I] = I<cost_name>_estimation(Y,ds,co)
   ...
   ```

Template 6 (Divergence estimator: estimation)

```
function [D] = D<cost_name>_estimation(Y1,Y2,co)
...
```

The unified implementation in the ITE toolbox, makes it possible to use high-level initialization and estimation of the information theoretical quantities. The corresponding functions are

- for initialization: `H_initialization.m`, `I_initialization`, `D_initialization`,

- for estimation: `H_estimation.m`, `I_estimation`, `D_estimation`

following the templates:

```
function [co] = H_initialization(cost_name,mult)
function [co] = I_initialization(cost_name,mult)
function [co] = D_initialization(cost_name,mult)
function [H] = H_estimation(Y,co)
function [I] = I_estimation(Y,ds,co)
function [D] = D_estimation(Y1,Y2,co)
```

Here, the `cost_name` of the entropy, mutual information and divergence estimator can be freely chosen in case of

- entropy: from the last column of Table 2 and Table 5.

- mutual information: from the last column of Table 3 and Table 6.

- divergence: from the last column of Table 4 and Table 7.

By the ITE construction, following for the

- entropy: Template 1 (initialization) and Template 4 (estimation),

- mutual information: Template 2 (initialization) and Template 5 (estimation),

- divergence: Template 3 (initialization) and Template 6 (estimation),

user-defined estimators can be immediately used. Let us demonstrate idea of the high-level initialization and estimation with a simple example, Example 2 can equivalently be written as:[11]

Example 15 (Entropy estimation (high-level, usage))

```
>Y = rand(5,1000);                    %generate the data of interest (d=5, T=1000)
>cost_name = 'Shannon_kNN_k';         %select the objective (Shannon entropy) and
                                      %its estimation method (k-nearest neighbor)
>mult = 1;                            %multiplicative constant is important
>co = H_initialization(cost_name,mult); %initialize the entropy estimator
>H = H_estimation(Y,co);              %perform entropy estimation
```

A more complex example family will be presented in Section 4. There, the basic idea will be the following:

1. Independent subspace analysis and its extensions can be formulated as the optimization of information theoretical quantities. There exist many equivalent formulations (objective functions) in the literature, as well as approximate objectives.

2. Choosing a given objective function, estimators following the template syntaxes (Template 1-6) can be used simply by giving their names (`cost_name`).

3. Moreover, the selected estimator can be immediately used in different optimization algorithms of the objective.

---

[11]One can perform mutual information and divergence estimations similarly.

4 ITE Application in Independent Process Analysis (IPA)

In this section we present an application of the entropy, mutual information and divergence estimators in independent subspace analysis (ISA) and its extensions (IPA, independent process analysis). Application of ITE in IPA serves as an illustrative example, how complex tasks formulated as information theoretical optimization problems can be tackled by the estimators detailed in Section 3.

Section 4.1 formulates the problem domain, the independent process analysis (IPA) problem family. In Section 4.2 the solution methods of IPA are detailed. Section 4.3 is about the Amari-index, which can be used to measure the precision of the IPA estimations. The IPA datasets included in the ITE package are introduced in Section 4.4.

4.1 IPA Models

In Section 4.1.1 we focus on the simplest linear model, which allows hidden, independent multidimensional sources (subspaces), the so-called independent subspace analysis (ISA) problem. Section 4.1.2 is about the extensions of ISA.

4.1.1 Independent Subspace Analysis (ISA)

The ISA problem is defined in the first paragraph. Then (i) the ISA ambiguities, (ii) equivalent ISA objective functions, and (iii) the ISA separation principle are detailed. Thanks to the ISA separation principle one can define many different equivalent *clustering* based ISA objectives and approximations; this is the topic of the next paragraph. ISA optimization methods are presented in the last paragraph.

The ISA equations One may think of independent subspace analysis (ISA)[12] [8, 13] as a cocktail party problem, where (i) more than one group of musicians (sources) are playing at the party, and (ii) we have microphones (sensors), which measure the mixed signals emitted by the sources. The task is to estimate the original sources from the mixed observations only.

Formally, let us assume that we have an observation ($\mathbf{x} \in \mathbb{R}^{D_x}$), which is instantaneous linear mixture ($\mathbf{A}$) of the hidden source ($\mathbf{e}$), that is,

$$\mathbf{x}_t = \mathbf{A}\mathbf{e}_t, \tag{35}$$

where

1. the unknown mixing matrix $\mathbf{A} \in \mathbb{R}^{D_x \times D_e}$ has full column rank,

2. source $\mathbf{e}_t = \left[\mathbf{e}_t^1; \ldots; \mathbf{e}_t^M\right] \in \mathbb{R}^{D_e}$ is a vector concatenated (using Matlab notation ';') of components $\mathbf{e}_t^m \in \mathbb{R}^{d_m}$ ($D_e = \sum_{m=1}^{M} d_m$), subject to the following conditions:

   (a) $\mathbf{e}_t$ is assumed to be i.i.d. (independent identically distributed) in time t,

   (b) there is at most one Gaussian variable among $\mathbf{e}^m$s; this assumption will be referred to as the 'non-Gaussian' assumption, and

   (c) $\mathbf{e}^m$s are independent, that is $I\left(\mathbf{e}^1, \ldots, \mathbf{e}^M\right) = 0$.

The goal of the ISA problem is to eliminate the effect of the mixing ($\mathbf{A}$) with a suitable $\mathbf{W} \in \mathbb{R}^{D_e \times D_x}$ *demixing matrix* and estimate the original source components $\mathbf{e}^m$s by using observations $\{\mathbf{x}_t\}_{t=1}^{T}$ only ($\hat{\mathbf{e}} = \mathbf{W}\mathbf{x}$). If all the $\mathbf{e}^m$ source components are one-dimensional ($d_m = 1, \forall m$), then the independent component analyis (ICA) task [31, 9, 10] is recovered. For $D_x > D_e$ the problem is called *undercomplete*, while the case of $D_x = D_e$ is regarded as *complete*.

The ISA objective function One may assume without loss of generality in case of $D_x \geq D_e$ for the full column rank matrix $\mathbf{A}$ that it is invertible–by applying principal component analysis (PCA) [25]. The estimation of the demixing matrix $\mathbf{W} = \mathbf{A}^{-1}$ in ISA is equivalent to the minimization of the mutual information between the estimated components ($\mathbf{y}^m$),

$$J_{\mathrm{I}}(\mathbf{W}) = I\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right) \to \min_{\mathbf{W} \in GL(D)}, \tag{36}$$

---

[12]ISA is also called multidimensional ICA, independent feature subspace analysis, subspace ICA, or group ICA in the literature. We will use the ISA abbreviation.

where $\mathbf{y} = \mathbf{Wx}$, $\mathbf{y} = \left[\mathbf{y}^1; \ldots; \mathbf{y}^M\right]$, $\mathbf{y}^m \in \mathbb{R}^{d_m}$, $GL(D)$ denotes the set of $D \times D$ sized invertible matrices, and $D = D_e$. The joint mutual information [Eq. (36)] can also be expressed from only *pair-wise* mutual information by recursive methods [12]

$$I\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right) = \sum_{m=1}^{M-1} I\left(\mathbf{y}^m, \left[\mathbf{y}^{m+1}, ..., \mathbf{y}^M\right]\right). \tag{37}$$

Thus, an equivalent information theoretical ISA objective to (36) is

$$J_{\text{Irecursive}}(\mathbf{W}) = \sum_{m=1}^{M-1} I\left(\mathbf{y}^m, \left[\mathbf{y}^{m+1}, ..., \mathbf{y}^M\right]\right) \rightarrow \min_{\mathbf{W} \in GL(D)}. \tag{38}$$

However, since in ISA, it can be assumed without any loss of generality—applying zero mean normalization and PCA—that

- $\mathbf{x}$ and $\mathbf{e}$ are *white*, i.e., their expectation value is zero, and their covariance matrix is the identity matrix ($\mathbf{I}$),

- mixing matrix $\mathbf{A}$ is orthogonal ($\mathbf{A} \in \mathcal{O}^D$), that is $\mathbf{A}^T\mathbf{A} = \mathbf{I}$, and

- the task is complete ($D = D_x = D_e$),

one can restrict the optimization in (36) and (38) to the orthogonal group ($\mathbf{W} \in \mathcal{O}^D$). Under the whiteness assumption, well-known identities of mutual information and entropy expressions [12] show that the ISA problem is equivalent to

$$J_{\text{sumH}}(\mathbf{W}) = \sum_{m=1}^{M} H\left(\mathbf{y}^m\right) \rightarrow \min_{\mathbf{W} \in \mathcal{O}^D}, \tag{39}$$

$$J_{H,I}(\mathbf{W}) = \sum_{m=1}^{M} \sum_{i=1}^{d_m} H(y_i^m) - \sum_{m=1}^{M} I\left(y_1^m, \ldots, y_{d_m}^m\right) \rightarrow \min_{\mathbf{W} \in \mathcal{O}^D}, \tag{40}$$

$$J_{I,I}(\mathbf{W}) = I\left(y_1^1, \ldots, y_{d_M}^M\right) - \sum_{m=1}^{M} I\left(y_1^m, \ldots, y_{d_m}^m\right) \rightarrow \min_{\mathbf{W} \in \mathcal{O}^D}, \tag{41}$$

where $\mathbf{y}^m = \left[y_1^m; \ldots; y_{d_m}^m\right]$.

The ISA ambiguities   Identification of the ISA model is ambiguous. However, the ambiguities of the model are simple: hidden components can be determined up to permutation of the subspaces and up to invertible linear transformations[13] within the subspaces [82].

The ISA separation principle   One of the most exciting and fundamental hypotheses of the ICA research is the ISA separation principle dating back to 1998 [8]: the ISA task can be solved by ICA preprocessing and then clustering of the ICA elements into statistically independent groups. While the extent of this conjecture, is still an open issue, it has recently been rigorously proven for some distribution types [74]. This principle

- forms the basis of the state-of-the-art ISA algorithms,

- can be used to design algorithms that scale well and efficiently estimate the dimensions of the hidden sources and

- can be extended to different linear-, controlled-, post nonlinear-, complex valued-, partially observed systems, as well as to systems with nonparametric source dynamics.

For a recent review on the topic, see [77]. The addressed extension directions are (i) presented in Section 4.1.2, (ii) are covered by the ITE package. In the ITE package the solution of the ISA problem is based on the ISA separation principle, for a demonstration, see `demo_ISA.m`.

---

[13]The condition of invertible linear transformations simplifies to orthogonal transformations for the 'white' case.

Equivalent clustering based ISA objectives and approximations According to the ISA separation principle, the solution of the ISA task, i.e., the *global* optimum of the ISA cost function can be found by permuting/clustering the ICA elements into statistically independent groups. Using the concept of demixing matrices, it is sufficient to explore forms

$$\mathbf{W}_{\mathrm{ISA}} = \mathbf{P}\mathbf{W}_{\mathrm{ICA}},\tag{42}$$

where (i) $\mathbf{P} \in \mathbb{R}^{D \times D}$ is a permutation matrix ($\mathbf{P} \in \mathcal{P}^D$) to be determined, (ii) $\mathbf{W}_{\mathrm{ICA}}$ and $\mathbf{W}_{\mathrm{ISA}}$ is the ICA and ISA demixing matrix, respectively. Thus, assuming that the ISA separation principle holds, and since permuting does not alter the ICA objective [see, e.g., the first term in (40) and (41)], the ISA problem is equivalent to

$$J_{\mathrm{I}}(\mathbf{P}) = I\left(\mathbf{y}^1, \ldots, \mathbf{y}^M\right) \to \min_{\mathbf{P} \in \mathcal{P}^D},\tag{43}$$

$$J_{\mathrm{Irecursive}}(\mathbf{P}) = \sum_{m=1}^{M-1} I\left(\mathbf{y}^m, \left[\mathbf{y}^{m+1}, ..., \mathbf{y}^M\right]\right) \to \min_{\mathbf{P} \in \mathcal{P}^D},\tag{44}$$

$$J_{\mathrm{sumH}}(\mathbf{P}) = \sum_{m=1}^{M} H\left(\mathbf{y}^m\right) \to \min_{\mathbf{P} \in \mathcal{P}^D},\tag{45}$$

$$J_{\mathrm{sum\text{-}I}}(\mathbf{P}) = -\sum_{m=1}^{M} I\left(y_1^m, ..., y_{d_m}^M\right) \to \min_{\mathbf{P} \in \mathcal{P}^D}.\tag{46}$$

Let us note that if our observations are generated by an ISA model then—unlike in the ICA task when $d_m = 1$ ($\forall m$)— pairwise independence is *not* equivalent to mutual independence [10]. However, minimization of the pairwise dependence of the estimated subspaces

$$J_{\mathrm{Ipairwise}}(\mathbf{P}) = \sum_{m_1 \neq m_2} I\left(\mathbf{y}^{m_1}, \mathbf{y}^{m_2}\right) \to \min_{\mathbf{P} \in \mathcal{P}^D}\tag{47}$$

is an efficient approximation in many situations. An alternative approximation is to consider only the pairwise dependence of the coordinates belonging to different subspaces:

$$J_{\mathrm{Ipairwise1d}}(\mathbf{P}) = \sum_{m_1, m_2=1; m_1 \neq m_2}^{M} \sum_{i_1=1}^{d_{m_1}} \sum_{i_2=1}^{d_{m_2}} I\left(y_{i_1}^{m_1}, y_{i_2}^{m_2}\right) \to \min_{\mathbf{P} \in \mathcal{P}^D}.\tag{48}$$

ISA optimization methods Let us fix an ISA objective J [Eq. (43)-(48)]. Our goal is to solve the ISA task, i.e., by the ISA separation principle to find the permutation ($\mathbf{P}$) of the ICA elements minimizing J. Below we list a few possibilities for finding $\mathbf{P}$; the methods are covered by ITE.

Exhaustive way: The possible number of all permutations, i.e., the number of $\mathbf{P}$ matrices is $D!$, where '!' denotes the factorial function. Considering that the ISA cost function is invariant to the exchange of elements *within* the subspaces (see, e.g., (46)), the number of relevant permutations decreases to $\frac{D!}{\prod_{m=1}^{M} d_m!}$. This number can still be enormous, and the related computations could be formidable justifying searches for efficient approximations that we detail below.

Greedy way: Two estimated ICA components belonging to different subspaces are exchanged, if it decreases the value of the ISA cost J, as long as such pairs exist [79].

'Global' way: Experiences show that greedy permutation search is often sufficient for the estimation of the ISA subspaces. However, if the greedy approach cannot find the true ISA subspaces, then global permutation search method of higher computational burden may become necessary [73]: the cross-entropy solution suggested for the traveling salesman problem [58] can be adapted to this case.

Spectral clustering: Now, let us assume that source dimensions (d_m) are not known in advance. The lack of such knowledge causes combinatorial difficulty in such a sense that one should try all possible

$$D = d_1 + \ldots + d_M \quad (d_m > 0, M \leq D)\tag{49}$$

> Construct an undirected graph with nodes corresponding to ICA coordinates and edge weights (similarities) defined by the *pairwise* statistical dependencies, i.e., the mutual information of the estimated ICA elements: $\mathbf{S} = [\hat{I}(\hat{e}_{\mathrm{ICA},i}, \hat{e}_{\mathrm{ICA},j})]_{i,j=1}^{D}$. Cluster the ICA elements, i.e., the nodes using similarity matrix $\mathbf{S}$.

Table 8: Well-scaling approximation for the permutation search problem in the ISA separation theorem in case of unknown subspace dimensions [`estimate_clustering_UD1_S.m`].

dimension allocations to the subspace ($\mathbf{e}^m$) dimensions, where D is the dimension of the hidden source $\mathbf{e}$. The number of these $f(D)$ possibilities grows quickly with the argument, its asymptotic behaviour is known [22, 83]:

$$f(D) \sim \frac{e^{\pi\sqrt{2D/3}}}{4D\sqrt{3}} \tag{50}$$

as $D \to \infty$. An efficient method with good scaling properties has been put forth in [49] for searching the permutation group for the ISA separation theorem (see Table 8). This approach builds upon the fact that the mutual information between different ISA subspaces $\mathbf{e}^m$ is zero due the assumption of independence. The method assumes that coordinates of $\mathbf{e}^m$ that fall into the same subspace can be paired by using the *pairwise dependence of the coordinates.* This approaches can be considered as objective (48), with unknown d_m subspace dimensions. One may carry out the clustering by applying spectral approaches (included in ITE), which are (i) robust and (ii) scale excellently, a single general desktop computer can handle about a million observations (in our case estimated ICA elements) within several minutes [87].

4.1.2 Extensions of ISA

Below we list some extensions of the ISA model and the ISA separation principle. These different extensions, however, can be used in combinations, too. In all these models, (i) the dimension of the source components (d_m) can be different and (ii) one can apply the Amari-index as the performance measure (Section 4.3). The ITE package directly implements the estimation of the following models[14] (the relations of the different models are summarized in Fig.1):

Linear systems:

AR-IPA:

Equations, assumptions: In the AR-IPA (autoregressive-IPA) task [26] ($d_m = 1$, $\forall m$), [50] ($d_m \geq 1$), the traditional *i.i.d.* assumption for the sources is generalized to AR time series: the hidden sources ($\mathbf{s}^m \in \mathbb{R}^{d_m}$) are not necessarily independent in time, only their driving noises ($\mathbf{e}^m \in \mathbb{R}^{d_m}$) are. The observation ($\mathbf{x} \in \mathbb{R}^D$, $D = \sum_{m=1}^{M} d_m$) is an instantaneous linear mixture ($\mathbf{A}$) of the source $\mathbf{s}$:

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t, \qquad\qquad \mathbf{s}_t = \sum_{i=1}^{L_s} \mathbf{F}_i \mathbf{s}_{t-i} + \mathbf{e}_t, \tag{51}$$

where L_s is the order of the AR process, $\mathbf{s}_t = [\mathbf{s}_t^1; \ldots; \mathbf{s}_t^M]$ and $\mathbf{e}_t = [\mathbf{e}_t^1; \ldots; \mathbf{e}_t^M] \in \mathbb{R}^D$ denote the hidden sources and the hidden driving noises, respectively. (51) can be rewritten in the following concise form:

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \qquad\qquad \mathbf{F}[z]\mathbf{s} = \mathbf{e} \tag{52}$$

using the polynomial of the time-shift operator $\mathbf{F}[z] := \mathbf{I} - \sum_{i=1}^{L_s} \mathbf{F}_i z^i \in \mathbb{R}[z]^{D \times D}$ [38]. We assume that
   1. polynomial matrix $\mathbf{F}[z]$ is *stable*, that is $\det(\mathbf{F}[z]) \neq 0$, for all $z \in \mathbb{C}, |z| \leq 1$,
   2. mixing matrix $\mathbf{A} \in \mathbb{R}^{D \times D}$ is invertible ($\mathbf{A} \in GL(D)$),
   3. $\mathbf{e}$ satisfies the ISA assumptions (see Section 4.1.1)

Goal: The aim of the AR-IPA task is to estimate hidden sources $\mathbf{s}^m$, dynamics $\mathbf{F}[z]$, driving noises $\mathbf{e}^m$ and mixing matrix $\mathbf{A}$ or its $\mathbf{W}$ inverse given observations $\{\mathbf{x}_t\}_{t=1}^T$. For the special case of $L_s = 0$, the ISA task is obtained.

---

[14]The ITE package includes demonstrations for all the touched directions. The name of the demo files are specified at the end the problem definitions, see paragraphs 'Separation principle'.

Separation principle: The AR-IPA estimation can be carried out by (i) applying AR fit to observation $\mathbf{x}$, (ii) followed by ISA on the estimated innovation of $\mathbf{x}$ [26, 50]. Demo: `demo_AR_IPA.m`.

MA-IPA:

Equations, assumptions: Here, the assumption on *instantaneous* linear mixture of the ISA model is weakened to convolutions. This problem is called moving average independent process analysis (MA-IPA, also known as blind subspace deconvolution) [74]. We describe this task for the undercomplete case. Assume that the convolutive mixture of hidden sources $\mathbf{e}^m \in \mathbb{R}^{d_m}$ is available for observation ($\mathbf{x} \in \mathbb{R}^{D_x}$)

$$\mathbf{x}_t = \sum_{l=0}^{L_e} \mathbf{H}_l \mathbf{e}_{t-l}, \tag{53}$$

where

1. $D_x > D_e$ (undercomplete, $D_e = \sum_{m=1}^{M} d_m$),
2. the polynomial matrix $\mathbf{H}[z] = \sum_{l=0}^{L_e} \mathbf{H}_l z^l \in \mathbb{R}[z]^{D_x \times D_e}$ has a (polynomial matrix) left inverse[15] and
3. source $\mathbf{e} = [\mathbf{e}^1; \ldots; \mathbf{e}^M] \in \mathbb{R}^{D_e}$ satisfies the conditions of ISA.

Goal: The goal of this undercomplete MA-IPA problem (uMA-IPA problem, where 'u' stands for undercomplete) is to estimate the original $\mathbf{e}^m$ sources by using observations $\{\mathbf{x}_t\}_{t=1}^T$ only. The case $L_e = 0$ corresponds to the ISA task, and in the blind source deconvolution problem [46] $d_m = 1$ ($\forall m$), and L_e is a non-negative integer.

Note: We note that in the ISA task the full column rank of matrix $\mathbf{H}_0$ was presumed, which is equivalent to the assumption that matrix $\mathbf{H}_0$ has left inverse. This left inverse assumption is extended in the uMA-IPA model for the polynomial matrix $\mathbf{H}[z]$.

Separation principle:

- By applying temporal concatenation (TCC) on the observation, one can reduce the uMA-IPA estimation problem to ISA [74]. Demo: `demo_uMA_IPA_TCC.m`.
- However, upon applying the TCC technique, the associated ISA problem can easily become 'high dimensional'. This dimensionality problem can be alleviated by the linear prediction approximation (LPA) approach, i.e., AR fit, followed by ISA on the estimation innovation [75]. Demo: `demo_uMA_IPA_LPA.m`.
- In the complete ($D_x = D_e$) case, the $\mathbf{H}[z]$ polynomial matrix does not have (polynomial matrix) left inverse in general. However, provided that the convolution can be represented by an infinite order autoregressive [AR(∞)] process, one [66] can construct an efficient estimation method for the hidden components via an asymptotically consistent LPA procedure augmented with ISA. Such AR(∞) representation can be guaranteed by assuming the stability of $\mathbf{H}[z]$ [17]. Demo: `demo_MA_IPA_LPA.m`.

Post nonlinear models:

Equations, assumptions: In the post nonlinear ISA (PNL-ISA) problem [78] the *linear* mixing assumption of the ISA model is alleviated. Assume that the observations ($\mathbf{x} \in \mathbb{R}^D$) are post nonlinear mixtures ($\mathbf{g}(\mathbf{A}\cdot)$) of multidimensional independent sources ($\mathbf{e} \in \mathbb{R}^D$):

$$\mathbf{x}_t = \mathbf{g}(\mathbf{A}\mathbf{e}_t), \tag{54}$$

where the

- unknown function $\mathbf{g} : \mathbb{R}^D \to \mathbb{R}^D$ is a component-wise transformation, i.e, $\mathbf{g}(\mathbf{v}) = [g_1(v_1); \ldots; g_D(v_D)]$ and $\mathbf{g}$ is invertible, and
- mixing matrix $\mathbf{A} \in \mathbb{R}^{D \times D}$ and hidden source $\mathbf{e}$ satisfy the ISA assumptions.

Goal: The PNL-ISA problem is to estimate the hidden source components $\mathbf{e}^m$ knowing only the observations $\{\mathbf{x}_t\}_{t=1}^T$. For $d_m = 1$, we get back the PNL-ICA problem [80] (for a review see [32]), whereas '$\mathbf{g}$=identity' leads to the ISA task.

Separation principle: the estimation of the PNL-ISA problem can be carried out on the basis of the mirror structure of the task, applying gaussianization followed by linear ISA [78]. Demo: `demo_PNL_ISA.m`.

---

[15]One can show for $D_x > D_e$ that under mild conditions $\mathbf{H}[z]$ has a left inverse with probability 1 [57]; e.g., when the matrix $[\mathbf{H}_0, \ldots, \mathbf{H}_{L_e}]$ is drawn from a continuous distribution.

Complex models:

Equations, assumptions: One can define the independence, mutual information and entropy of complex random variables via the Hilbert transformation [Eq. (27), (28), (29)]. Having these definitions at hand, the complex ISA problem can be formulated analogously to the real case, the observations ($\mathbf{x}_t \in \mathbb{C}^D$) are generated as the instantaneous linear mixture ($\mathbf{A}$) of the hidden sources ($\mathbf{e}_t$):

$$\mathbf{x}_t = \mathbf{A}\mathbf{e}_t, \tag{55}$$

where

- the unknown $\mathbf{A} \in \mathbb{C}^{D \times D}$ mixing matrix is invertible ($D = \sum_{m=1}^{M} d_m$),
- $\mathbf{e}_t$ is assumed to be i.i.d. in time t,
- $\mathbf{e}^m \in \mathbb{C}^{d_m}$s are independent, that is $I\left(\varphi_v\left(\mathbf{e}^1\right), \ldots, \varphi_v\left(\mathbf{e}^M\right)\right) = 0$.

Goal: The goal is to estimate the hidden source $\mathbf{e}$ and the mixing matrix $\mathbf{A}$ (or its $\mathbf{W} = \mathbf{A}^{-1}$ inverse) using the observation $\{\mathbf{x}_t\}_{t=1}^{T}$. If all the components are one-dimensional ($d_m = 1$, $\forall m$), one obtains the complex ICA problem.

Separation principle:

- Supposing that the $\varphi_v(\mathbf{e}^m) \in \mathbb{R}^{2d_m}$ variables are 'non-Gaussian', and exploiting the operation preserving property of the Hilbert transformation the solution of the complex ISA problem can be reduced to a ISA task over the real domain with observation $\varphi_v(\mathbf{x})$ and M pieces of $2d_m$-dimensional hidden components $\varphi_v(\mathbf{e}^m)$. The consideration can be extended to *linear models* including AR, MA, ARMA (autoregressive moving average), ARIMA (integrated ARMA), ...terms [69]. Demo: `demo_complex_ISA.m`.
- Another possible solution is to apply the ISA separation theorem, which remains valid even for complex variables [74]: the solution can be accomplished by complex ICA and clustering of the complex ICA elements. Demo: `demo_complex_ISA_C.m`.

Controlled models:

Equations, assumptions: In the *ARX-IPA* (ARX–autoregressive with exogenous input) problem [68] the AR-IPA assumption holds (Eq. (51)), but the time evolution of the hidden source $\mathbf{s}$ can be influenced via *control* variable $\mathbf{u}_t \in \mathbb{R}^{D_u}$ through matrices $\mathbf{B}_j \in \mathbb{R}^{D \times D_u}$:

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t \qquad \mathbf{s}_t = \sum_{i=1}^{L_s} \mathbf{F}_i \mathbf{s}_{t-i} + \sum_{j=1}^{L_u} \mathbf{B}_j \mathbf{u}_{t+1-j} + \mathbf{e}_t. \tag{56}$$

Goal: The goal is to estimate the hidden source $\mathbf{s}$, the driving noise $\mathbf{e}$, the parameters of the dynamics and control matrices ($\{\mathbf{F}_i\}_{i=1}^{L_s}$ and $\{\mathbf{B}_j\}_{j=1}^{L_u}$), as well as the mixing matrix $\mathbf{A}$ or its inverse $\mathbf{W}$ by using observations $\mathbf{x}_t$ and controls $\mathbf{u}_t$. In the special case of $L_u = 0$, the ARX-IPA task reduces to AR-IPA.

Separation principle: The solution can be reduced to ARX identification followed by ISA [68]. Demo: `demo_ARX_IPA.m`.

Partially observed models:

Equations, assumptions: In the *mAR-IPA* (mAR–autoregressive with missing values) problem [67], the AR-IPA assumptions (Eq. (51)) are relaxed by allowing a few coordinates of the mixed AR sources $\mathbf{x}_t \in \mathbb{R}^D$ to be *missing* at certain time instants. Formally, we observe $\mathbf{y}_t \in \mathbb{R}^D$ instead of $\mathbf{x}_t$, where 'mask mappings' $\boldsymbol{\mathcal{M}}_t : \mathbb{R}^D \mapsto \mathbb{R}^D$ represent the coordinates and the time indices of the non-missing observations:

$$\mathbf{y}_t = \boldsymbol{\mathcal{M}}_t(\mathbf{x}_t), \qquad \mathbf{x}_t = \mathbf{A}\mathbf{s}_t, \qquad \mathbf{s}_t = \sum_{i=1}^{L_s} \mathbf{F}_i \mathbf{s}_{t-i} + \mathbf{e}_t. \tag{57}$$

Goal: Our task is the estimation of the hidden source $\mathbf{s}$, its driving noise $\mathbf{e}$, parameters of the dynamics $\mathbf{F}[z]$, mixing matrix $\mathbf{A}$ (or its inverse $\mathbf{W}$) from observation $\{\mathbf{y}_t\}_{t=1}^{T}$. The special case of '$\boldsymbol{\mathcal{M}}_t =$ identity' corresponds to the AR-IPA task.

ARX-IPA

$L_u=0$

mAR-IPA $\xrightarrow{\mathbf{M}_t:\text{identity}(\forall t)}$ AR-IPA $\xleftarrow{\mathbf{f}:\text{ known, linear}}$ fAR-IPA

$L_s=0$

MA-IPA (BSSD) $\xrightarrow{L_e=0}$ ISA (I.I.D.-IPA) $\xleftarrow{\mathbf{g}:\text{ known, identity}}$ PNL-ISA

$d_m=1(\forall m)$ $d_m=1(\forall m)$ $d_m=1(\forall m)$

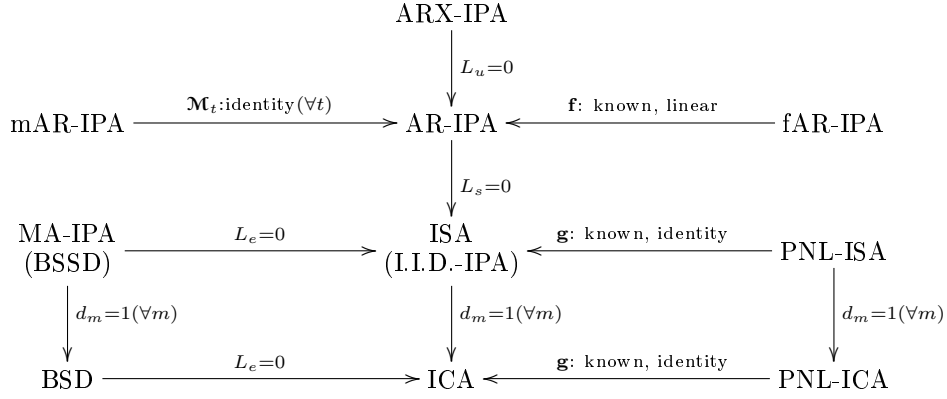BSD $\xrightarrow{L_e=0}$ ICA $\xleftarrow{\mathbf{g}:\text{ known, identity}}$ PNL-ICA

Figure 1: IPA problem family, relations. Arrows point to special cases. For example, 'ISA$\xrightarrow{d_m=1(\forall m)}$ICA' means that ICA is a special case of ISA, when all the source components are one-dimensional.

Separation principle: One can reduce the solution to mAR identification followed by ISA on the estimated innovation process [67]. Demo: `demo_mAR_IPA.m`.

Models with nonparametric dynamics:

Equations, assumptions: In the *fAR-IPA* (fAR–functional autoregressive) problem [72], the *parametric* assumption for the dynamics of the hidden sources is circumvented by functional AR sources:

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t, \qquad\qquad \mathbf{s}_t = \mathbf{f}(\mathbf{s}_{t-1},\ldots,\mathbf{s}_{t-L_s}) + \mathbf{e}_t. \qquad (58)$$

Goal: The goal is to estimate the hidden sources $\mathbf{s}^m \in \mathbb{R}^{d_m}$ including their dynamics $\mathbf{f}$ and their driving innovations $\mathbf{e}^m \in \mathbb{R}^{d_m}$ as well as mixing matrix $\mathbf{A}$ (or its inverse $\mathbf{W}$) given observations $\{\mathbf{x}_t\}_{t=1}^T$. If we knew the parametric form of $\mathbf{f}$ and if it were linear, then the problem would be AR-IPA.

Separation principle: The problem can be solved by nonparametric regression followed by ISA [72]. Demo: `demo_fAR_IPA.m`.

4.2   Estimation via ITE

Having (i) the information theoretical estimators (Section 3), (ii) the ISA/IPA problems and separation principles (Section 4.1) at hand, we now detail the solution methods offered by the ITE package. Due the *separation principles* of the IPA problem family, the solution methods can be implemented in a completely *modular* way; the estimation techniques can be built up from the solvers of the obtained *subproblems*. From developer point of view, this flexibility makes it possible to easily modify/extend the ITE toolbox. For example, (i) in case of ISA, one can select/replace the ICA method and clustering technique applied independently, (ii) in case of AR-IPA one has freedom in chosing/extending the AR identificator and the ISA solver, etc. This is the underlying idea of the solvers offered by the ITE toolbox.

In Section 4.2.1 the solution techniques for the ISA task are detailed. Extensions of the ISA problem are in the focus of Section 4.2.2.

4.2.1   ISA

As it has been detailed in Section 4.1.1, the ISA problem can be formulated as the optimization of information theoretical objectives (see Eqs. (43), (44), (45), (46), (47), (48)). In the ITE package,

All the detailed ISA formulations:

- are available by the appropriate choice of the variable `cost_type` (see Table 9), and
- can be used by *any* entropy/mutual information estimator satisfying the ITE template construction (see Table 2, Table 3, Table 5, Table 6 and Section 3.3).

Cost function to minimize	Name (`cost_type`)
$I\left(\mathbf{y}^1,\ldots,\mathbf{y}^M\right)$	`'I'`
$\sum_{m=1}^{M} H\left(\mathbf{y}^m\right)$	`'sumH'`
$-\sum_{m=1}^{M} I\left(y_1^m,\ldots,y_{d_m}^M\right)$	`'sum-I'`
$\sum_{m=1}^{M-1} I\left(\mathbf{y}^m,\left[\mathbf{y}^{m+1},\ldots,\mathbf{y}^M\right]\right)$	`'Irecursive'`
$\sum_{m1\neq m2} I\left(\mathbf{y}^{m_1},\mathbf{y}^{m_2}\right)$	`'Ipairwise'`
$\sum_{m_1,m_2=1;m_1\neq m_2}^{M} \sum_{i_1=1}^{d_{m_1}} \sum_{i_2=1}^{d_{m_2}} I\left(y_{i_1}^{m_1},y_{i_2}^{m_2}\right)$	`'Ipairwise1d'`

Table 9: ISA formulations. $1-4^{th}$ row: equivalent, $5-6^{th}$ row: necessary conditions.

Optimization technique (`opt_type`)	Principle	Environment
`'NCut'`	normalized cut	Matlab
`'SP1'`	unnormalized cut	Matlab, Octave
`'SP2'`, `'SP3'`	2 normalized cut methods	Matlab, Octave

Table 10: Spectral clustering optimizers for given number of subspaces (M) [`unknown_dimensions=1`]: `clustering_UD1.m`: `estimate_clustering_UD1_S.m`.

The dimension of the subspaces can be given/unknown: the priori knowledge about the dimension of the subspaces can be conveyed by the variable `unknown_dimensions`. `unknown_dimensions=0` (=1) means given $\{d_m\}_{m=1}^{M}$ subspace dimensions (unknown subspace dimensions, it is sufficient to give M, the number of subspaces). In case of

- given subspace dimensions: clustering of the ICA elements can be carried out in ITE by the exhaustive (`opt_type = 'exhaustive'`), greedy (`opt_type = 'greedy'`), or the cross-entropy (`opt_type = 'CE'`) method.

- unknown subspace dimensions: clustering of the ICA elements can be performed by applying spectral clustering. In this case, the clustering is based on the pairwise mutual information of the one-dimensional ICA elements (Table 9) and the objective is (48), i.e., `cost_type = 'Ipairwise1d'`. The ITE package supports 4 different spectral clustering methods/implementations (Table 10):

  - the unnormalized cut method (`opt_type = 'SP1'`), and two normalized cut techniques (`opt_type = 'SP2'` or `opt_type = 'SP3'`) [63, 45, 85] – the implemetations are purely Matlab/Octave, and

  - a fast, normalized cut implementation [63, 11] in C++ with compilable mex files (`opt_type = 'NCut'`).

The ISA estimator capable of handling these options is called `estimate_ISA.m`, and is accompanied by the demo file `demo_ISA.m`. Let us take some examples for the parameters to set in `demo_ISA.m`:

Example 16 (ISA-1)

- *Goal: the subspace dimensions $\{d_m\}_{m=1}^{M}$ are known; apply sum of entropy based ISA formulation (Eq. (45)); estimate the entropy via the Rényi entropy using k-nearest neighbors ($S = \{1,\ldots,k\}$); optimize the objective in a greedy way.*

- *Parameters to set:* `unknown_dimensions = 0;` `cost_type = 'sumH';` `cost_name = 'Renyi_kNN_1tok',` `opt_type = 'greedy'.`

Example 17 (ISA-2)

- *Goal: the subspace dimensions $\{d_m\}_{m=1}^{M}$ are known; apply an ISA formulation based on the sum of mutual information within the subspaces (Eq. (46)); estimate the mutual information via the KCCA method; optimize the objective in a greedy way.*

- *Parameters to set:* `unknown_dimensions = 0;` `cost_type = 'sum-I';` `cost_name = 'KCCA',` `opt_type = 'greedy'.`

Cost type (`cost_type`)	Recommended/chosen optimizer
'I', 'Irecursive'	`clustering_UD0_greedy_general.m`
'sumH', 'sum-I'	`clustering_UD0_greedy_additive_wrt_subspaces.m`
'Ipairwise'	`clustering_UD0_greedy_pairadditive_wrt_subspaces.m`
'Ipairwise1d'	`clustering_UD0_greedy_pairadditive_wrt_coordinates.m`

Table 11: Recommended/chosen optimizers for given subspace dimensions ($\{d_m\}_{m=1}^M$) [`unknown_dimensions=0`] applying greedy [`opt_type='greedy'`] ISA optimization: `clustering_UD0.m`.

Cost type (`cost_type`)	Recommended/chosen optimizer
'I', 'sumH', 'sum-I', 'Irecursive', 'Ipairwise'	`clustering_UD0_CE_general.m`
'Ipairwise1d'	`clustering_UD0_CE_pairadditive_wrt_coordinates.m`

Table 12: Recommended/chosen optimizers for given subspace dimensions ($\{d_m\}_{m=1}^M$) [`unknown_dimensions=0`] applying cross-entropy [`opt_type='CE'`] ISA optimization: `clustering_UD0.m`.

Example 18 (ISA-3)

- *Goal: the subspace dimensions are unknown, only M, the number of the subspaces is given; the ISA objective is based on the pairwise mutual information of the estimated ICA elements (Eq. (48)); estimate the mutual information using the KGV method; optimize the objective via the NCut normalized cut method.*

- *Parameters to set:* `unknown_dimensions = 0; cost_type = 'KGV'; cost_name = 'KGV', opt_type = 'NCut'`.

In case of given subspace dimensions, the special structure of the ISA objectives can be taken into account to further increase the efficiency of the **optimization**, i.e., the clustering step. The ITE package realizes this idea:

- In case of (i) one-dimensional mutual information based ISA formulation (Eq. (48)), and (ii) cross-entropy or exhaustive optimization the $\mathbf{S} = [I(\hat{e}_{\text{ICA,i}}, \hat{e}_{\text{ICA,j}})]_{i,j=1}^D$ similarity matrix can be precomputed.

- In case of greedy optimization:

    - upon applying ISA objective (48), the $\mathbf{S} = [I(\hat{e}_{\text{ICA,i}}, \hat{e}_{\text{ICA,j}})]_{i,j=1}^D$ similarity matrix can again be precomputed giving rise to more efficient optimization.
    - ISA formulations (45), (46) are both additive w.r.t. the estimated subspaces. Making use of this special structure of these objective, it is sufficient to recompute the objective only on the touched subspaces while greedily testing a new permutation candidate. Provided that the number of the subspaces (M) is high, the decreased computational load of the specialized method is emphasized.
    - objective (47) is pair-additive w.r.t. the subspaces. In this case, it is enough to recompute the objective on the subspaces connected the actual subspace estimates. Again the increased efficiency is striking in case of large number of subspaces.

The general and the recommended (which are chosen by default in the toolbox) ISA optimization methods of ITE are listed Table 11 (greedy), Table 12 (cross-entropy), Table 13 (exhaustive).

Extending the capabilities of the ITE toolbox: In case of

Cost type (`cost_type`)	Recommended/chosen optimizer
'I', 'sumH', 'sum-I', 'Irecursive', 'Ipairwise'	`clustering_UD0_exhaustive_general.m`
'Ipairwise1d'	`clustering_UD0_exhaustive_pairadditive_wrt_coordinates.m`

Table 13: Recommended/chosen optimizers for given subspace dimensions ($\{d_m\}_{m=1}^M$) [`unknown_dimensions=0`] applying exhaustive [`opt_type='exhaustive'`] ISA optimization: `clustering_UD0.m`.

- known subspaces dimensions ($\{d_m\}_{m=1}^{M}$): the clustering is carried out in `clustering_UD0.m`. Before clustering, first the importance of the constant multipliers must be set in `set_mult.m`.[16]
  - To add a new ISA formulation (`cost_type`):
    * to be able to carry it out general optimization: it is sufficient to add the new `cost_type` entry to `clustering_UD0.m`, and the computation of the new objective to `cost_general.m`.
    * to be able to perform an existing, specialized (not general) optimization: add the new `cost_type` entry to `clustering_UD0.m`, and the computation of the new objective to the corresponding cost procedure. For example, in case of a new objective being additive w.r.t. subspaces (similarly to (45), (46)) it is sufficient to modify `cost_additive_wrt_subspaces_one_subspace.m` in `cost_additive_wrt_subspaces.m`.
    * to be able to perform a non-existing optimization: add the new `cost_type` entry to `clustering_UD0.m` with the specialized solver.
  - To add a new optimization method (`opt_type`): please follow the 3 examples included in `clustering_UD0.m`.
- unknown subspace dimensions (M): `clustering_UD1.m` is responsible for the clustering step. It first computes the $\mathbf{S} = [\hat{I}(\hat{e}_{\text{ICA},i}, \hat{e}_{\text{ICA},j})]_{i,j=1}^{D}$ similarity matrix, and then performs spectral clustering (see Table 8). To include a new clustering technique, one only has to add it to a new case in `estimate_clustering_UD1_S.m`.

4.2.2 Extensions of ISA

Due to the IPA separation principles, the solution of the problem family can be carried out in a *modular* way. The solution of all the presented IPA directions are demonstrated through examples in ITE, the demo files and the actual estimators are listed in Table 14. For the obtained subtasks the ITE package provides many efficient estimators (see Table 15):

ICA, complex ICA: The fastICA method [27] and its complex variant [7] is one of the most popular ICA approach, it is available in ITE. See `estimate_ICA.m` and `estimate_complex_ICA.m`.

AR identification: Identification of AR processes can be carried in the ITE toolbox in 5 different ways (see `estimate_AR.m`):

- using the online Bayesian technique with normal-inverted Wishart prior [33, 48],
- applying [29]
  - nonlinear least squares estimator based on the subspace representation of the system,
  - exact maximum likelihood optimization using the BFGS (Broyden-Fletcher-Goldfarb-Shannon; or the Newton-Raphson) technique,
  - the combination of the previous two approaches.
- making use of the stepwise least squares technique [44, 60].

ARX identification: Identification of ARX processes can be carried out by the D-optimal technique of [48] assuming normal-inverted Wishart prior; see `estimate_ARX_IPA.m`.

mAR identification: The

- online Bayesian technique with normal-inverted Wishart prior [33, 48],
- nonlinear least squares [29],
- exact maximum likelihood [29], and
- their combination [29]

are available for the identification of mAR processes; see `estimate_mAR.m`.

fAR identification: Identification of fAR processes in ITE can be carried out by the strongly consistent, recursive Nadaraya-Watson estimator [23]; see `estimate_fAR.m`.

spectral clustering: The ITE toolbox provides 4 methods to perform spectral clustering (see `estimate_clustering_UD1_S.m`):

---

[16]For example, upon applying objective (45) multiplicative constants are irrelevant (important) in case of equal (different) d_m subspace dimensions.

IPA model	Reduction		Demo (Estimator)
	Task1	Task2	
ISA	ICA	clustering of the ICA elements	`demo_ISA.m` `(estimate_ISA.m)`
AR-IPA	AR fit	ISA	`demo_AR_IPA.m` `(estimate_AR_IPA.m)`
ARX-IPA	ARX fit	ISA	`demo_ARX_IPA.m` `(estimate_ARX_IPA.m)`
mAR-IPA	mAR fit	ISA	`demo_mAR_IPA.m` `(estimate_mAR_IPA.m)`
complex ISA	Hilbert transformation	real ISA	`demo_complex_ISA.m` `(estimate_complex_ISA.m)`
complex ISA	complex ICA	clustering of the ICA elements	`demo_complex_ISA_C.m` `(estimate_complex_ISA_C.m)`
fAR-IPA	nonparametric regression	ISA	`demo_fAR_IPA.m` `(estimate_fAR_IPA.m)`
(complete) MA-IPA	linear prediction (LPA)	ISA	`demo_MA_IPA_LPA.m` `(estimate_MA_IPA_LPA.m)`
undercomplete MA-IPA	temporal concatenation (TCC)	ISA	`demo_uMA_IPA_TCC.m` `(estimate_uMA_IPA_TCC.m)`
undercomplete MA-IPA	linear prediction (LPA)	ISA	`demo_uMA_IPA_LPA.m` `(estimate_uMA_IPA_LPA.m)`
PNL-ISA	gaussianization	ISA	`demo_PNL_ISA.m` `(estimate_PNL_ISA.m)`

Table 14: IPA separation principles.

- the unnormalized cut method, and two normalized cut techniques [63, 45, 85] – the implemetations are purely Matlab/Octave, and
- a fast, normalized cut implementation [63, 11] in C++ with compilable mex files.

gaussianization: Gaussianization of the observations can be carried out by the efficient rank method [89], see `estimate_gaussianization.m`.

Extending the capabilities of the ITE toolbox: additional methods for the obtained subtasks can be easily embedded and instantly used in IPA, by simply adding a new 'switch: case' entry to the subtask solvers listed in Table 15. Beyond the solvers for the IPA subproblems detailed above, the ITE toolbox offers:

- 4 different alternatives for *k-nearest neighbor* estimation (Table 16):
  - exact nearest neighbors: based on fast computation of pairwise distances and C++ partial sort (knn package).
  - exact nearest neighbors: based on fast computation of pairwise distances.

Subtask	Estimator	Method
ICA	`estimate_ICA.m`	`'fastICA'`
complex ICA	`estimate_complex_ICA.m`	`'fastICA'`
AR fit (LPA)	`estimate_AR.m`	`'NIW'`, `'subspace'`, `'subspace-LL'`, `'LL'`, `'stepwiseLS'`
ARX fit	`estimate_ARX.m`	`'NIW'`
mAR fit	`estimate_mAR.m`	`'NIW'`, `'subspace'`, `'subspace-LL'`, `'LL'`
fAR fit	`estimate_fAR.m`	`'recursiveNW'`
spectral clustering	`estimate_clustering_UD1_S.m`	`'NCut'`, `'SP1'`, `'SP2'`, `'SP3'`
gaussianization	`estimate_gaussianization.m`	`'rank'`

Table 15: IPA subtasks and estimators.

co.kNNmethod	Principle	Environment
'knnFP1'	exact NNs, fast pairwise distance computation and C++ partial sort	Matlab, Octave
'knnFP2'	exact NNs, fast pairwise distance computation	Matlab, Octave
'knnsearch'	exact NNs, Statistics Toolbox ∈ Matlab	Matlab
'ANN'	approximate NNs, ANN library	Matlab, Octave[a]

Table 16: k-nearest neighbor (kNN) methods. The main kNN function is `kNN_squared_distances.m`.

[a]See Table 1.

co.MSTmethod	Method	Environment
'MatlabBGL_Prim'	Prim algorithm (MatlabBGL)	Matlab, Octave[a]
'MatlabBGL_Kruskal'	Kruskal algorithm (MatlabBGL)	Matlab, Octave
'pmtk3_Prim'	Prim algorithm (pmtk3)	Matlab, Octave
'pmtk3_Kruskal'	Kruskal algorithm (pmtk3)	Matlab, Octave

Table 17: Minimum spanning tree (MST) methods. The main MST function is `compute_MST.m`.

[a]See Table 1.

- exact nearest neighbors: carried out by the `knnsearch` function of the Statistics Toolbox in Matlab.
- approximate nearest neighbors: implemented by the ANN library.

The method applied for the estimation can be chosen by setting `co.method` to 'knnFP1', 'knnFP2', 'knnsearch', or 'ANN'. For examples, please see:

- `HRenyi_GSF_initialization.m`, `HShannon_kNN_k_initialization.m`, `HRenyi_kNN_1tok_initialization.m`, `HRenyi_kNN_k_initialization.m`, `HRenyi_kNN_S_initialization.m`, `HRenyi_weightedkNN_initialization.m`,
- `DL2_kNN_k_initialization.m`, `DRenyi_kNN_k_initialization.m`, `DTsallis_kNN_k_initialization.m`.

The central function of kNN computations is `kNN_squared_distances.m`.

- 4 techniques for *minimum spanning tree* computation (Table 17):

  - the two functions of the MatlabBGL library can be envoked by setting `co.STmethod` to 'MatlabBGL_Prim' or 'MatlabBGL_Kruskal'.
  - the purely Matlab/Octave implementations based on the pmtk3 toolbox can be called by setting `co.STmethod` to 'pmtk3_Prim' or 'pmtk3_Kruskal'.

  For an example, please see `H_Renyi_MST_initialization.m`. The central function for MST computation is `compute_MST.m`.

To **extend** the capabilities of ITE in k-nearest neighbor or minimum spanning tree computation (which is also immediately inherited to entropy, mutual information, divergence estimation), it sufficient to the add the new method to `kNN_squared_distances.m` or `compute_MST.m`.

4.3 Performance Measure, the Amari-index

Here, we introduce the Amari-index, which can be used to measure the efficiency of the estimators in the ISA problem and its extensions.

Identification of the ISA model is ambiguous. However, the ambiguities of the model are simple: hidden components can be determined up to permutation of the subspaces and up to invertible linear transformations within the subspaces [82]. Thus, in the ideal case, the product of the estimated ISA demixing matrix $\hat{\mathbf{W}}_{\text{ISA}}$ and the ISA mixing matrix $\mathbf{A}$, i.e., matrix

$$\mathbf{G} = \hat{\mathbf{W}}_{\text{ISA}} \mathbf{A} \tag{59}$$

is a block-permutation matrix (also called block-scaling matrix [81]). This property can also be measured for source components with different dimensions by a simple extension [72] of the Amari-index [2], that we present below. Namely,
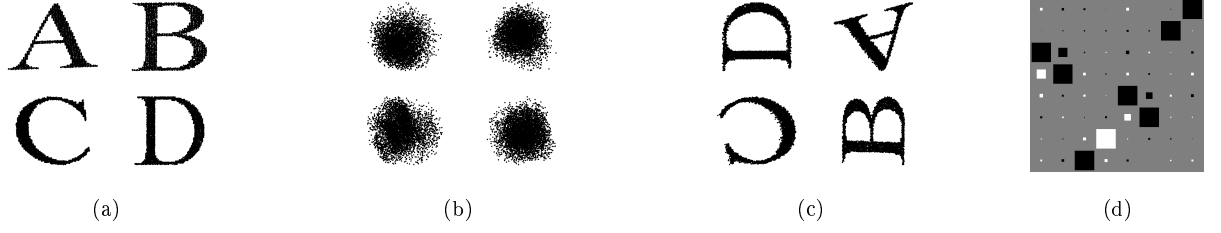
Figure 2: ISA demonstration (`demo_ISA.m`). (a): hidden components ($\{\mathbf{e}^m\}_{m=1}^M$). (b): observed, mixed signal ($\mathbf{x}$). (c): estimated components ($\{\hat{\mathbf{e}}^m\}_{m=1}^M$). (d): Hinton-diagram: the product of the mixing matrix and the estimated demixing matrix; approximately block-permutation matrix with 2×2 blocks.

assume that we have a weight matrix $\mathbf{V} \in \mathbb{R}^{M \times M}$ made of positive matrix elements, and a $q \geq 1$ real number. Loosely speaking, we shrink the $d_i \times d_j$ blocks of matrix $\mathbf{G}$ according to the weights of matrix $\mathbf{V}$ and apply the traditional Amari-index for the matrix we obtain. Formally, one can (i) assume without loss of generality that the component dimensions and their estimations are ordered in increasing order ($d_1 \leq \ldots \leq d_M$, $\hat{d}_1 \leq \ldots \leq \hat{d}_M$), (ii) decompose $\mathbf{G}$ into $d_i \times d_j$ blocks ($\mathbf{G} = \left[\mathbf{G}^{ij}\right]_{i,j=1,\ldots,M}$) and define g^{ij} as the ℓ_q norm[17] of the elements of the matrix $\mathbf{G}^{ij} \in \mathbb{R}^{d_i \times d_j}$, weighted with V_{ij}:

$$g^{ij} = V_{ij} \left(\sum_{k=1}^{d_i} \sum_{l=1}^{d_j} | \left(\mathbf{G}^{ij}\right)_{k,l} |^q \right)^{\frac{1}{q}} . \tag{60}$$

Then the Amari-index with parameters $\mathbf{V}$ can be adapted to the ISA task of possibly different component dimensions as follows

$$r_{\mathbf{V},q}(\mathbf{G}) := \frac{1}{2M(M-1)} \left[\sum_{i=1}^{M} \left(\frac{\sum_{j=1}^{M} g^{ij}}{\max_j g^{ij}} - 1 \right) + \sum_{j=1}^{M} \left(\frac{\sum_{i=1}^{M} g^{ij}}{\max_i g^{ij}} - 1 \right) \right] . \tag{61}$$

One can see that $0 \leq r_{\mathbf{V},q}(\mathbf{G}) \leq 1$ for any matrix $\mathbf{G}$, and $r_{\mathbf{V},q}(\mathbf{G}) = 0$ if and only if $\mathbf{G}$ is block-permutation matrix with $d_i \times d_j$ sized blocks. $r_{\mathbf{V},q}(\mathbf{G}) = 1$ is in the worst case, i.e, when all the g^{ij} elements are equal. Let us note that this measure (61) is invariant, e.g., for multiplication with a positive constant: $r_{c\mathbf{V}} = r_{\mathbf{V}} (\forall c > 0)$. Weight matrix $\mathbf{V}$ can be uniform ($V_{ij} = 1$), or one can use weighing according to the size of the subspaces: $V_{ij} = 1/(d_i d_j)$. The Amari-index [Eq. (61)] is available in the ITE package, see `Amari_index_ISA.m`. The $\mathbf{G}$ global matrix can be visualized by its Hinton-diagram (`hinton_diagram.m`), Fig. 2 provides an illustration. This illustration has been obtained by running `demo_ISA.m`.

The Amari-index can also be used to measure the efficiency of the estimators of the IPA problem family detailed in Section 4.1.2. The demo files in the ITE toolbox (see Table 14) contain detailed examples for the usage of the Amari-index in the extensions of ISA.

4.4 Dataset-, Model Generators

One can generate observations from the ISA model and its extensions (Section 4.1.2) by the functions listed in Table 18. The sources/driving datasets can be chosen from many different types in ITE (see `sample_subspaces.m`):

3D-geom: In the *3D-geom* test [53] $\mathbf{e}^m$s are random variables uniformly distributed on 3-dimensional geometric forms ($d_m = 3$, $M \leq 6$), see Fig. 3(a). The dataset generator is `sample_subspaces_3D_geom.m`.

Aw, ABC, GreekABC: In the $A\omega$ database [79] the distribution of the hidden sources $\mathbf{e}^m$ are uniform on 2-dimensional images ($d_m = 2$) of the English ($M_1 = 26$) and Greek alphabet ($M_2 = 24$). The number of components can be $M = M_1 + M_2 = 50$. Special cases of the database are the *ABC* ($M \leq 26$) [52] and the *GreekABC* ($M \leq 24$) [79] subsets. For illustration, see Fig. 3(d). The dataset generators are called `sample_subspaces_Aw.m`, `sample_subspaces_ABC.m` and `sample_subspaces_GreekABC.m`, respectively.

mosaic: The *mosaic* test [76] has 2-dimensional source components ($d_m = 2$) generated from mosaic images. Sources $\mathbf{e}^m$ are generated by sampling 2-dimensional coordinates proportional to the corresponding pixel intensities. In

---

[17]Alternative norms could also be used.

32

other words, 2-dimensional images are considered as density functions. For illustration, see Fig. 3(h). The dataset generator is `sample_subspaces_mosaic.m`.

IFS: Here [78], components $\mathbf{s}^m$ are realizations of IFS[18] based 2-dimensional ($d = 2$) self-similar structures. For all m a $(\{\mathbf{h}_k\}_{k=1,\dots,K}, \mathbf{p} = (p_1,\dots,p_K), \mathbf{v}_1)$ triple is chosen, where

- $\mathbf{h}_k : \mathbb{R}^2 \to \mathbb{R}^2$ are affine transformations: $\mathbf{h}_k(\mathbf{z}) = \mathbf{C}_k \mathbf{z} + \mathbf{d}_k$ ($\mathbf{C}_k \in \mathbb{R}^{2 \times 2}, \mathbf{d}_k \in \mathbb{R}^2$),
- $\mathbf{p}$ is a distribution over the indices $\{1,\dots,K\}$ ($\sum_{k=1}^K p_k = 1, p_k \geq 0$), and
- for the initial value we chose $\mathbf{v}_1 := \left(\frac{1}{2}, \frac{1}{2}\right)$.

In the *IFS* dataset, T samples are generated in the following way: (i) $\mathbf{v}_1$ is given ($t = 1$), (ii) an index $k(t) \in \{1,\dots,K\}$ is drawn according to the distribution $\mathbf{p}$ and (iii) the next sample is generated as $\mathbf{v}_{t+1} := \mathbf{h}_{k(t)}(\mathbf{v}_t)$. The resulting series $\{\mathbf{v}_1,\dots,\mathbf{v}_T\}$ was taken as a hidden source component $\mathbf{s}^m$ and this way 9 components ($M = 9$, $D = 18$) were constructed (see Fig. 3(c)). The generator of the dataset is `sample_subspaces_IFS.m`.

ikeda: In the *ikeda* test [72], the hidden $\mathbf{s}_t^m = [s_{t,1}^m, s_{t,2}^m] \in \mathbb{R}^2$ sources realize the ikeda map

$$s_{t+1,1}^m = 1 + \lambda_m [s_{t,1}^m \cos(w_t^m) - s_{t,2}^m \sin(w_t^m)], \tag{62}$$

$$s_{t+1,2}^m = \lambda_m [s_{t,1}^m \sin(w_t^m) + s_{t,2}^m \cos(w_t^m)], \tag{63}$$

where λ_m is a parameter of the dynamical system and

$$w_t^m = 0.4 - \frac{6}{1 + (s_{t,1}^m)^2 + (s_{t,2}^m)^2}. \tag{64}$$

There are 2 components ($M = 2$) with initial points $\mathbf{s}_1^1 = [20; 20]$, $\mathbf{s}_1^2 = [-100; 30]$ and parameters $\lambda_1 = 0.9994$, $\lambda_2 = 0.998$, see Fig. 3(f) for illustration. Observation can be generated from this dataset using `sample_subspaces_ikeda.m`.

lorenz: In the *lorenz* dataset [76], the sources ($\mathbf{s}^m$) correspond to 3-dimensional ($d_m = 3$) deterministic chaotic time series, the so-called Lorenz attractor [41] with different initial points (x_0, y_0, z_0) and parameters (a, b, c). The Lorenz attractor is described by the following ordinary differential equations:

$$\dot{x}_t = a(y_t - x_t), \tag{65}$$

$$\dot{y}_t = x_t(b - z_t) - y_t, \tag{66}$$

$$\dot{z}_t = x_t y_t - c z_t. \tag{67}$$

The differential equations are computed by the explicit Runge-Kutta (4,5) method in ITE. The number of components can be $M = 3$. The dataset generator is `sample_subspaces_lorenz.m`. For illustration, see Fig. 3(g).

all-k-independent: In the *all-k-independent* database [52, 73], the d_m-dimensional hidden components $\mathbf{v} := \mathbf{e}^m$ are created as follows: coordinates v_i ($i = 1,\dots,k$) are independent uniform random variables on the set $\{0,\dots,\text{k-1}\}$, whereas v_{k+1} is set to $mod(v_1 + \dots + v_k, k)$. In this construction, every k-element subset of $\{v_1,\dots,v_{k+1}\}$ is made of independent variables and $d_m = k + 1$. The database generator is `sample_subspaces_all_k_independent.m`.

multiD-geom (multiD$_1$-...-D$_M$-geom): In this dataset $\mathbf{e}^m$s are random variables uniformly distributed on d_m-dimensional geometric forms. Geometrical forms were chosen as follows: (i) the surface of the unit ball, (ii) the straight lines that connect the opposing corners of the unit cube, (iii) the broken line between $d_m + 1$ points $\mathbf{0} \to \mathbf{e}_1 \to \mathbf{e}_1 + \mathbf{e}_2 \to \dots \to \mathbf{e}_1 + \dots + \mathbf{e}_{d_m}$ (where $\mathbf{e}_i$ is the i canonical basis vector in $\mathbb{R}^{d_m}$, i.e., all of its coordinates are zero except the ith, which is 1), and (iv) the skeleton of the unit square. Thus, the number of components M can be equal to 4 ($M \leq 4$), and the dimension of the components (d_m) can be scaled. In the *multiD-geom* case the dimensions of the subspaces are equal ($d_1 = \dots = d_M$); in case of the *multiD$_1$-...-D$_M$-geom* dataset, the d_m subspace dimensions can be different. For illustration, see Fig. 3(e). The associated dataset generator is called `sample_subspaces_multiD_geom.m`.
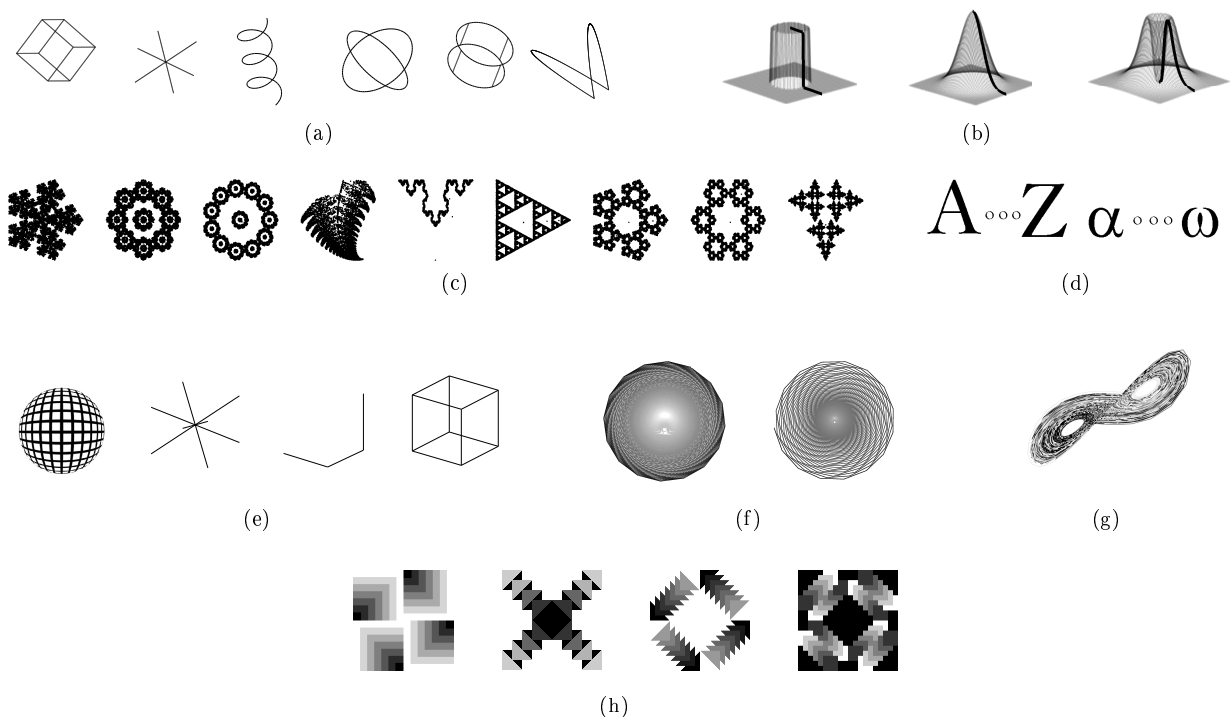
---

[18]IFS stands for iterated function system.

33

Figure 3: Illustration of the *3D-geom* (a), *multiD-spherical (multiD$_1$-...-D$_M$-spherical)* (b), *IFS* (c), *Aw* (subset on the left: *ABC*, right: *GreekABC*) (d), *multiD-geom (multiD$_1$-...-D$_M$-geom)* (e), *ikeda* (f), *lorenz* (g), and *mosaic* (h) datasets.

multiD-spherical (multiD$_1$-...-D$_M$-spherical): In this case hidden sources $\mathbf{e}^m$ are spherical random variables [15]. Since spherical variables assume the form $\mathbf{v} = \rho\mathbf{u}$, where $\mathbf{u}$ is uniformly distributed on the d_m-dimensional unit sphere, and ρ is a non-negative scalar random variable independent of $\mathbf{u}$, they can be given by means of ρ. 3 pieces of stochatistic representations ρ were chosen: ρ was uniform on $[0,1]$, exponential with parameter $\mu = 1$ and lognormal with parameters $\mu = 0$, $\sigma = 1$. For illustration, see Fig. 3(b). In this case, the number of component can be 3 ($M \leq 3$) The dimension of the source components (d_m) is fixed (can be varied) in the *multiD-spherical (multiD$_1$-...-D$_M$-spherical)* dataset. Observations can be obtained from these datasets by `sample_subspaces_multiD_spherical.m`.

The datasets and their generators are summarized in Table 19 and Table 20. The `plot_subspaces.m` function can be used to plot the databases (samples/estimations).

Model	Generator
ISA	`generate_ISA.m`
complex ISA	`generate_complex_ISA.m`
AR-IPA	`generate_AR_IPA.m`
ARX-IPA	`generate_ARX_IPA_parameters.m`
(u)MA-IPA	`generate_MA_IPA.m`
mAR-IPA	`generate_mAR_IPA.m`
fAR-IPA	`generate_fAR_IPA.m.m`

Table 18: IPA model generators. Note: in case of the ARX-IPA model, the observations are generated online in accordance with the online D-optimal ARX identification method.

Dataset (`data_type`)	Description	Subspace dimensions	# of components	i.i.d.
'3D-geom'	uniformly distributed (U) on 3D forms	$d_m = 3$	$M \le 6$	Y
'Aw'	U on English and Greek letters	$d_m = 2$	$M \le 50$	Y
'ABC'	U on English letters	$d_m = 2$	$M \le 26$	Y
'GreekABC'	U on Greek letters	$d_m = 2$	$M \le 24$	Y
'mosaic'	distributed according to mosaic images	$d_m = 2$	$M \le 4$	Y
'IFS'	self-similar construction	$d_m = 2$	$M \le 9$	N
'ikeda'	Ikeda map	$d_m = 2$	$M = 2$	N
'lorenz'	Lorenz attractor	$d_m = 3$	$M \le 3$	N
'all-k-independent'	k-tuples in the subspaces are independent	scalable ($d_m = k + 1$)	$M \ge 1$	Y
'multid-geom'	U on d-dimensional geometrical forms	scalable ($d = d_m \ge 1$)	$M \le 4$	Y
'multid$_1$-d$_2$-...-d$_M$-geom'	U on d_m-dimensional geometrical forms	scalable ($d_m \ge 1$)	$M \le 4$	Y
'multid-spherical'	spherical subspaces	scalable ($d = d_m \ge 1$)	$M \le 3$	Y
'multid$_1$-d$_2$-...-d$_M$-spherical'	spherical subspaces	scalable ($d_m \ge 1$)	$M \le 3$	Y

Table 19: Description of the datasets. Last column: Y–yes, N–no.

Dataset (`data_type`)	Generator
'3D-geom'	`sample_subspaces_3D_geom.m`
'Aw'	`sample_subspaces_Aw.m`
'ABC'	`sample_subspaces_ABC.m`
'GreekABC'	`sample_subspaces_GreekABC.m`
'mosaic'	`sample_subspaces_mosaic.m`
'IFS'	`sample_subspaces_IFS.m`
'ikeda'	`sample_subspaces_ikeda.m`
'lorenz'	`sample_subspaces_lorenz.m`
'all-k-independent'	`sample_subspaces_all_k_independent.m`
'multid-geom', 'multid$_1$-d$_2$-...-d$_M$-geom'	`sample_subspaces_multiD_geom.m`
'multid-spherical', 'multid$_1$-d$_2$-...-d$_M$-spherical'	`sample_subspaces_multiD_spherical.m`

Table 20: Generators of the datasets. The high-level sampling function of the datasets is `sample_subspaces.m`.

5 Directory Structure of the Package

In this section, we describe the directory structure of the ITE toolbox. Directory

- *code*: code of ITE,
  - *H_I_D*: entropy-, mutual information-, divergence estimators (see Section 3).
    * *base*: contains the base estimators; initialization and estimation functions (see Section 3.1).
    * *meta*: the folder of meta estimators; initialization and estimation functions (see Section 3.2).
    * *utilities*: code shared by *base* and *meta*.
  - *IPA*: application of the information theoretical estimators in ITE (see Section 4):
    * *data_generation*: IPA generators corresponding to different datasets and models.
      · *datasets*: sampling from and plotting of the sources (see Table 19, Table 20, Fig. 3).
      · *models*: IPA model generators, see Table 18.
    * *demos*: IPA demonstrations and estimators, see Table 14 and Table 15.
    * *optimization*: IPA optimization methods (see Table 9, Table 10, Table 11, Table 12, and Table 13).
  - *shared*: code shared by *H_I_D* and *IPA*.
    * *downloaded, embedded*: downloaded and embedded packages (see Section 2).
- *doc*: documentation of the ITE toolbox; contains the current manual.

A Citation of the ITE Toolbox

The citing information of the ITE toolbox is provided below in BibTeX format:

```
@ARTICLE{szabo12separation,
  AUTHOR =        {Zolt{\'a}n Szab{\'o} and Barnab{\'a}s P{\'o}czos and Andr{\'a}s L{\H{o}}rincz},
  TITLE =         {Separation Theorem for Independent Subspace Analysis and its Consequences},
  JOURNAL =       {Pattern Recognition},
  YEAR =          {2012},
  volume =        {45},
  issue =         {4},
  pages =         {1782-1791},
}


@ARTICLE{szabo07undercomplete,
  AUTHOR =        {Zolt{\'a}n Szab{\'o} and Barnab{\'a}s P{\'o}czos and Andr{\'a}s L{\H{o}}rincz},
  TITLE =         {Undercomplete Blind Subspace Deconvolution},
  JOURNAL =       {Journal of Machine Learning Research},
  YEAR =          {2007},
  volume =        {8},
  pages =         {1063-1095},
}
```

B Abbreviations

The abbreviations used in the paper are listed in Table 21.

C Functions with Octave-Specific Adaptations

Functions with Octave-specific adaptations are summarized in Table 22.

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66:671–687, 2003.

[2] Shun-ichi Amari, Andrzej Cichocki, and Howard H. Yang. A new learning algorithm for blind signal separation. *Neural Information Processing Systems (NIPS)*, pages 757–763, 1996.

[3] Rosa I. Arriga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projections. *Machine Learning*, 63:161–182, 2006.

[4] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[5] Michéle. Basseville. Divergence measures for statistical data processing - an annotated bibliography. *Signal Processing*, 2012. To appear. `hal.inria.fr/docs/00/54/23/37/PDF/PI-1961.pdf`.

[6] J. Beirlant, E.J. Dudewicz, L. Győrfi, and E.C. van der Meulen. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 6:17–39, 1997.

[7] Ella Bingham and Aapo Hyvärinen. A fast fixed-point algorithm for independent component analysis of complex-valued signals. *International Journal of Neural Systems*, 10(1):1–8, 2000.

[8] Jean-François Cardoso. Multidimensional independent component analysis. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1941–1944, 1998.

Abbreviation	Meaning
ANN	approximate nearest neighbor
AR	autoregressive
ARIMA	integrated ARMA
ARMA	autoregressive moving average
ARX	AR with exogenous input
BFGS	Broyden-Fletcher-Goldfarb-Shannon
BSD	blind source deconvolution
BSSD	blind subspace deconvolution
CE	cross-entropy
fAR	functional AR
GV	generalized variance
HSIC	Hilbert-Schmidt independence criterion
ICA/ISA/IPA	independent component/subspace/process analysis
i.i.d.	independent identically distributed
IFS	iterated function system
ITE	information theoretical estimators
JFD	joint f-decorrelation
KCCA	kernel canonical correlation analysis
KGV	kernel generalized variance
kNN	k-nearest neighbor
LPA	linear prediction approximation
MA	moving average
mAR	AR with missing values
MMD	maximum mean discrepancy
NIW	normal-inverted Wishart
NN	nearest neighbor
PCA	principal component analysis
PNL	post nonlinear
RKHS	reproducing kernel Hilbert space
RP	random projection

Table 21: Abbrevations.

Function	Role
`ITE_install.m`	installation of the ITE package
`hinton_diagram.m`	Hinton-diagram
`estimate_clustering_UD1_S.m`	spectral clustering
`control.m`	D-optimal control
`sample_subspace_lorenz.m`	sampling from the *lorenz* dataset
`clinep.m`	the core of the 3D trajectory plot
`plot_subspaces_3D_trajectory.m`	3D trajectory plot
`IGV_similarity_matrix.m`	similarity matrix for the GV measure
`calculateweight.m`	weight computation in the weighted kNN method
`kNN_squared_distances.m`	kNN computation
`initialize_Octave_ann_wrapper_if_needed.m`	ann Octave wrapper initialization

Table 22: Functions with Octave-specific adaptations.

[9] Jean-François Cardoso and Antoine Souloumiac. Blind beamforming for non-gaussian signals. *IEE Proceedings F, Radar and Signal Processing*, 140(6):362–370, 1993.

[10] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36:287–314, 1994.

[11] Timothee Cour, Stella Yu, and Jianbo Shi. Normalized cut segmentation code. Copyright 2004 University of Pennsylvania, Computer and Information Science Department.

[12] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, USA, 1991.

[13] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. Fetal electrocardiogram extraction by source subspace separation. In *IEEE SP/Athos Workshop on Higher-Order Statistics*, pages 134–138, 1995.

[14] Jan Eriksson. Complex random vectors and ICA models: Identifiability, uniqueness and separability. *IEEE Transactions on Information Theory*, 52(3), 2006.

[15] Kai-Tai Fang, Samuel Kotz, and Kai Wang Ng. *Symmetric multivariate and related distributions*. Chapman and Hall, 1990.

[16] Peter Frankl and Hiroshi Maehara. The Johnson-Lindenstrauss Lemma and the sphericity of some graphs. *Journal of Combinatorial Theory Series A*, 44(3):355 – 362, 1987.

[17] Wayne A. Fuller. *Introduction to Statistical Time Series*. Wiley-Interscience, 1995.

[18] Sandra Gaißer, Martin Ruppert, and Friedrich Schmid. A multivariate version of Hoeffding's phi-square. *Journal of Multivariate Analysis*, 101:2571–2586, 2010.

[19] M. N. Goria, Nikolai N. Leonenko, V. V. Mergel, and P. L. Novi Inverardi. A new class of random vector entropy estimators and its applications in testing statistical hypotheses. *Journal of Nonparametric Statistics*, 17:277–297, 2005.

[20] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

[21] Arthur Gretton, Olivier Bousquet, Alexander Smola, and Bernhard Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *International Conference on Algorithmic Learnng Theory (ALT)*, pages 63–78, 2005.

[22] Godfrey H. Hardy and Srinivasa I. Ramanujan. Asymptotic formulae in combinatory analysis. *Proceedings of the London Mathematicl Society*, 17(1):75–115, 1918.

[23] Nadine Hilgert and Bruno Portier. Strong uniform consistency and asymptotic normality of a kernel based error density estimator in functional autoregressive models. *Statistical Inference for Stochastic Processes*, 15(2):105–125, 2012.

[24] W. Hoeffding. Massstabinvariante korrelationstheorie. *Schriften des Mathematischen Seminars und des Instituts für Angewandte Mathematik der Universität Berlin*, 5:181–233, 1940.

[25] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[26] Aapo Hyvärinen. Independent component analysis for time-dependent stochastic processes. In *International Conference on Artificial Neural Networks (ICANN)*, pages 541–546, 1998.

[27] Aapo Hyvärinen and Erkki Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

[28] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing, 1998*, pages 604–613.

[29] Miguel Jerez, Jose Casals, and Sonia Sotoca. *Signal Extraction for Linear State-Space Models: Including a free MATLAB Toolbox for Time Series Modeling and Decomposition*. LAP LAMBERT Academic Publishing, 2011.

[30] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[31] Christian Jutten and Jeanny Hérault. Blind separation of sources: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

[32] Christian Jutten and Juha Karhunen. Advances in blind source separation (BSS) and independent component analysis (ICA) for nonlinear systems. *International Journal of Neural Systems*, 14(5):267–292, 2004.

[33] K. Rao Kadiyala and Sune Karlsson. Numerical methods for estimation and inference in bayesian VAR-models. *Journal of Applied Econometrics*, 12:99–132, 1997.

[34] Sergey Kirshner and Barnabás Póczos. ICA and ISA using Schweizer-Wolff measure of dependence. In *International Conference on Machine Learning (ICML)*, pages 464–471, 2008.

[35] L. F. Kozachenko and Nikolai N. Leonenko. A statistical estimate for the entropy of a random vector. *Problems of Information Transmission*, 23:9–16, 1987.

[36] Solomon Kullback and Richard Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[37] Jan Kybic. High-dimensional mutual information estimation for image registration. In *International Conference on Image Processing (ICIP)*, pages 1779–1782, 2004.

[38] Russell H. Lambert. *Multichannel Blind Deconvolution: FIR matrix algebra and separation of multipath mixtures*. PhD thesis, University of Southern California, 1996.

[39] Nikolai Leonenko, Luc Pronzato, and Vippal Savani. A class of Rényi information estimators for multidimensional densities. *Annals of Statistics*, 36(5):2153–2182, 2008.

[40] Ping Li, Trevor J. Hastie, and Kenneth W. Hastie. Very sparse random projections. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 287–296, 2006.

[41] Edward Norton Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20:130–141, 1963.

[42] Jiří Matoušek. On variants of the Johnson-Lindenstrauss lemma. *Random Structures and Algorithms*, 33(2):142–156, 2008.

[43] Roger B. Nelsen. *An Introduction to Copulas (Springer Series in Statistics)*. Springer, 2006.

[44] Arnold Neumaier and Tapio Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software*, 27(1):27–57, 2001.

[45] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 849–856, 2002.

[46] Michael S. Pedersen, Jan Larsen, Ulrik Kjems, and Lucas C. Parra. A survey of convolutive blind source separation methods. In *Springer Handbook of Speech Processing*. Springer, 2007.

[47] Barnabás Póczos, Zoubin Ghahramani, and Jeff Schneider. Copula-based kernel dependency measures. In *International Conference on Machine Learning (ICML)*, 2012.

[48] Barnabás Póczos and András Lőrincz. Identification of recurrent neural networks by Bayesian interrogation techniques. *Journal of Machine Learning Research*, 10:515–554, 2009.

[49] Barnabás Póczos, Zoltán Szabó, Melinda Kiszlinger, and András Lőrincz. Independent process analysis without a priori dimensional information. In *International Conference on Independent Component Analysis and Signal Separation (ICA)*, pages 252–259, 2007.

[50] Barnabás Póczos, Bálint Takács, and András Lőrincz. Independent subspace analysis on innovations. In *European Conference on Machine Learning (ECML)*, pages 698–706, 2005.

[51] Dávid Pál, Barnabás Póczos, and Csaba Szepesvári. Estimation of Rényi entropy and mutual information based on generalized nearest-neighbor graphs. In *Neural Information Processing Systems (NIPS)*, pages 1849–1857, 2011.

[52] Barnabás Póczos and András Lőrincz. Independent subspace analysis using geodesic spanning trees. In *International Conference on Machine Learning (ICML)*, pages 673–680, 2005.

[53] Barnabás Póczos and András Lőrincz. Independent subspace analysis using k-nearest neighborhood estimates. In *International Conference on Artificial Neural Networks (ICANN)*, pages 163–168, 2005.

[54] Barnabás Póczos and Jeff Schneider. On the estimation of α-divergences. In *International conference on Artificial Intelligence and Statistics (AISTATS)*, pages 609–617, 2011.

[55] Barnabás Póczos, Zoltán Szabó, and Jeff Schneider. Nonparametric divergence estimators for independent subspace analysis. In *European Signal Processing Conference (EUSIPCO)*, pages 1849–1853, 2011.

[56] Barnabás Póczos, Liang Xiong, and Jeff Schneider. Nonparametric divergence: Estimation with applications to machine learning on distributions. In *Uncertainty in Artificial Intelligence (UAI)*, pages 599–608, 2011.

[57] Ravikiran Rajagopal and Lee C. Potter. Multivariate MIMO FIR inverses. *IEEE Transactions on Image Processing*, 12:458–465, 2003.

[58] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method*. Springer, 2004.

[59] Alfréd Rényi. On measures of information and entropy. In *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*, pages 547–561, 1961.

[60] Tapio Schneider and Arnold Neumaier. Algorithm 808: ARfit - a Matlab package for the estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software*, 27(1):58–65, 2001.

[61] B. Schweizer and E. F. Wolff. On nonparametric measures of dependence for random variables. *The Annals of Statistics*, 9:879–885, 1981.

[62] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[63] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[64] Harshinder Singh, Neeraj Misra, Vladimir Hnizdo, Adam Fedorowicz, and Eugene Demchuk. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 23:301–321, 2003.

[65] Kumar Sricharan and Alfred. O. Hero. Weighted k-NN graphs for Rényi entropy estimation in high dimensions. In *IEEE Workshop on Statistical Signal Processing (SSP)*, pages 773–776, 2011.

[66] Zoltán Szabó. Complete blind subspace deconvolution. In *International Conference on Independent Component Analysis and Signal Separation (ICA)*, pages 138–145, 2009.

[67] Zoltán Szabó. Autoregressive independent process analysis with missing observations. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 159–164, 2010.

[68] Zoltán Szabó and András Lőrincz. Towards independent subspace analysis in controlled dynamical systems. In *ICA Research Network International Workshop (ICARN)*, pages 9–12, 2008.

[69] Zoltán Szabó and András Lőrincz. Complex independent process analysis. *Acta Cybernetica*, 19:177–190, 2009.

[70] Zoltán Szabó and András Lőrincz. Fast parallel estimation of high dimensional information theoretical quantities with low dimensional random projection ensembles. In *International Conference on Independent Component Analysis and Signal Separation (ICA)*, pages 146–153, 2009.

[71] Zoltán Szabó and András Lőrincz. Distributed high dimensional information theoretical image registration via random projections. *Digital Signal Processing*, 22(6):894–902, 2012.

[72] Zoltán Szabó and Barnabás Póczos. Nonparametric independent process analysis. In *European Signal Processing Conference (EUSIPCO)*, pages 1718–1722, 2011.

[73] Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Cross-entropy optimization for independent process analysis. In *International Conference on Independent Component Analysis and Blind Source Separation (ICA)*, pages 909–916, 2006.

[74] Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Undercomplete blind subspace deconvolution. *Journal of Machine Learning Research*, 8:1063–1095, 2007.

[75] Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Undercomplete blind subspace deconvolution via linear prediction. In *European Conference on Machine Learning (ECML)*, pages 740–747, 2007.

[76] Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Auto-regressive independent process analysis without combinatorial efforts. *Pattern Analysis and Applications*, 13:1–13, 2010.

[77] Zoltán Szabó, Barnabás Póczos, and András Lőrincz. Separation theorem for independent subspace analysis and its consequences. *Pattern Recognition*, 45:1782–1791, 2012.

[78] Zoltán Szabó, Barnabás Póczos, Gábor Szirtes, and András Lőrincz. Post nonlinear independent subspace analysis. In *International Conference on Artificial Neural Networks (ICANN)*, pages 677–686, 2007.

[79] Zoltán Szabó and András Lőrincz. Real and complex independent subspace analysis by generalized variance. In *ICA Research Network International Workshop (ICARN)*, pages 85–88, 2006.

[80] Anisse Taleb and Christian Jutten. Source separation in post-nonlinear mixtures. *IEEE Transactions on Signal Processing*, 10(47):2807–2820, 1999.

[81] Fabian J. Theis. Blind signal separation into groups of dependent signals using joint block diagonalization. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 5878–5881, 2005.

[82] Fabian J. Theis. Towards a general independent subspace analysis. In *Neural Information Processing Systems (NIPS)*, pages 1361–1368, 2007.

[83] James V. Uspensky. Asymptotic formulae for numerical functions which occur in the theory of partitions. *Bulletin of the Russian Academy of Sciences*, 14(6):199–218, 1920.

[84] T. Villmann and S. Haase. Mathematical aspects of divergence based vector quantization using Fréchet-derivatives. Technical report, University of Applied Sciences Mittweida, 2010.

[85] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007.

[86] Quing Wang, Sanjeev R. Kulkarni, and Sergio Verdú. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55:2392–2405, 2009.

[87] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 907–916, 2009.

[88] Joseph E. Yukich. Probability theory of classical Euclidean optimization problems. *Lecture Notes in Mathematics*, 1675, 1998.

[89] Andreas Ziehe, Motoaki Kawanabe, Stefan Harmeling, and Klaus-Robert Müller. Blind separation of postnonlinear mixtures using linearizing transformations and temporal decorrelation. *Journal of Machine Learning Research*, 4:1319–1338, 2003.