minerva and minepy: a C engine for the MINE suite and its R, Python and MATLAB wrappers

Davide Albanese1,+, Michele Filosi1,2,+, Roberto Visintainer1,3,+, Samantha Riccadonna1, Giuseppe Jurman1 and Cesare Furlanello1*

1Fondazione Bruno Kessler, via Sommarive 18, I-38123 Povo (Trento), Italy
2CIBIO, University of Trento, via delle Regole 101, I-38123 Mattarello (Trento), Italy
3DISI, University of Trento, Via Sommarive 5, I-38123 Povo (Trento), Italy

Associate Editor: Dr. Jonathan Wren

ABSTRACT
Summary: We introduce a novel implementation in ANSI C of the MINE family of algorithms for computing maximal information-based measures of dependence between two variables in large datasets, with the aim of a low memory footprint and ease of integration within bioinformatics pipelines. We provide the libraries minerva (with the R interface) and minepy for Python, MATLAB, Octave and C++. The C implementation has good upscaling properties, and offers a native parallelization for the R interface. Low memory requirements are demonstrated on the MINE benchmarks as well as on large (n=1340) microarray and Illumina GAII RNA-seq transcriptomics datasets.

Availability and Implementation: Source code and binaries are freely available for download under GPL3 licence at http://minepy.sourceforge.net for minepy and through the CRAN repository http://cran.r-project.org for the R package minerva. All software is multiplatform (MS Windows, Linux and OSX).

Contact: furlan@fbk.eu

Supplementary information: Supplementary information is available at the website http://mpba.fbk.eu/minepy-minerva

1 INTRODUCTION
The Maximal Information-based Nonparametric Exploration (MINE) family of statistics, including the Maximal Information Coefficient (MIC) measure, was recently introduced in (Reshef et al., 2011), aimed at fast exploration of two-variable relationships in many-dimensional data sets. MINE consists of the algorithms for computing four measures of dependence — MIC, Maximum Asymmetry Score (MAS), Maximum Edge Value (MEV), Minimum Cell Number (MCN) — between two variables, having the generality and equitability property. Generality is the ability of capturing variable relationships of different nature, while equitability is the property of penalizing similar levels of noise in the same way, regardless of the nature of the relation between the variables. The MINE suite received immediate appraisal as a real breakthrough in the data mining of complex biological data (Speed, 2011; Nat. Biotech., 2012) as well as criticisms1. Many groups worldwide have already proposed its use for explorative data analysis in computational biology, from networks dynamics to virus ranking (Weiss et al., 2012; Das et al., 2012; Anderson et al., 2012; Karpinets et al., 2012; Faust and Raes, 2012). Together with the algorithm description, the MINE authors provided a Java implementation (MINE.jar), two wrappers (R and Python), and four reference datasets (Reshef et al., 2011). However, applicability and scalability of MINE.jar on large datasets is currently limited due to memory requirements and lack of programming interfaces. Further, a native parallelization, currently unavailable, would be of significant benefit. These issues are hurdles for a systematic application of MINE algorithms to high-throughput omics data — for example, as a substitute of Pearson correlation in network studies. Inspired by these considerations, we propose an ANSI C implementation of the MINE algorithms, and the interfaces for R (minerva), and for C++, Python and MATLAB/Octave.

2 THE MINE C ENGINE AND ITS WRAPPERS
The novel engine (libmine) is written in ANSI C as a clean-room implementation of the algorithms originally described in (Reshef et al., 2011), as the Java source code is not distributed. Libmine provides three structures describing the data, the parameter configuration and the maximum normalized mutual information scores. The core function mine_compute_score() takes a dataset structure and a configuration one as input, returning a score structure as output, from which four functions compute the MINE statistics. The minepy Python module works with Python ≥ 2.6, with NumPy ≥ 1.3.0 as the sole requirement: the interface consists of the class minepy.MINE whose methods match the C functions. The R package minerva is built as an R wrapper (R and Python), and for C++, Python and MATLAB/Octave.

*to whom correspondence should be addressed
+ authors equally contributed to this work

1 See comments and referenced experiments by Simon and Tibshirani and by Gorfin et al at http://comments.sciencemag.org/content/10.1126/science.1205438
mine, whenever multi-core hardware is available. The curated version of the CDC15 Spellman yeast dataset (Spellman et al., 1998) used in (Reshef et al., 2011) is included as example. Documentation (on-line and PDF) for minepy is available at the minepy website, also as on-line help in R for minerva.

Performance comparison The suite was tested for consistency with MINE.jar v1.0.1 on the Spellman and microbiome datasets from http://www.exploredata.net. For the Spellman dataset (4381 transcripts and 23 timepoints), MIC values were computed on all features pairs with MINE.jar and minepy (both with \( \alpha = 0.67 \)) requiring much higher computing time as sample size increases (Fig. S7). We additionally tested the suite on two recent high-throughput transcriptomics datasets, of Affymetrix HumanExon 1.0ST human brain tissues and Illumina Genome Analyzer II sequenced human non-small cell lung cancer (Tab. 1). Details on datasets and experiments are reported in SI.

Table 1. Performance of minerva and minepy (1-vs.-all) on microarray and RNA-seq datasets listed by GEO accession number. \( n \): number of samples. \( p \): number of features. CPU: Elapsed time used by the process (in seconds). RAM: resident set size (in kilobytes) for minerva (R) and minepy (P).

<table>
<thead>
<tr>
<th>GEO</th>
<th>Acc. no.</th>
<th>( n )</th>
<th>( p )</th>
<th>CPU R</th>
<th>CPU P</th>
<th>RAM R</th>
<th>RAM P</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSE25219</td>
<td>14</td>
<td>1,340</td>
<td>17,565</td>
<td>41,880</td>
<td>34,359</td>
<td>533,008</td>
<td>1,509,692</td>
</tr>
<tr>
<td>GSE34914</td>
<td>2</td>
<td>16</td>
<td>20,422</td>
<td>42</td>
<td>3</td>
<td>35,716</td>
<td>31,956</td>
</tr>
</tbody>
</table>

1 Kang et al. (2011) 2 Kalari et al. (2012)

ACKNOWLEDGEMENTS

Funding: This work was supported by the EU FP7 Project HiPER-DART.

REFERENCES


