X-MIFS: Exact Mutual Information for Feature Selection

Mauro Brunato
DISI — University of Trento, Italy
Email: mauro.brunato@unitn.it

Roberto Battiti
DISI — University of Trento, Italy
Email: roberto.battiti@unitn.it

***Abstract*—In machine learning, an information-theory optimal way to filter the best input features, without reference to any specific machine learning models, consists of maximizing the mutual information between the selected features and the model output, a choice which will minimize the uncertainty in the output to be predicted, given the feature values.**

Although this criterion is optimal in the context of information theory, a practical difficulty in using it lies in the need to estimate the mutual information from a limited set of input-output examples, in possibly very-high-dimensional input spaces. Estimating probability densities from some data points in these conditions is far from trivial.

Starting from the seminal proposals in [1], different approaches focus on approximating the mutual information by considering a limited set of variable dependencies (like dependencies among couples or triplets), or by assuming specific forms for the probability densities (like Gaussian forms). In this paper we study the effect of considering the exact mutual information between selected features and output, without resorting to any approximation (apart from that implicit and unavoidable in estimating it from experimental data). The objectives of this investigation are: to assess how far one can go by adopting the exact mutual information in terms of CPU time and number of features, and to measure what is lost by adopting some popular approximations which consider only relationships among small subsets of features, assumptions about the distribution of feature values (e.g. Gaussian) or upper bounds on the mutual information as proxies to maximize instead of the exact value.

The experimental results show a significant performance advantage when the feature sets identified by exact mutual information are used in both binary and multi-valued classification tasks, with longer CPU times.

I. Introduction

In machine learning (ML) classification or regression problems, the feature selection task consists of determining the most useful features among an initial set which may contain too many or redundant features. Selecting a subset of features can have a positive effect on the performance of machine learning algorithms by reducing the hypothesis search space, and, in some cases, by reducing the storage requirement. Last but not least, a reduced set of features makes it easier to understand and explain the obtained ML systems [2].

Mutual Information (MI) measures arbitrary dependencies between random variables, and it is suitable for assessing the *information content* of features in complex classification tasks. Features filtered according to their MI reduce the output uncertainty and the classification error.

The MI measure does not depend on the particular machine learning model considered or on the choice of coordinates. Because of Fano's inequality, having features with high MI is a necessary condition for building an effective classifier (unfortunately, the condition is not sufficient, because the available information content must be *extracted* by the particular ML model considered).

MI is now widely used for filtering features. Nonetheless, the published papers mention that calculating the mutual information for tasks characterized by high input dimensionality requires suitable approximations because of the prohibitive demands on computation and samples. Starting from the seminal paper[1], greedy heuristic algorithms have been proposed taking both the mutual information with respect to the output class and with respect to the already-selected features into account. The MI between a set of features and the output is approximated by a heuristic function of the MI between couples of variables or, at most, between triplets. Alternatively, upper bounds are obtained and optimized instead of the initial MI.

This paper takes the radical approach to estimate the *exact* MI between sets of inputs and output in the general, non-binary case. With the data structure and algorithms proposed, the MI calculation requires larger CPU times w.r.t. approximated algorithms. Nonetheless, in practice, the MI can be calculated for systems with tens of input features in tens of minutes, pawing the way to innovative algorithms, both to filter features and to extract particularly informative ("deep") features.

The experimental results show that the proposed approach with the suggested data structures is now viable in terms of CPU time, in particular considering that feature selection is done once when building a new ML model, and it leads to a significantly higher accuracy in popular benchmark classification tasks.

This paper is organized as follows. Section II introduces the information-theoretical definitions, puts them in the context of feature selection tasks and motivates their use in the wider area of supervised machine learning. Section III discusses the state of the art in the use of MI to filter and select features, and describes the main techniques our proposal is be compared to. Section IV discusses the problem of computing the exact MI of a sample drawn from a population and describes the proposed X-MIFS method. Section V compares the proposed algorithms

978-1-5090-0620-5/16/$31.00

with the state-of-the-art techniques in terms of achieved MI, classifier performance and CPU time. Finally, Sec. VI draws some conclusions and proposes possible future developments.

II. Definition of Mutual Information

Given a classification task, let C be the set of output classes and denote by $\Pr(c)$ the probabilities for class $c \in C$ to occur. Then the *a priori* uncertainty in the classes is measured by the entropy:

$$\mathrm{H}(C) = -\sum_{c \in C} \Pr(c) \log \Pr(c).$$

Given some input information F, e.g., feature vectors in a dataset, the uncertainty on the class outcome after knowing a specific feature vector $f \in F$ is the conditional entropy:

$$\mathrm{H}(C|F = f) = -\sum_{c \in C} \Pr(c|f) \log \Pr(c|f).$$

Given a probability distribution of the feature vector, $\Pr(f)$, the expected uncertainty on the outcome becomes therefore:

$$\mathrm{H}(C|F) = \sum_{f \in F} \Pr(f) \mathrm{H}(C|F = f).$$

In general, the conditional entropy will be less or equal to the initial entropy and will be equal if and only if the features and the output class are independent random variables. The amount by which the class C entropy decreases by knowledge of input information F is the mutual information:

$$\mathrm{MI}(C; F) = \mathrm{H}(C) - \mathrm{H}(C|F).$$

Mutual Information is symmetric in C and F and it can be shown that

$$\mathrm{MI}(C; F) = \sum_{c \in C, f \in F} \Pr(c, f) \log \frac{\Pr(c, f)}{\Pr(c) \Pr(f)}. \qquad (1)$$

A. Mutual Information for feature selection

In the context of classification tasks, an observation (e.g., an individual in a population) is associated to an $(n+1)$-tuple of random variables $(F_1, \ldots, F_n, C)$; individual random variables $F_1, \ldots, F_n$ are called *features* of the observation; the last one, C, is called the *class*. We will assume that all random variables have finite support (e.g., binary).

Given an integer $1 \leq k < n$, the problem of feature selection can be informally defined as follows:

> *Let $\mathcal{F} = \{F_1, \ldots, F_n\}$ be the collection of all features; find the subset $\mathcal{S} \subseteq \mathcal{F}$ such that $|\mathcal{S}| = k$ and $\mathcal{S}$ is maximally informative about C.*

In the following, we will identify the set of features $\mathcal{F} = \{F_1, \ldots, F_n\}$ with the multi-valued random variable $\mathcal{F} = (F_1, \ldots, F_n)$. In this way, whenever we choose $\mathcal{S} \subseteq \mathcal{F}$, the conditional entropy $\mathrm{H}(C|\mathcal{S})$ and the mutual information $\mathrm{MI}(\mathcal{S}; C)$ are well defined.

Following [1], by using the entropy as measure of the statistical uncertainty in the output to predict, and the mutual information as a measure of information content in a set of features, the above task above can be defined as the following *MIFS principle* (Mutual Information for Feature Selection):

> *Let $\mathcal{F} = \{F_1, \ldots, F_n\}$ be the collection of all features; find the subset $\mathcal{S} \subseteq \mathcal{F}$ such that $|\mathcal{S}| = k$ and $\mathcal{S}$ minimizes $\mathrm{H}(C|\mathcal{S})$ or, equivalently, maximizes $\mathrm{MI}(\mathcal{S}; C)$:*

$$\mathcal{S} = \operatorname*{arg\,min}_{\mathcal{F}' \subseteq \mathcal{F}, |\mathcal{F}'| = k} \mathrm{H}(C|\mathcal{F}') = \operatorname*{arg\,max}_{\mathcal{F}' \subseteq \mathcal{F}, |\mathcal{F}'| = k} \mathrm{MI}(\mathcal{F}'; C).$$

B. Application to supervised machine learning

The idea that mutual information is related to the accuracy of a learning system, also called the *Infomax Principle* [3], stems from Fano's information-theoretical inequality [4] that uses the information loss in a noisy channel to compute a theoretical lower bound to the decoding error probability of a receiver.

Let the random variables X and Y represent the input the output messages of a noisy channel. A receiver operates a function $f(y)$ which, given the received message y, reconstructs an interpretation of the original input $\hat{x} = f(y)$. Let E be the binary random variable associated to an error event: $E = 1$ means that, given the channel's output y, the decoder's result $\hat{x} = f(y)$ is not the original channel input x. Fano's inequality is

$$\mathrm{H}(X|Y) \leq \mathrm{H}(E) + \Pr(E = 1) \log(N - 1), \qquad (2)$$

where N is the number of different channel inputs (the size of the support of X).

By equating the received message Y to a feature subset $\mathcal{S} \subseteq \mathcal{F}$, the original message X to the class C to be discovered, and the receiver's f to a supervised learning algorithm, then $\Pr(E = 1)$ is the classifier's error rate. By Fano's inequality (2), the error probability is bounded from below by a quantity that depends on the conditional entropy $\mathrm{H}(C|\mathcal{S})$, whose reduction is a necessary condition to lowering the error rate.

III. State of the art: filtering features and Mutual Information

To estimate the relevance or discrimination power of the individual features, one can proceed in a direct manner by directly testing the tentative feature set through repeated runs of the considered training model, or via proxy measures which do not depend on the machine learning model used [5], [6].

In detail, one identifies three classes of methods.

- *Wrapper methods* are built "around" a specific predictive model [7]. Each feature subset is used to train a model. The generalization performance of the trained model gives the score for that subset. Wrapper methods are computationally intensive, but usually provide the best performing feature set for the specific model.
- *Filter methods* use a proxy measure instead of the error rate to score a feature subset. Common measures include the the correlation coefficient [8] and Mutual Information between individual features and output class. Many filters provide a feature ranking rather than an explicit best feature subset.

1. **function** A-MIFS ($\mathcal{F}$, C, k, β)
2. **for each** feature $F \in \mathcal{F}$ compute $\mathrm{MI}(F, C)$;
3. find $F \in \mathcal{F}$ that maximizes $\mathrm{MI}(F, C)$;
4. $\mathcal{F} \leftarrow \mathcal{F} \setminus \{F\}$; $\mathcal{S} \leftarrow \{F\}$;
5. **repeat until** $|\mathcal{S}| = k$:
6. **for each** $F \in \mathcal{F}$ and $F' \in \mathcal{S}$:
7. compute $\mathrm{MI}(F, F')$ if not already available;
8. find $F \in \mathcal{F}$ that maximizes
9. $\mathrm{MI}(F, C) - \beta \sum_{F' \in \mathcal{S}} \mathrm{MI}(F, F')$;
10. $\mathcal{F} \leftarrow \mathcal{F} \setminus \{F\}$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{F\}$;
11. **return** $\mathcal{S}$;

Fig. 1. A-MIFS: given the set $\mathcal{F}$ of sample feature vectors and the corresponding output class vector C, return a subset $\mathcal{S}$ of k features; β is a weight factor (see text).

- *Embedded methods* perform feature selection as part of the model construction process. An example of this approach is the LASSO [9] method for constructing a linear model, which penalizes the regression coefficients, shrinking many of them to zero, so that the corresponding features can be eliminated.

Feature selection through mutual information consists of selecting a subset of features that have a high *joint* mutual information with the output. However, estimating mutual information for high dimensional random variables can be computationally intensive or even impossible. Feature selection algorithms based on mutual information usually reduce the problem to the evaluation of mutual information between couples of random variables. For example the Mutual Information-based Feature Selection algorithm [1] (here called A-MIFS to stress that it is based on an "Approximated" evaluation) can be described by the procedure shown in Fig. 1.

The parameter β regulates the relative importance of the MI between the candidate feature and the already-selected features with respect to the MI with the output class. A particular case occurs for $\beta = 0$, in which case only the mutual information of the individual features to the class matters (also called information gain or *InfoGain*), without any modification caused by their joint informativeness. A MI estimator based on Parzen density estimates is proposed in [10] to improve the performance of A-MIFS.

A normalized version of the average mutual information (NMIFS) is considered in [11], in which normalization is by the minimum entropy of both features. A drawback of A-MIFS is that the right-hand side term in the selection criterion (line 9 of Fig. 1) increase in magnitude with respect to the first term as the cardinality of the subset of selected features increases, so this term can become predominant in the choice of the optimizer. In addition A-MIFS requires searching for a proper value of β for controlling the redundancy penalization, which can depend on the problem. Now, from the definition of Mutual Information it follows that, for any couple of features:

$$0 \leq \mathrm{MI}(F_1; F_2) \leq \min\{\mathrm{H}(F_1), \mathrm{H}(F_2)\} \tag{3}$$

where $\mathrm{H}(\cdot)$ represent the entropy of a feature. As the entropy of a feature can vary greatly, measuring the importance of a feature using the MI without normalizing it can result in significant bias in the selection. The authors define the normalized Mutual Information as follows:

$$\mathrm{NI}(F_1, F_2) = \frac{\mathrm{MI}(F_1; F_2)}{\min\{\mathrm{H}(F_1), \mathrm{H}(F_2)\}}.$$

Furthermore, the authors define an alternative measure of redundancy between feature F and the subset of selected features $\mathcal{S}$, that take into account the two drawbacks previously presented:

$$\frac{1}{|\mathcal{S}|} \sum_{F' \in \mathcal{S}} \mathrm{NI}(F, F'). \tag{4}$$

And the selection criterion consists of choosing the feature that maximizes the measure G

$$G = \mathrm{MI}(F; C) - \frac{1}{|\mathcal{S}|} \sum_{F' \in \mathcal{S}} \mathrm{NI}(F, F'). \tag{5}$$

Now the parameter β has been substituted by an adaptive parameter that takes into account both the cardinality of the set $\mathcal{S}$ and the range of the Mutual Information. The NMIFS algorithm only differs from A-MIFS by the selection criterion in line 9 of Fig. 1 is replaced by (5) and by the absence of parameter β.

The heuristic Fast Correlation-Based Filter (FCBF) algorithm [12] uses symmetrical uncertainty $MI(X,Y)/\big(\mathrm{H}(X)+\mathrm{H}(Y)\big)$ as a quantitative criterion and adds features to the pool which are more highly correlated with the class than any of the features already in the pool. Let's note that the term "correlation" can be misleading in this context given that it is usually associated to the study of linear relationships.

An approximation based on triplets of random variables with with Conditional Mutual Information (CMIM) is proposed in [13]). In CMIM a feature F is good only if $\mathrm{MI}(C; F|F')$ (i.e., shared information between F and C, conditioned to the knowledge of F') is large *for every* F' already picked (this is why the *maximum* conditional MI is considered in the selection).

Other methods for MI estimation include quadratic divergence approaches [14] for use in feature extraction.A recent proposal in [15] argues that rather than compromising on the joint behavior of the selected features it is preferable to consider specific approximated density models. Their methods combines a Gaussian modeling of the feature responses, with derived upper bounds on their mutual information with the class label (which are subsequently maximized instead of the true MI estimates), and an advanced implementation to reduce the CPU time. In detail, a first approach, dubbed the "Gaussian compromise" (GC), uses the entropy of a single Gaussian of same expectation and variance as the mixture to get a reasonable estimate of the real entropy. A second approach, is based on a decomposition of the mutual information, in the binary case, as a sum of Kullback-Leibler divergence terms, which can be efficiently approximated (GKL). The four methods proposed maximize the entropy (GC.E) or the

mutual information (GC.MI) under the Gaussian compromise bound approximation, and maximizing the KL-based entropy (GKL.E) and mutual information (GKL.MI).

IV. Mutual Information in higher dimensions

The methods described up to now only consider pairwise MI (between features, or between a feature and the class). While this approach can be very efficient, it may lead to unnecessarily suboptimal results, because they do not attempt at maximizing the actual information content of the selected features as a whole.

In the following we present a simple algorithm for feature selection based on the evaluation of joint mutual information between the whole set of selected features and the class.

A. Exact mutual information of a sample

In the notation introduced in Sec. II-A, we suppose that a sample of N observations (individuals from a population) is given, where an observation $(f_1, \ldots, f_n, c)$ is an instantiation vector of the corresponding random variables $(F_1, \ldots, F_n, C)$.

Various methods have been described in the literature to estimate MI in an efficient way and with low bias, e.g., by estimating entropies on the basis of k-nearest neighbor distances [16]. In order to keep the code complexity to a minimum, however, we choose to use the sample entropy and MI measures as estimators for the population's true values, by approximating probabilities with the corresponding frequencies:

$$\begin{aligned} H(C) &\approx \sum_{c \in D_C} \mathrm{Fr}(c) \log \mathrm{Fr}(c) \\ &= \frac{1}{N} \left(\sum_{c \in D_C} N(c) \log \frac{N(c)}{N} \right), \end{aligned}$$

where D_C is the domain of class C, $N(c)$ is the number of occurrences of instance $c \in C$ among the observations and $\mathrm{Fr}(c) = N(c)/N$ is the corresponding frequency. If $\mathcal{S} \subseteq \mathcal{F}$ is a subset of features, then

$$\begin{aligned} \mathrm{MI}(\mathcal{S}; C) &\approx \sum_{\boldsymbol{f} \in D_{\mathcal{S}}} \sum_{c \in D_C} \mathrm{Fr}(\boldsymbol{f}, c) \log \frac{\mathrm{Fr}(\boldsymbol{f}, c)}{\mathrm{Fr}(\boldsymbol{f}) \mathrm{Fr}(c)} \\ &= \frac{1}{N} \left(\sum_{\boldsymbol{f} \in D_{\mathcal{S}}} \sum_{c \in D_C} N(\boldsymbol{f}, c) \log \frac{N \cdot N(\boldsymbol{f}, c)}{N(\boldsymbol{f}) N(c)} \right), \end{aligned}$$

where the outer sum spans all tuples $\boldsymbol{f} = (f_1, \ldots, f_{|\mathcal{S}|}) \in D_{\mathcal{S}}$ whose components belong to the domains of individual features in $\mathcal{S}$, while $\mathrm{Fr}(\boldsymbol{f}, c) = \mathrm{Fr}(f_1, \ldots, f_{|\mathcal{S}|}, c)$ is the joint frequency and $N(\boldsymbol{f}, c) = N(f_1, \ldots, f_{|\mathcal{S}|}, c)$ is the corresponding numerosity.

As long as the total number of values spanned by the sums (i.e., the joint domains of features in $\mathcal{S}$ and of C) is small with respect to the number N of samples, frequencies are good estimates of the corresponding probabilities, and the error with respect to the true values is low. However, the size of the joint domain of $\mathcal{S}$ grows exponentially with the number of features added to it, at least doubling every time a new feature is selected:

$$|D_{\mathcal{S}}| = \prod_{F \in S} D_f,$$

and this limits the number of features that can be added to $\mathcal{S}$. Moreover, from an implementation point of view, the size of the array of integer counters ("bins") needed to keep track of individual $N(\boldsymbol{f}, c)$ values can occupy a significant portion of memory and require long time for the final sum. This problem can be mitigated in various ways.

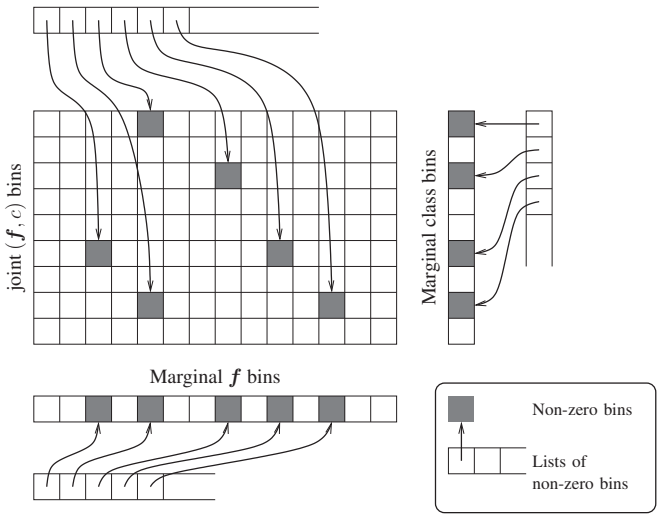

Fig. 2. Lists of non-zero bins used for optimization on sparse distributions.

1) Structures for sparseness: As the number of bins grows with respect to the sample size, many bins will remain zero (i.e., $N(\boldsymbol{f}, c)$ is sparse); therefore, ancillary data structures are created in order to span only nonzero counters. As shown in Fig. 2, a list of indices of non-zero entries is maintained for each of the counter arrays (one for the joint distribution of $(\boldsymbol{f}, c)$ and two for the marginal distributions of $\boldsymbol{f}$ and c). In this way, after the counting phase only a possibly small number of cells need to be scanned in the final sum.

2) Hashing: As the number of features exceeds a maximum number dictated by the available memory and CPU time for the computation, a hashing scheme is introduced to reduce the number of bins: the feature values $\boldsymbol{f}$ are not directly used to index the corresponding bin counters (the cells of Fig. 2), but they are mapped via a hashing function $H(\boldsymbol{f})$ to a smaller set of values. In our case, features are first mapped to a numerical index, and the remainder modulo the prime number $2^{20} - 3$ is taken

As shown in Sec.V, the CPU time required by this approach, while higher than that required by techniques based on radical approximations, is still manageable and generates results with higher quality.

B. Exact Mutual Information for Feature Selection: X-MIFS

As a result of the previous considerations, the proposed algorithm X-MIFS, described in Fig. 3, is a simple greedy

function X-MIFS ($\mathcal{F}$, C, k)
  $\mathcal{S} \leftarrow \emptyset$;
  repeat until $|\mathcal{S}| = k$:
    find $F \in \mathcal{F}$ that maximizes $\mathrm{MI}(\mathcal{S} \cup \{F\}, C)$;
    $\mathcal{F} \leftarrow \mathcal{F} \setminus \{F\}$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{F\}$;
  return $\mathcal{S}$;

Fig. 3. X-MIFS: input and output parameters are the same described in A-MIFS (see Fig. 1), with the exception β, which is not used.

selection algorithm based on the evaluation of exact joint mutual information: given a subset of already selected features $\mathcal{S} \subset \mathcal{F}$ (initially empty), the next feature to be selected is the one that maximizes $\mathrm{MI}(\mathcal{S} \cup \{F\}; C)$, i.e., the mutual information when added to $\mathcal{S}$.

The algorithm starts from an empty feature subset (differently from A-MIFS and NMIFS there is no need to bootstrap $\mathcal{S}$ by inserting the most informative feature) and proceeds until the required number of features has been selected.

V. Experimental results

This Section describes the results obtained by comparing the proposed algorithm with other state-of-the-art techniques, considering the performance of the feature extraction algorithm (Sec. V-C), the use of the resulting feature subset in a training task (Sec. V-D and V-E), and CPU times (Sec. V-F).

A. Datasets

1) Thrombin: The `thrombin` dataset[1] [17] describes the outcome of a drug design task. The dataset consists of 1,909 observations, each representing a compound tested for its ability to bind to a target site on thrombin. Each observation corresponds to a different chemical compound with 139,351 binary features describing chemical and spatial properties and a binary outcome ("active" or "inactive") related to interaction with thrombin. The dataset is extremely unbalanced: only 42 compounds out of 1,909 are found to be active, amounting to 2.2% of the samples, and features are very sparse.

2) CIFAR-10: The `CIFAR-10` dataset[2] [18] is meant for image classification tasks and contains 32×32-pixel RGB images representing objects from 10 different classes. Each class is represented by 5,000 samples in the training set and 1,000 samples in the test set, making the collection perfectly balanced with respect to the output class. Pixel values are numeric (and can be considered as continuous), class values range from 0 to 9. The images have been pre-processed as described in [19] by using the software provided by the authors, so that each image is finally represented by a vector of 2,048 real-valued features.

3) STL-10: The `STL-10` dataset[3] [20] contains 96×96-pixel RGB images representing objects from 10 different classes. Each class is represented by 500 samples in the training set and 800 samples in the test set. The dataset also contains unlabeled images then haven't been used in our work. The same pre-processing used for the `CIFAR-10` has also been adapted to this case to obtain 4,096 real-valued feature vectors.

[1] http://pages.cs.wisc.edu/~dpage/kddcup2001/
[2] http://www.cs.toronto.edu/~kriz/cifar.html
[3] https://cs.stanford.edu/~acoates/stl10/

B. Implementation

X-MIFS, A-MIFS and NMIFS have been implemented by the Authors in C++. CMIM has been tested on the C++ code provided by the author; the same code also implements the simpler MIM method, which just selects the k features sharing the highest mutual information with the class, and that we will use as baseline for evaluations.

The results for the GC.E, GC.MI, GKL.E and GKL.MI methods have been taken from Tables 5 and 6 of [15].

The MATLAB code needed to replicate the pre-processing steps for the two image datasets has been provided by the authors of [19], and the `octave` interpreter has been used to execute it.

The classifiers used in the tests (CART tree and SVM) are provided by the Python 2.7 `scikit-learn` package, namely `tree.DecisionTreeClassifier` and `svm.SVC`, created with the constructors' default parameters.

All tests have been run on an 8-core Intel Xeon 2.8GHz server with 32MB RAM running a 64-bit Linux OS; a single core has been used in all tests.

C. Comparison on binary feature extraction

As a first test we consider the task of selecting a small number of binary features out of 139,351 from the `thrombin` dataset. X-MIFS has been compared to A-MIFS, NMIFS, CMIM, and MIM, all suited for binary datasets.

The following task was repeated 30 times:

1) randomly extract 75% of the 1,909 observations as a training subset (the remaining 25% will be used in Sec. V-D as a test subset for the learning task);
2) compute the output class entropy on the training subset;
3) select 10 features by means of each tested algorithm;
4) compute the training subset's MI between the feature set and the outcome class by considering the first n features identified by each algorithm, $n = 1, \ldots, 10$.

Fig. 4 compares the outcome of the four algorithms. The top horizontal curve is the mean class entropy of the 30 training subsets, representing the theoretical upper bound for the mutual information. The four tested algorithms use the same criterion to select the first feature, so they always agree when $n = 1$ (apart from ties, which did not occur in this test); differences become more and more apparent as the number of selected features increases. Following [21], the difference between the population and the sample MI can be roughly estimated as

$$\Delta\mathrm{MI} \approx \frac{|D_C| \cdot |D_{\mathcal{S}}| - |D_C| - |D_{\mathcal{S}}|}{2N}. \tag{6}$$

In this case, since all data are binary, $|D_C| = 2$ and $|D_{\mathcal{S}}| = 2^{|\mathcal{S}|}$; for instance, given the low cardinality of the

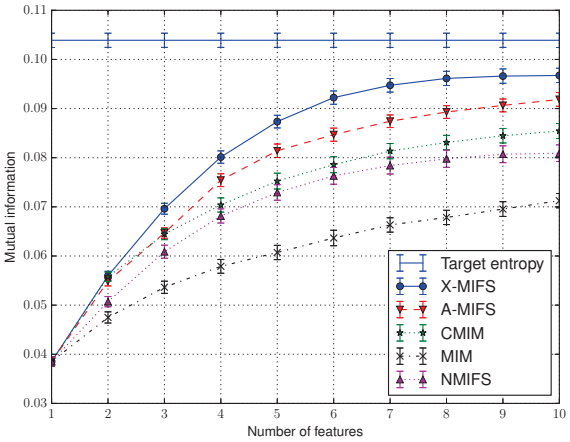

Fig. 4. Selecting features for the thrombin dataset. Bars represent the error of the mean across 30 experiments.

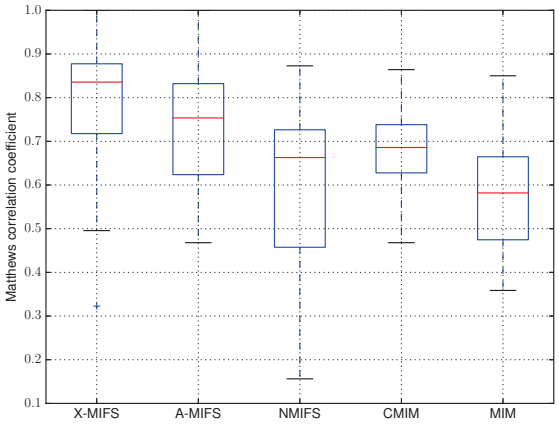

Fig. 5. Matthews correlation coefficient for the `thrombin` dataset: comparison among the different feature selection methods.

training subset ($N = 1,431$ samples), with 5 selected features we would have $\Delta\mathrm{MI} \approx 0.01$, which is comparable to the class entropy. The error bars shown in Fig. 4, representing the standard error of the average across the 30 training subsets, show that the actual difference is lower, and that the error (6) is clearly overestimated. However, the selection of a large number features would render the sample MI not very representative of the population's MI, and we decided to stop at 10.

X-MIFS is able to identify a more informative set of features, outperforming the state-of-the-art CMIM by about 10%, and the simpler A-MIFS technique by 5%.

In the following Section we verify whether the improvement in the training sample's mutual information obtained by X-MIFS corresponds to an improvement in the predictive power of a machine learning algorithm trained on the selected feature set.

D. Comparison on learning task

The features extracted in the experiments of Sec. V-C were used in a class prediction task. For each of the 30 $(75\%, 25\%)$-splits of the original `thrombin` dataset, and for each of the 4 feature selection algorithms, a CART tree was trained with the 10 features identified by the algorithm. The CART training algorithm performs the best Gini index-based split at each node, with no constraints on the number of levels or samples per node and uniform sample weight. Although this choice would easily lead to extreme overfitting in the original dataset, the 10-feature reduced training set mitigates the risk. Moreover, one of the aims of the test is to test the performance of each feature subset under suboptimal conditions; therefore no parameter of the training algorithm was changed from its default value.

The results on the test set are reported in Table I, where the median and inter-quartile range (IQR) of 30 training/test runs are reported for different evaluation criteria. The 30 runs

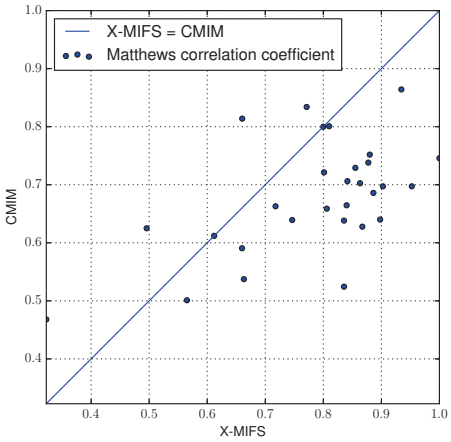

Fig. 6. Matthews correlation coefficient for the `thrombin` dataset: run-by-run comparison between X-MIFS and CMIM.

are executed on the same subset split used for feature selection. Given the large imbalance, many performance indices are quite high (trivially, a constant negative classifier would achieve .978 accuracy). Differences between the proposed feature subsets appear when balancing the overall accuracy by assigning equal weight sums to the positive and negative samples (balanced accuracy in the table), or when considering Positive-class sensitivity, precision of F_1-measure.

The Matthews correlation coefficient (MCC) in the last row of Table I is specifically designed to be robust with respect to strong imbalances; it assigns a much higher performance figure to X-MIFS with respect to the features selected by other techniques. The boxplot in Fig. 6 shows the MCC comparison in more detail; the boxes represent the inter-quartile ranges, while the whiskers extend to the first and last sample within 1.5IQR from the box. Finally, a sample-by-sample comparison

TABLE I
RESULTS OF THE CART TESTS FOR THE THROMBIN DATASET

		X-MIFS		A-MIFS		NMIFS		CMIM		MIM	
		Median	IQR	Median	IQR	Median	IQR	Median	IQR	Median	IQR
Accuracy:	Raw	0.992	0.006	0.990	0.006	0.987	0.008	0.987	0.006	0.985	0.004
	Balanced	0.863	0.097	0.808	0.114	0.792	0.226	0.798	0.083	0.711	0.098
Positive class:	Sensitivity	0.727	0.190	0.615	0.227	0.588	0.456	0.600	0.167	0.429	0.196
	Precision	0.889	0.188	0.909	0.167	0.833	0.286	0.778	0.175	0.750	0.250
	F_1-measure	0.828	0.161	0.737	0.199	0.667	0.358	0.667	0.116	0.533	0.205
Negative class:	Sensitivity	0.998	0.004	0.998	0.002	0.998	0.006	0.996	0.002	0.996	0.004
	Precision	0.994	0.004	0.992	0.006	0.989	0.010	0.991	0.006	0.987	0.006
	F_1-measure	0.996	0.003	0.995	0.003	0.994	0.004	0.994	0.003	0.992	0.002
Matthews correlation coefficient		0.836	0.160	0.754	0.208	0.663	0.269	0.686	0.110	0.582	0.190

of the outcomes of X-MIFS and CMIM is shown in Fig. 6, where the MCC obtained by each of the 30 tests on the X-MIFS feature subset is plotted against the MCC obtained with the MCC obtained with the CMIM features on the same dataset split. The diagonal line represents the locus of equal performance, points below the line correspond to tests where X-MIFS outperforms CMIM. Only 4 out of the 30 tests resulted in a better MCC performance for the CMIM-selected features.

E. Image Recognition Task

X-MIFS has been tested on the feature sets obtained after pre-processing the `CIFAR-10` and `STL-10` datasets as described in Sec. V-A.

Before feature selection, the real-valued features obtained in the pre-processing phase were binarized, using zero as threshold. Since A-MIFS, NMIFS and X-MIFS can work with non-binary features and classes, the 10-valued output class was not modified.

Table II shows the test accuracy (ratio of correct classifications) for a linear-kernel and a RBF-kernel SVM trained on 10- and 25-feature subsets identified by A-MIFS, NMIFS and X-MIFS, and compares them with the accuracies reported by [15] in the same conditions for the four algorithms GC.E, GC.MI, GKL.E, GKL.MI.

Boldface figures show the best performing algorithm; in almost all cases, X-MIFS shows better performance, particularly in the `STL-10` case, with a larger feature pool.

F. CPU time comparison

Fig. 7 shows the CPU time required for selecting an increasing number of features from the `thrombin` dataset by the two algorithms implemented by us (X-MIFS and A-MIFS) and the two for which we could run the original code provided by the authors (CMIM and MIM). The initial gap between the techniques is due to the fact that the MI computations in CMIM and MIM are optimized for binary features and classes, while in the X-MIFS and A-MIFS cases variables can have any finite domain size.

Time increase for A-MIFS, CMIM and MIM is almost linear (consider that the vertical scale of Fig. 7 is logarithmic), and Fig. 8 shows the same results in greater detail for these three methods. In about 0.1s the MIM implementation just fills up a logarithm lookup table, computes the MI between all features

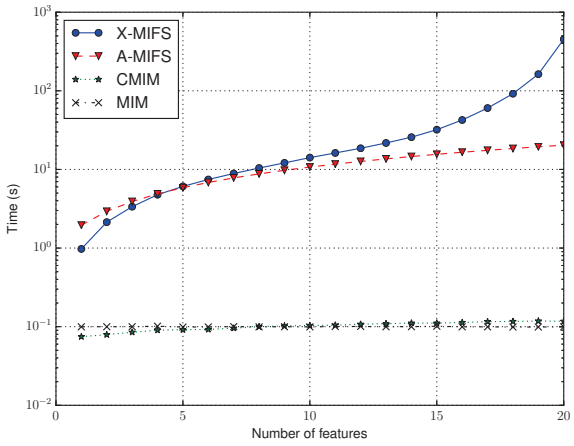

Fig. 7. CPU time comparison among four feature selection algorithms running on the `thrombin` dataset.

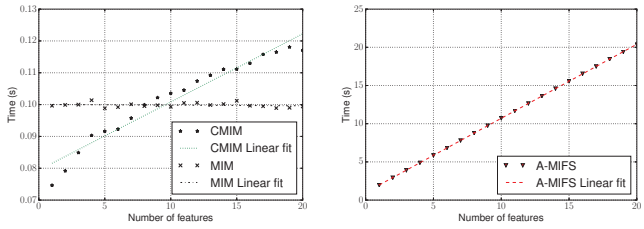

Fig. 8. CPU time vs. number of selected features for MIM, CMIM (left) and A-MIFS (right) running on the `thrombin` dataset.

and the class and performs a QuickSort run on the results; then it returns the largest ones, therefore additional features do not require extra time. The overhead of CMIM is lower, about 0.08s, because it does not need a sorting phase, but it requires additional computations at each step. In A-MIFS, more time-consuming data structures meant for general-case optimization are initialized (see Sec. IV-A), with a larger initial overhead of 1 second, after which every new feature requires about 1 second. The almost perfect alignment of measures in Fig. 8 is due to the fact that all times were measured within the same execution, while each timing of CMIM and MIM was measured in a different execution in order to minimize

TABLE II
COMPARISON WITH OTHER FEATURE-SELECTION METHODS ON IMAGE DATASET SVM TRAINING. BEST TWO RESULTS IN BOLD, BEST RESULT IS UNDERLINED.

Dataset	Features	Kernel	GC.E	GC.MI	GKL.E	GKL.MI	A-MIFS	NMIFS	X-MIFS
CIFAR-10 (2,048 features)	10	Linear	32.45	36.47	37.51	33.71	**37.69**	36.78	**38.76**
		RBF	35.29	39.57	39.84	34.49	**41.72**	40.17	**42.60**
	25	Linear	42.54	44.55	**46.41**	40.04	45.51	43.50	**46.89**
		RBF	51.12	49.91	52.80	43.09	**53.90**	50.80	**54.90**
STL-10 (4,096 features)	10	Linear	31.20	32.50	33.44	32.16	**38.62**	37.82	**39.21**
		RBF	36.16	35.86	39.67	35.95	**41.01**	40.34	**41.26**
	25	Linear	37.60	39.75	38.62	39.35	**46.62**	44.72	**45.75**
		RBF	42.64	43.35	46.31	41.65	**48.85**	46.56	**49.50**

the disruption of the original code. In all cases, data input and pre-processing were not included in the measures.

X-MIFS, on the other hand, requires a longer time at each feature because of the increasing size of the combined feature domain. We can identify an initial phase in which growth is almost linear, followed by a later steep increase when the required memory size causes an increasing number of cache misses, reducing the access speed and causing a significant increase in overall CPU time.

VI. CONCLUSIONS

When designing classification or regression systems in Machine Learning, maximizing the MI between input features and output is a powerful way to select a subset of features, independently of the specific ML model used. Arbitrary dependencies can be estimated (going beyond linear correlation) and insight can be derived.

Given the difficulties of estimating MI from samples in large-dimensional spaces, previous proposals considered radical approximations of the exact MI, specific distributions (notably Gaussian), approximated upper bounds.

This paper takes a bold approach of using the exact MI to select features. The boldness is in part justified by the much bigger CPU and memory available in current computers and by specific *ad hoc* data structures used for the task.

The results demonstrate that one can actually use the exact MI for significant applicative tasks and that superior results are indeed obtained w.r.t. the different approximated methods. The CPU times (tens of minutes in our largest experiments) are in many cases well acceptable, in particular if one considers that feature selection is normally done once in the lifetime of a ML model. In the future evolution of this work we are confident that more careful data structure and algorithm design will reduce the CPU times even more, therefore making bigger and bigger tasks solvable with the proposed X-MIFS algorithm.

ACKNOWLEDGMENTS

The research of Roberto Battiti was supported by the Russian Science Foundation, project no. 15–11–30022 "Global optimization, supercomputing computations, and applications".

REFERENCES

[1] R. Battiti, "Using the mutual information for selecting features in supervised neural net learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 537–550, 1994.

[2] R. Battiti and M. Brunato, *The LION way. Machine Learning* plus *Intelligent Optimization*. LIONlab, University of Trento, Italy, 2014.

[3] R. Linsker, "Self-organization in a perceptual network," *Computer*, vol. 21, no. 3, pp. 105–117, 1988.

[4] R. Fano, *Transmission of information; a statistical theory of communications*. M.I.T. Press, 1961.

[5] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[6] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 2, pp. 153–158, 1997.

[7] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[8] M. A. Hall and L. A. Smith, "Feature subset selection: a correlation based filter approach," in *International Conference on Neural Information Processing and Intelligent Information Systems*. Springer, 1997, pp. 855–858.

[9] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[10] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on parzen window," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 12, pp. 1667–1671, December 2002.

[11] P. Estevez, M. Tesmer, C. Perez, and J. Zurada, "Normalized mutual information feature selection," *Neural Networks, IEEE Transactions on*, vol. 20, no. 2, pp. 189–201, Feb 2009.

[12] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *ICML*, vol. 3, 2003, pp. 856–863.

[13] F. Fleuret, "Fast binary feature selection with conditional mutual information," *Journal of Machine Learning Research*, vol. 5, pp. 1531–1555, 2004.

[14] K. Torkkola, "Feature extraction by non-parametric mutual information maximization," *Journal of Machine Learning Research*, vol. 3, pp. 1415–1438, March 2003.

[15] L. Lefakis and F. Fleuret, "Jointly informative feature selection," *Journal of Machine Learning Research (JMLR)*, 2016, to appear. [Online]. Available: http://fleuret.org/papers/lefakis-fleuret-jmlr2016-preprint.pdf

[16] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical Review E*, vol. 69, no. 6, 2004.

[17] J. Cheng, C. Hatzis, H. Hayashi, M.-A. Krogel, S. Morishita, D. Page, and J. Sese, "Kdd cup 2001 report," *SIGKDD Explor. Newsl.*, vol. 3, no. 2, pp. 47–64, Jan. 2002. [Online]. Available: http://doi.acm.org/10.1145/507515.507523

[18] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep. MSc Thesis, Apr. 2009.

[19] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 921–928.

[20] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *International conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[21] W. Li, "Mutual information functions versus correlation functions," *Journal of statistical physics*, vol. 60, no. 5-6, pp. 823–837, 1990.