This is a *pre-print* version of the article accepted for inclusion in *IEEE Trans. Neural Netw. Learn. Syst.* with DOI: 10.1109/TNNLS.2018.2830700

Feature Selection based on the Neighborhood Entropy

Andrea Mariello, *Member, IEEE,* and Roberto Battiti, *Fellow, IEEE*

***Abstract*—In feature selection, a measure that captures non-linear relationships between features and class is the Mutual Information, which is based on how information in the features reduces the uncertainty in the output. In this work we propose a new measure that is related to Mutual Information, called Neighborhood Entropy, and a novel filter method based on its minimization in a greedy procedure. Our algorithm integrates sequential forward selection with approximated nearest-neighbors techniques and locality-sensitive hashing (LSH). Experiments show that the classification accuracy is usually higher than that of other state-of-the-art algorithms, with the best results obtained with problems that are highly unbalanced and non-linearly separable. The order by which the features are selected is also better, leading to a higher accuracy for fewer features. The experimental results indicate that our technique can be employed effectively in off-line scenarios when one can dedicate more CPU time to achieve superior results and more robustness to noise and to class imbalance.**

***Index Terms*—Feature evaluation and selection, Information theory, Indexing methods, Machine learning.**

I. Introduction

EXPLOITING huge collections of data is critical in many applications of data mining and machine learning. However, when using such datasets in classification tasks, one faces the so-called *curse of dimensionality*: the number of dimensions can become too high to get a good classifier in a reasonable time. In this context, dimensionality reduction can help to:

- generalize better by using a smaller number of free parameters and the most relevant information;
- avoid the so-called *peaking phenomenon* [1], that is, the situation in which the classification accuracy steadily grows as the number of features grows until a point is reached when adding new dimensions to the system makes the classification accuracy stable or even worse;
- understand the obtained machine learning systems, by making them more interpretable by humans [2].

Dimensionality reduction can be achieved by *extracting* new features (e.g. through projections [3]–[5]) or by *selecting* a subset of the original features. In order to select the most relevant features with respect to the output class, several methods have been proposed. By considering the relationship between feature selection and classification algorithms, the various methods can be classified into:

- *wrapper* methods, which use a predictive model and its generalization performance to rank the features [6];
- *embedded* methods, which select the most relevant features during the model construction [7]–[10];
- *filter* methods, which rank and select features by using a proxy measure (*relevance* score) instead of the classifier error rate.

Filter methods ignore the interaction with the classifier and are usually faster than wrapper methods, so that they can easily scale to very high-dimensional datasets. According to the particular criterion used to obtain the relevance score, different techniques are based on:

- the Pearson correlation coefficient [11], [12];
- statistical tests like the t-test [13];
- the Fisher linear discriminant [14];
- the class of the nearest neighbors [15], [16];
- the trace ratio criterion for graph-based Fisher and Laplacian scores [17];
- the mutual information [5], [18]–[21].

The methods based on the correlation coefficient capture linear relationships between random variables but they cannot deal with non-linear relationships. The methods based on the t-test and the Fisher linear discriminant require that the features originate from a normal probability distribution. The mutual information (MI) measures arbitrary dependencies between random variables, and it is suitable for assessing the *information content* of features in complex classification tasks. Moreover, the MI measure does not depend on the particular machine learning model or choice of coordinates.

This work proposes a new algorithm for filtering features that maximizes the MI between the selected subset and the class variable and at the same time tries to minimize the MI between the selected features. A new MI-related measure is introduced, called Neighborhood Entropy, which can be applied to multiple features, so as to overcome the limitation of traditional pairwise MI estimators. Moreover, this measure can be computed by using efficient data structures for indexing a dataset and some approximated techniques for reducing the execution time in case of huge collections of data. The proposed score does not involve the explicit estimation of probability distributions or the computation of local and global affinity matrices of graph-based methods, and can be used with features of integer or real values.

The structure of the remainder of this paper is as follows. Section II describes in more detail some related work and several concepts related to the use of Information Theory for selecting features. Our algorithm and the proposed new measure are presented in Section III. Section IV explains how methods for finding approximated nearest neighbors and locality-sensitive hashing (LSH) are used in our procedure.

A. Mariello and R. Battiti are with the Department of Information Engineering and Computer Science - DISI, University of Trento, Povo (TN), Italy. E-mails: {andrea.mariello, roberto.battiti}@unitn.it

Finally, Section V reports an experimental comparison of our solution with respect to the state of the art. Experiments show that classifiers trained on features selected by our technique achieve a better accuracy with fewer features in noisy and unbalanced contexts, at the expense of more computation time during feature selection.

II. Distribution-Independent Filter Methods

Let us fix the notation and introduce some useful definitions. N is the number of samples (points or patterns) in a dataset and D the number of features (or dimensions). The set of samples can be represented as a matrix $\mathcal{F}$ of size $N \times D$. Each sample is assigned a class (category or label) belonging to the set $\{1, 2, \ldots, N_c\}$. These values are collected in a *class vector* C of size N. The d-th column vector of size N from $\mathcal{F}$, $1 \leq d \leq D$, is the *feature vector* F_d.

The simplified notation $\Pr(x)$ indicates the probability of X being equal to x, that is, $\Pr(X = x)$, and $\mathrm{Fr}(x)$ is its sample estimator.

A. RELIEF and RELIEFF

RELIEF [15] is a feature selection algorithm for binary or continuous input data in classification tasks. There are no particular assumptions on the distribution of data, apart from the hypothesis that similar samples tend to belong to the same class. This assumption underlies most machine learning techniques, for different measures of similarity. RELIEF expresses the importance of each feature by a weight in a D-dimensional vector W, which is initialized to zero. Then the following procedure is repeated an arbitrary number of times:

1) select at random a sample X from $\mathcal{F}$;
2) find the nearest neighbor that belongs to the same class of X. This sample is called the *near-hit* of X, $h(X)$;
3) find the nearest neighbor that belongs to the class that is different from the class of X. This sample is called the *near-miss* of X, $m(X)$;
4) update the d-th weight in W in the following way:

$$W_d \leftarrow W_d - (X_d - h_d(X))^2 + (X_d - m_d(X))^2.$$

The weight of a feature increases if the near-hit is closer than the near-miss and decreases otherwise. Then one can pick the features with weights that are greater than a given threshold or sort the features according to the weights and pick the ones corresponding to the s biggest weights.

RELIEFF [16] is an improved version of RELIEF extended to multi-class problems that updates the weights with the average contribution of the k-near hits and the k-near misses for each of the classes that are different from the class of the random sample. As concerns the near misses, it averages the contribution of each class, weighted with the sample estimator of its prior probability.

B. Feature Selection based on Mutual Information

This section presents some of the most successful approaches based on the mutual information, starting from [18].

Definition 1 (Entropy): Given a classification task, denote by $\Pr(c), c = 1, \ldots N_c$ the probabilities for the different classes. The initial uncertainty in the classes is measured by the entropy:

$$H(C) = -\sum_{c=1}^{N_c} \Pr(c) \log \Pr(c). \tag{1}$$

Definition 2 (Conditional Entropy): Let $\mathcal{D}_d$ be the set of all possible values that the d-th feature can assume, $1 \leq d \leq D$, and call $\mathcal{D}_C$ the set of the different classes. The average uncertainty after knowing the d-th feature is the conditional entropy, which is defined as follows:

$$H(C|F_d) = -\sum_{f \in \mathcal{D}_d} \Pr(f) \sum_{c=1}^{N_c} \Pr(c|f) \log \Pr(c|f). \tag{2}$$

Definition 3 (Mutual Information): The mutual information (MI) between the class C and a feature F_d is defined as follows:

$$\mathrm{I}(C; F_d) = \sum_{c \in \mathcal{D}_C, f \in \mathcal{D}_d} \Pr(c, f) \log \frac{\Pr(c, f)}{\Pr(c) \Pr(f)}. \tag{3}$$

In general, the conditional entropy is less or equal to the initial entropy and is equal if and only if the feature and the output class are independent random variables. The mutual information corresponds to the amount by which the entropy decreases when the feature F_d is known:

$$\mathrm{I}(C; F_d) = H(C) - H(C|F_d). \tag{4}$$

Note that the mutual information is symmetric in C and F_d.

The so-called *Infomax Principle* [22] states that the MI is indeed related to the accuracy of a learning system because of Fano's inequality [23].

Theorem 1 (Fano's Inequality): Let the random variables X and Y represent the input and the output messages of a noisy channel. A receiver operates a function $f(y)$ which, given the received message y, tries to reconstruct the original input x with $\hat{x} = f(y)$. Let E be the binary random variable associated with the event of an error, that is, $E = 1$ when, given the channel's output y, the result of the decoder $\hat{x}$ is different from the original input x. Then the following inequality holds:

$$H(X|Y) \leq H(E) + \Pr(E = 1) \log(N - 1), \tag{5}$$

where N is the number of different channel inputs.

By equating Y to a feature F_d, X to the class C, and the function operated by the receiver $f(\cdot)$ to an algorithm for classification, then $\Pr(E = 1)$ can be seen as the error rate of the classifier. By (5), this probability has a lower bound that is proportional to the conditional entropy $H(C|F_d)$. Therefore, reducing this quantity, which corresponds to increasing the MI, is a necessary condition to lowering the error rate and building an effective classifier. Unfortunately, the condition is not sufficient, because one needs a machine learning model capable of *extracting* the available information content.

A naïve strategy for selecting features according to their MI is as follows. One computes the value of the MI between the class variable and each individual feature. Then one selects the features with the highest s values. We refer to this method as Feature Selection based on the MI and the criterion of *Maximum Relevance* (MR-MIFS), as in [24]. The probabilities involved in the computation of the pairwise MI can be estimated by using histograms as in [18], [19] or the Kernel Density Estimation (KDE) as in [25].

Another procedure for feature selection is introduced in [18]. The criterion used is that of *minimum redundancy* and *Maximum Relevance*: the algorithm starts by selecting the most relevant feature, according to the MI with the class, and then it keeps adding features that are relevant but not redundant, that is, having a high value of MI with respect to the class but a low value of MI with respect to each individual feature that has already been selected. This twofold objective is achieved by maximizing the following quantity:

$$\mathrm{I}(C; F_d) - \beta \sum_{F' \in \mathcal{S}} \mathrm{I}(F_d, F'), \tag{6}$$

where β is a weight that regulates the trade-off between relevance and redundancy, and $\mathcal{S}$ is the set of already-selected features. For $\beta = 0$ the algorithm is equal to MR-MIFS.

A drawback of the previous procedure, namely mRMR-MIFS, is that the second term in (6) can increase in magnitude with respect to the first term when the cardinality of $\mathcal{S}$ grows [19]. Therefore, it can become predominant in the choice of the next feature to select. Moreover, the algorithm needs a proper value of the β parameter. In [20] an alternative measure of redundancy is proposed, in which the parameter β is substituted by an adaptive parameter that takes into account the cardinality of the set $\mathcal{S}$.

The method proposed in [21] selects features according to a quadratic divergence approach, while the technique in [26] uses fixed approximated density models (derived from Gaussian models) to maximize upper bounds on the MI with the class, without the evaluation of the joint behavior of the selected features.

The relevance and redundancy criteria of mRMR-MIFS have also been used in methods not based on the MI. A semi-supervised method based on correlation has been recently proposed in [12] and is called RRPC. The main difference from mRMR-MIFS is the maximization of the following quantity:

$$\mathrm{P}(C_L; F_{Ld}) - \frac{1}{|\mathcal{S}|} \sum_{F' \in \mathcal{S}} \mathrm{P}(F_d, F'), \tag{7}$$

where $\mathrm{P}(C_L; F_{Ld})$ is the Pearson correlation coefficient between the class and a feature, which is computed only on the labeled data, while $\mathrm{P}(F_d, F')$ includes also the unlabeled data.

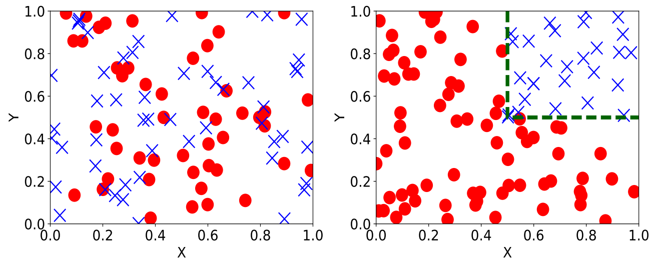
The previous methods use a pairwise computation of the chosen score, either between a feature and the class or between two features. They account for relevance and redundancy separately. The technique proposed in [19] is instead based on a generalization of the MI to multiple features. The X-MIFS algorithm evaluates the MI between the whole set of selected features and the class, so as to add only those features that are relevant when considered together. The main difference of this approach is that it does not consider two one-dimensional random variables as before (the class variable C and one feature F_d). It computes the MI between a one-dimensional random variable (C) and a multidimensional random variable $\mathcal{S} = (F_{i_1}, F_{i_2}, \ldots, F_{i_m})$, where $0 < m \leq D$, consisting of m distinct features from $\{F_1, F_2, \ldots, F_D\}$. $\mathcal{S}$ assumes values in $\mathcal{D}_\mathcal{S} = \mathcal{D}_{i_1} \times \mathcal{D}_{i_2} \times \cdots \times \mathcal{D}_{i_m}$. To compute the MI according to (3), the sum has to be taken over all possible values in $\mathcal{D}_\mathcal{S}$. The probability distributions for the computation of the MI as in (3) are evaluated by using histograms and the relative frequencies of each single value in $\mathcal{D}_\mathcal{S}$.

A drawback of X-MIFS is the memory necessary to store the frequencies for all the values in $\mathcal{D}_\mathcal{S}$. As an example, with binary features the number of these values is equal to $2^{|\mathcal{S}|}$. Another possibility for a fast evaluation of the MI by using a certain level of approximation and an index with a smaller memory footprint is given in [27]. The authors estimate the MI between two continuous variables after making the hypothesis that, for a given point, the distribution in its neighborhood (within a sufficiently small radius) remains constant. They propose two estimators based on the approximated k-nearest neighbors retrieved from the space identified by the two variables. These estimators can be also used to compute the MI between multiple variables, but they cannot be used for classification, when one needs the MI between a numerical feature and a categorical class, since there is no concept of distance among categories. Our technique evaluates an MI-related measure with the same idea of approximated neighborhoods. However, the search for the nearest neighbors is performed only on the features, thus making our method suitable for classification.

III. NEFS: Feature Selection based on the Neighborhood Entropy

The algorithm proposed in this work starts by selecting features through a greedy procedure that maximizes the MI, as in X-MIFS. To maximize (4), which is equivalent to (3), the first term in the equation (the class entropy) can be omitted because it is constant. One is left with the minimization of the conditional entropy $H(C|F_d), 1 \leq d \leq D$. One of the necessary conditions for a successful classifier is that similar points should belong to the same class, with the exception of those points that are on the class boundary. If this assumption holds true for a specific feature, then the uncertainty in determining the class (the conditional entropy) should be very small and the learning process can benefit from considering that feature. On the contrary, when for a specific feature the class entropy is very high, the classification accuracy cannot improve by considering that feature (sometimes it can even get worse).

A toy example is illustrated in Figure 1, with 100 points drawn from a uniform distribution defined on the unit square $[0, 1] \times [0, 1]$ and with only 2 classes. In one case each class is randomly assigned to one half of the points (see Figure 1a), while in the other all the points with both of the coordinates that are greater than 0.5 belong to one class and all the remaining points belong to the other class (see Figure 1b). In the first case the conditional class entropy is maximum and equal to the class entropy ($\log 2$), and the features (X and Y) cannot be used effectively for the classification. In the second case, the separation surface is completely defined by the two features, which are then relevant for the classification since



(a) (b)

Fig. 1. The Neighborhood Entropy evaluated for the case of high uncertainty (a) is higher than the value retrieved for the case of low uncertainty (b).

their conditional entropy is minimum.
A possible strategy for evaluating the class entropy for a specific feature without estimating probability distributions is the following: one considers the class values of the samples in the neighborhood of a point, computes the class entropy only for those points and then takes the average over all the points in the dataset. In the case of very high entropy (as in Figure 1a), the average uncertainty computed in this way remains very close to the theoretical value of $\log 2$. In the second case (Figure 1b), for most of the points there is no uncertainty in their neighborhood and the average conditional class entropy is very close to 0. Therefore, one can use the conditional class entropy evaluated on the neighborhoods of the points as a relevance score for selecting the most informative features. It is also worth noting that, even in the situation of non-uniform distributions, the previous observations remain valid if one considers neighborhoods with a fixed number of neighbors instead of a fixed radius. The concept of Neighborhood Entropy is related to the previous motivations.

Definition 4 (Neighborhood Conditional Entropy): Given a classification task as defined in Section II and a multidimensional random variable $\mathcal{S}$, $\mathcal{N}(X;k)$ is the set of a point X, which is an instance of $\mathcal{S}$, and its k-nearest neighbors. Then the relative frequency of a class $c \in \mathcal{D}_C$ within $\mathcal{N}(X;k)$ is:

$$\mathrm{f}_k(c,X) = \mathrm{Fr}(c|\mathcal{N}(X;k)) = \sum_{Y \in \mathcal{N}(X;k)} \frac{n(C=c,\mathcal{S}=Y)}{|\mathcal{N}(X;k)|},$$

with $n(\cdot)$ counting the occurrences of an event.
The k-Neighborhood Conditional Entropy (NE) of the class variable C with respect to $\mathcal{S}$ is then defined as follows:

$$\mathrm{NE}_k(C;\mathcal{S}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c\in\mathcal{D}_C} \mathrm{f}_k(c,X^i)\cdot\log(\mathrm{f}_k(c,X^i)), \quad (8)$$

where X^i represents the i-th instance of $\mathcal{S}$.

Note that our k has an effect similar to the bandwidth of a kernel density estimator. Choosing a large k corresponds to using wider kernels for approximating the distribution at a point.
Selecting features by minimizing the NE is equivalent to maximizing the MI as in (3) or (4), since one can approximate the conditional entropy in (2) with the following:

$$\mathbb{E}_{\mathcal{N}(\mathcal{S};k)}[H(C|\mathcal{S})] \approx \frac{1}{N}\sum_{i=1}^{N} H(C|\mathcal{N}(X^i;k)) = \mathrm{NE}_k(C;\mathcal{S}).$$

```
1: **function** NEFS($\mathcal{F}$,$C$,$k$,$s$)
2:     $\mathcal{S} \leftarrow \emptyset$; $\mathcal{R} \leftarrow \{F_1, F_2, \ldots, F_D\}$
3:     **repeat**
4:         find $F^*$ that minimizes $\mathrm{NE}_k(C;\mathcal{S}\cup\{F\}), \forall F \in \mathcal{R}$
5:         $\mathcal{R} \leftarrow \mathcal{R} \setminus \{F^*\}$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{F^*\}$
6:     **until** $|\mathcal{S}| = s$
7:     **return** $\mathcal{S}$
8: **end function**
```

Fig. 2. NEFS: feature selection based on the Neighborhood Entropy (8).

The distinctive feature of the NE with respect to the MI is that the former allows selecting features by maximizing the MI with no need to estimate the feature probability distributions directly. Instead it employs an adaptive estimator of the class uncertainty by focusing only on the class values in a neighborhood of a point. Our proposed strategy, called *NEFS*, is reported in Figure 2. It is based on the same greedy procedure used by X-MIFS: at each iteration the algorithm selects the feature that minimizes the NE of the output variable with respect to the union of that feature with the already-selected subset. The main difference is in the cost of evaluating the NE ($\mathcal{C}_{\mathrm{NE}}$), which is related to the specific implementation of the nearest neighbor algorithm. If i is the number of already-selected features at the iteration $i+1$, the CPU time complexity of NEFS for selecting s features can be expressed as follows:

$$\mathcal{C}_{\mathrm{NEFS}} \;=\; \mathcal{C}_{\mathrm{NE}}\sum_{i=0}^{s-1} D - i = O(\mathcal{C}_{\mathrm{NE}}\;Ds). \quad (9)$$

Since a crucial building block is a method to search for neighbors, the next Section explores some of the most efficient techniques for searching the nearest neighbors, with particular attention to *Locality-Sensitive Hashing* (LSH). LSH is the chosen method in our proposal, since it is the fastest when one can tolerate a certain level of approximation.

IV. NEFS Implementation

To evaluate the NE as in (8), one needs to search for the k-nearest neighbors for each point in the dataset, by using distances computed on spaces of varying dimensions. Let us define the different problems related to searching for nearest neighbors.
Assume a set $\mathcal{F}$ of N points defined on a space $\mathcal{M}$ of D dimensions, with a distance function $dist : \mathcal{M} \times \mathcal{M} \mapsto \mathbb{R}_0^+$.

Definition 5 (k-Nearest Neighbors problem (k-NN)): Given any point $Q \in \mathcal{M}$, find the points $P_1, P_2, \ldots, P_k \in \mathcal{F}$ that have the smallest, the second-smallest, ..., the k-th smallest distances to Q, respectively.

Definition 6 (ϵ-approximated k-Nearest Neighbors problem (ϵk-NN)): Given any point $Q \in \mathcal{M}$ and an approximation factor $\epsilon > 0$, find a set of points $P_1, P_2, \ldots, P_k \in \mathcal{F}$ such that $dist(P_i, Q) \leq (1+\epsilon)dist(P_i^*, Q), i = 1, 2, \ldots, k$, with P_i^* being the i-th true nearest neighbor of Q.

Let us make the hypothesis of having a dataset with $D < N < e^D$. This corresponds to having more samples than features, with a very loose upper bound that can hardly be reached in real non-trivial feature selection problems (all the datasets used for the experiments except one fall into this category). In this context, computing the distances for each point is $O(DN)$. If one uses a *min-heap* for sorting, building the heap with the N distances and then retrieving the k smallest is $O(N + k \log N)$. It is also required that $k < N$ (usually $k \ll N$). Under the hypothesis of $D < N < e^D$, which implies $D > \log N$, the asymptotic complexity of the algorithm that solves the k-NN problem is dominated by the term relative to the computation of distances $O(DN)$. However, since NEFS needs to find the nearest neighbors of each individual point in the dataset to estimate the NE, one is actually faced with two other problems, defined below.

Definition 7 (All k-Nearest Neighbors problem (A-k-NN)): For each point $Q \in \mathcal{F}$, find its k-nearest neighbors $P_1, P_2, \ldots, P_k \in \mathcal{F} \setminus \{Q\}$.

Definition 8 (All ϵ-approximated k-Nearest Neighbors problem (A-ϵk-NN)): Given an approximation factor $\epsilon > 0$, for each point $Q \in \mathcal{F}$, find its ϵ-approximated k-nearest neighbors $P_1, P_2, \ldots, P_k \in \mathcal{F} \setminus \{Q\}$.

The A-k-NN problem can be solved in $O(DN^2)$ time, because one can solve an instance of the k-NN problem for each point in the dataset. With a fixed D, the same problem can be solved in $O(N \log N)$ time by recursively building a tree of neighboring boxes [28], [29]. However, these algorithms have an exponential dependency on the number of dimensions, with a worst-case running time of $O((cD)^D N \log N)$, for a dimension-independent constant c.

In our context, one aims at solving the A-ϵk-NN problem, so as to reduce the query time by tolerating a certain level of approximation. Intuitively, retrieving approximated neighbors corresponds to possibly using a larger radius for the neighborhood of a point. Depending on the particular dataset, if k is sufficiently small, the retrieved points are still good indicators of the class entropy near the query point. It is therefore convenient to revert to some approximated techniques that solve the A-ϵk-NN problem in $O(DN^\alpha)$ time, with $\alpha \approx 1$.

The main idea in this case is to build an index that groups points in their approximated neighborhoods, before the actual evaluation of distances. During the querying phase, the index should help in discarding most of the points in approximately constant time. Then one is left with the computation of the distances and the search for the nearest neighbors among the remaining points. In [30] the authors define a technique called *Locality-Sensitive Hashing (LSH)* for the NN search. They propose to hash the points in a dataset with particular hash functions that ensure, with some probability, that points that are closer to each other collide into the same bucket, while distant points fall into different buckets. One can then retrieve the list of candidates by looking at the points of the bucket corresponding to the query point.

Other possibilities include space-partitioning schemes based on regular grids, which work better with uniformly distributed data, or clustering techniques such as the *k-means* algorithm, which dynamically adapts to the dataset distribution [31]. However, the index building time grows significantly compared to LSH (described in the next section), since in the new scenario a clustering algorithm has to be run to find the partition.

The space can be also partitioned by using a k-dimensional tree or kd tree [32], [33]. In general, kd trees are the preferred choice for speeding up the exact NN search in low-dimensional spaces but they can be as inefficient as an exhaustive search in case of high-dimensional spaces [31].

The NEFS algorithm is implemented together with the LSH method, since it is the fastest and provides guarantees on the accuracy of the results, as shown in the next Section.

A. Locality-Sensitive Hashing

Let us introduce some useful concepts about LSH derived from the original paper [30].

Let *dist* be a distance function between elements of a set $\mathcal{F}$ of N points defined in a metric space $\mathcal{M}$ of D dimensions, and for any point $P \in \mathcal{F}$ let $\mathcal{B}(P; r)$ denote the set of points that are within a distance r from P, according to *dist*.

Definition 9 (Locality-Sensitive Hash functions): A family $\mathcal{H}$ of hash functions $h(\cdot)$ is called (r_1, r_2, p_1, p_2)-sensitive for *dist*, with $r_1 < r_2$ and $p_1 > p_2$, if for any couple of points $P, Q \in \mathcal{F}$:

- if $P \in \mathcal{B}(Q; r_1)$ then $\Pr_{\mathcal{H}}(h(Q) = h(P)) \geq p_1$;
- if $P \notin \mathcal{B}(Q; r_2)$ then $\Pr_{\mathcal{H}}(h(Q) = h(P)) \leq p_2$.

In NEFS, LSH is implemented with a family of data-independent locality-sensitive hash functions that are suitable for Euclidean spaces and are proposed in [34].

These hash functions are defined as:

$$h(v) = \left\lfloor \frac{a \cdot v + b}{w} \right\rfloor, \tag{10}$$

where w is a fixed positive integer, b is randomly drawn from $[0, w]$ and $a \in \mathbb{R}^D$ is drawn from a Cauchy distribution, in case of a procedure using l_1-norm distances, or from a Gaussian distribution, in case of l_2-norm distances. Since the guarantees provided by LSH hash functions do not change when one uses the l_1-norm instead of the l_2-norm, and one is interested only in an approximated neighborhood, for the implementation of NEFS we opted for the l_1-norm distance, which is faster to compute. This can reduce the running time significantly, since the computation of a distance is one of the most frequent operations of the procedure.

In the standard version of NEFS, a new LSH index is built each time the NE between a set of features and the class is evaluated. This happens $O(Ds)$ times as reported in (9). However, L new hash functions are generated only once for each iteration of the loop in Figure 2, that is, when the cardinality of the set of selected features changes. For each iteration $i = 1, 2, \ldots, s$ the procedure generates L new hash functions. For each function, it generates the scalar b and the vector $a \in \mathbb{R}^i$. This corresponds to selecting a number of coefficients equal to $\sum_{i=1}^{s} (i + 1)L = O(Ls^2)$.

The actual index construction is done every time a new feature

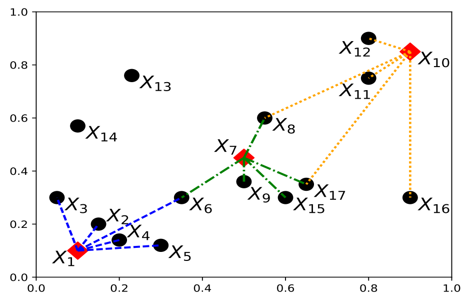


Fig. 3. First 3 iterations of NEFS on 17 two-dimensional points and $k = 5$. The first iteration selects the nearest neighbors of X_1. Since $X_i, i = 2, \dots, 6$ have already been considered, the second iteration selects the neighbors of X_7. In this case X_6 is selected again. The third neighborhood is that of X_{10}.

is considered. In this case, each iteration requires hashing each point in the dataset L times by using (10). The total number of iterations for building the indexes is $\sum_{i=0}^{s-1} D - i$ as in (9). Since, for $i = 0, 1, \dots, s-1$, applying (10) is $O(2i+3)$ and the number of hashed values is NL, the total cost of the building step is $O\left(NL\sum_{i=0}^{s-1}(D-i)(2i+3)\right) = O(NLDs^2)$. Once the index has been built, the nearest neighbors for each point in the dataset are retrieved by using the hash values of the points to find the promising candidates in each of the L hash tables. Then the standard nearest neighbor search is performed on the list of candidate points to find the k-nearest neighbors among them. As reported in [30], the time for one query on D-dimensional points is $O(DN^{1/(1+\epsilon)})$.
If $i+1$ represents the number of features on which each index is built for $i = 0, 1, \dots, s-1$, retrieving the neighbors for one point is $O((i+1)N^{1/(1+\epsilon)})$. Therefore the total cost for solving the A-ϵk-NN problem for estimating the NE for all the subsets of features is

$$O\left(\sum_{i=0}^{s-1}(D-i)N(i+1)N^{1/(1+\epsilon)}\right) = O(N^{(2+\epsilon)/(1+\epsilon)}Ds^2).$$

Consequently, if $\alpha = (2+\epsilon)/(1+\epsilon)$, the complexity of NEFS can be expressed as follows:

$$\mathcal{C}_{\text{NEFS}} = O(Ls^2 + NLDs^2 + N^\alpha Ds^2) = O(Ds^2N^\alpha). \quad (11)$$

Note that the last equality holds because, according to the equations in [30], given a high probability of collision for neighbors (e.g. 0.9) and a low probability of collision for distant points, one has $L \approx N^{1/(1+\epsilon)}$.

To reduce the time taken by the algorithm for the NN queries, in our implementation the NE is evaluated on the neighborhoods of points not already visited within the neighborhood of another point (see Figure 3). Then the average is taken only on the visited points. This is equivalent to reducing the number of estimators of the NE from N to roughly N/k. The decrease in the running time of NEFS is proportional to k (not equal because the index is still built with all the points in the dataset to exploit as much information content as possible). This modification resembles the random selection of points in RELIEFF, with the main difference being that the choice of points is not random but guided by the points themselves.

Two different implementations of NEFS have been developed: in the first version an LSH index is built each time the NE is evaluated, by using only the current subset of features involved in the NE computation. In the second version, which we refer to as *Single-indexed* NEFS (SNEFS), the LSH index is built only once on the D-dimensional space identified by all the features. When the NE on a subset of features is evaluated, at first the procedure retrieves the neighbors of a point in the full-dimensional space. Then a search is done for the nearest neighbors among the candidates, after they have been projected onto the reduced space. This idea originates from the fact that neighbors in D dimensions are also neighbors in s dimensions, with $s < D$, and one does not need the exact neighbors as well as *all* the neighbors for estimating the class entropy near each point. A further code optimization reduces also the time spent for querying the index. While the NN search on the projected candidates has to be done whenever the subset of features changes, the (full-dimensional) candidate neighbors for each point in the dataset do not change and can be retrieved only once. If each candidate point is projected onto $i+1$ features for $i = 0, 1, \dots, s-1$, the NN search for one point is still $O((i+1)N^{1/(1+\epsilon)})$, with a total cost for solving the A-ϵk-NN problems equal to $O(N^\alpha Ds^2)$. Accordingly, the complexity of the modified procedure is:

$$\mathcal{C}_{\text{SNEFS}} = O(LD + NLD + N^\alpha Ds^2) = O(Ds^2N^\alpha). \quad (12)$$

This modification usually gives better results in terms of classification accuracy and, even though the asymptotic complexity remains equal to that of NEFS, it makes the actual running time reduce significantly, as shown in the next Section.

V. Experiments

In this section, we report some experimental results[1] obtained by comparing our method, with the two different implementations NEFS and SNEFS, to RELIEFF, X-MIFS, RRPC and mRMR-MIFS. The comparison is related to the classification accuracy of models learned by using the features selected by the different algorithms. For a fair comparison, we use the RRPC method in its original semi-supervised version (RRPC_SS), with 80% of (artificially) unlabeled data, as well as in a completely supervised version (RRPC_S). As concerns mRMR-MIFS, we fix $\beta = \frac{1}{|S|}$ to be consistent with the choice made by RRPC in (7). For each algorithm and dataset, we used:

- a Random Forest (RF) of 20 trees, with *bootstrap* and the *Information Gain* split criterion;
- a Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel;
- a k-Nearest Neighbors classifier (kNN) with $k = 4$;
- a Multi-Layer Perceptron (MLP) with one hidden layer of 20 neurons, the tanh activation function and *early stopping*.

We used the implementation provided by the Python library *scikit-learn* [35]. The method used for the optimization of MLP weights is *stochastic gradient descent* (SGD), with

[1]All the experimental results are available as supplementary material at http://lion.disi.unitn.it/~mariello/nefs/ (accessed Apr. 2018)

an adaptive learning rate to reduce convergence time. The technique provided by the library is similar to the *bold driver* method [36], with the addition of early stopping (ES) to avoid overfitting. At each iteration, 20% of the training set associated with one fold of cross validation is kept apart as an ES-validation set. Starting from an initial value of 1, the learning rate is kept constant as long as the training loss keeps decreasing. Each time two consecutive epochs fail to decrease the training loss or to increase the classification accuracy on the ES-validation set by a tolerance value, the current learning rate is divided by 5. The highest accuracy is reached by stopping after 100 iterations with an improvement in the validation score less than 10^{-8}.

The tests were conducted on a 64-bit 8-core machine with a CPU frequency of 2.3 GHz and a total of 4 GB of RAM. Only a sequential implementation of all the algorithms was tested. In particular, we developed our own code for NEFS, SNEFS, RELIEFF, RRPC and mRMR-MIFS, while we used the original implementation of X-MIFS developed by the authors in [19].

Several benchmarks are taken directly from the UCI Machine Learning Repository [37]. Two other datasets (SPAMBASE_N and SEGMENT_N) are modified versions of the same datasets in [37], taken from the KEEL dataset repository [38]. The modified versions add a 20% noise level to the features according to the schema proposed in [39].

We also use the CORRAL synthetic dataset [40], which includes 6 features (A0, A1, B0, B1, I and C). The class is defined by $(A0 \wedge A1) \vee (B0 \wedge B1)$. I is randomly generated, and C is highly correlated with the class, with 25% of error rate. Moreover, another dataset (HYPERSPHERES) was generated to test the algorithms against highly unbalanced binary classes, defined by non-linear relationships with a small subset of features (7 among 100). We generated 100-dimensional samples normally distributed (with zero mean and unit variance) in the space $[-10, 10]^{100}$. A class 1 was then assigned to samples satisfying at least one of the following:

$$((X_6^2 + X_{20}^2 + X_{53}^2 + X_{22}^2 + X_{87}^2) \leq 100) \quad \vee$$
$$(((X_{10} - 8)^2 + (X_{44} + 3)^2 + (X_{53} - 5)^2) \leq 25),$$

where X_i is the value of the i-th feature of the sample X. With this choice of parameters and features, among 5000 generated samples we obtained nearly 1000 belonging to class 1, which is represented by the union of points in two hyperspheres (5D and 3D) with non-empty intersection. In this way one is able to assess the behavior of the feature selection techniques in the case of (known) complex dependencies between the features and the class, within a very noisy context. As a case study of $D \gg N$ and N small, we also consider the SMK dataset [41]. This represents a worst-case scenario for our techniques and RELIEFF, since they do not make use of all the samples for each iteration and therefore can be seriously affected by the scarcity of data. Finally, we compare the algorithms also on the USPS dataset for handwritten digit recognition [42].

For each dataset, we removed the samples with missing values and we transformed categorical variables into one-hot encoded variables. The final number of samples, features and classes for each dataset are reported in Table I.

TABLE I
DATASETS USED FOR THE TESTS

Name	N	D	N_c
ABALONE	4177	10	28
ARRHYTHMIA	452	274	13
CORRAL	128	6	2
GAS	13910	128	6
HYPERSPHERES	5000	100	2
SEGMENT_N	2310	19	7
SMK	187	19993	2
SPAMBASE_N	4597	57	2
USPS	9298	256	10

For the tests on the HYPERSPHERES dataset, the classifiers are trained on 7 features, 4 for CORRAL, while for the other datasets 10 features are used. In this way one can see the ability of the algorithms in selecting the best features as soon as possible, independently from the highest level of accuracy that they can reach with a customized threshold. Despite the fact of selecting all the features, for ABALONE we make the algorithms run without stopping prematurely. In each case all the classifiers are validated by using a 10-fold cross validation. The number of neighbors k was fixed to 4 to obtain a more precise approximation of the probability distributions, and the number of iterations of RELIEFF was set to N/k to make it comparable to our technique, after the description in Section IV-A. For each dataset and feature subset we assess the statistical significance of the results by using the Friedman test [43]. In the following, our findings are expressed in terms of averages and standard errors and, unless otherwise noted, they are statistically significant with a significance level $\alpha = 0.01$.

For ABALONE there is no significant difference among the algorithms, which reach always a poor accuracy level (≈ 0.3) because of the scarcity of samples in relation to the high number of classes. For ARRHYTHMIA, X-MIFS and mRMR-MIFS tend to perform slightly better than (S)NEFS with more than 5 features, while RRPC and RELIEFF achieve slightly worse results (see the supplementary material).

In the comparison with CORRAL, the algorithms do not present statistically significant differences in the average accuracy because of the very limited set of samples. However, the techniques based on the nearest neighbors (NEFS, SNEFS and RELIEFF) are able to select the 4 most relevant features most of the times, while the others tend to select the less relevant feature C as the first one (see Table II).

In the case of GAS, the convergence rate is usually higher for SNEFS. As an example, for the MLP there is an increase in accuracy of 5.4% with respect to X-MIFS with 4 features (see Figure 4a).

In the case of spam detection, the best classification accuracy ($\approx$0.85) is reached by the RF trained on the features selected by NEFS and SNEFS, as reported in Figure 4b. This accounts for approximately a 3.7% increase with respect to RELIEFF, 14.9% compared to X-MIFS, and 25% for RRPC and mRMR-

TABLE II
MOST FREQUENTLY SELECTED FEATURES FOR THE CORRAL DATASET.

Method	Rank			
	1st	2nd	3rd	4th
X-MIFS	C	A1	B1	B0
RELIEFF	B0	B1	A1	A0
SNEFS	B0	B1	A1	A0
NEFS	B0	A1	B1	A0
RRPC_S	C	B0	B1	A1
RRPC_SS	C	B1	A1	A0
mRMR	C	B0	B1	A1

MIFS. Note that our techniques have better results even with one or two features. We also compare the algorithms on random subsets of SPAMBASE_N, to show the robustness to data scarcity of our NN-based estimators. As expected, the estimation error is higher for all the compared algorithms and the classification accuracy is slightly lower. This is due to the reduced amount of samples, which leads to higher variance and generalization error. However, our techniques still lead to generally better results, as reported in Figure 4c for $N = D$.

For SEGMENT_N, NEFS and RELIEFF are not as robust as with spam detection, especially with fewer features (see Figure 5a). SNEFS instead leads to better generalization and less overfitting, because it limits the construction of the index to the set of all the features.

For HYPERSPHERES (see Figure 5b), none of the algorithms is able to select all the 7 features used to generate the class. However, it is evident that SNEFS and NEFS reach significantly higher values of accuracy, since they are able to select 5 or 6 of the features needed in most cases. RELIEFF and RRPC are clearly the worst, since they are not able to go further than 1 or 2 *relevant* features. It is also worth noting that the third feature selected by X-MIFS and mRMR-MIFS corresponds most of the times to the 53rd feature in the dataset, which is at the intersection between the two hyperspheres. This explains why X-MIFS and mRMR-MIFS are slightly better than the others with the first 3 features. However, the effect of the 93 noisy features becomes evident with more than 3 selected features and those methods lead to a classification accuracy up to 12.3% less than NEFS.

For USPS, X-MIFS achieves the best accuracy, followed by SNEFS and mRMR-MIFS (see Figure 5c). RELIEFF and RRPC provide worse results. This confirms the robustness of our techniques, which provide consistent results across different domains by combining the features of NN-based methods as well as MI-based methods.

For SMK, the scarcity of samples makes the estimators used by all the methods suffer from higher variance, thus leading to lower discriminative power and feature subsets that do not differ significantly from each other. This, coupled with classifiers that are not fine-tuned to the specific dataset, explains also the relatively low classification accuracy. However, for the kNN the highest level of accuracy is reached again by X-MIFS and SNEFS with 5 features (see Figure 5d).

TABLE III
ASYMPTOTIC TIME COMPLEXITIES.

NEFS/SNEFS	RELIEFF	X-MIFS	RRPC_S/mRMR
$O(DN^{\alpha}s^2)$	$O(DNT)$	$O(DNs^2)$	$O(DNs^2)$

A. CPU Time Models

To compare the algorithms in terms of the CPU time, let us start from the complexity analysis of Section IV and extend it to RELIEFF, X-MIFS and RRPC by considering the asymptotic cost models with the explicit indication of the dependency on the number of selected features s (whenever possible), since most of the experiments used $s \ll D$.

As shown in [16], the cost of RELIEFF for T iterations is:

$$\mathcal{C}_{\text{RELIEFF}} = O(DNT). \tag{13}$$

The X-MIFS algorithm follows the same greedy procedure of NEFS. The only difference is the evaluation of the MI instead of the NE. To compute the MI the procedure uses a modified version of (3), in which the probability distribution functions are replaced by the estimators based on multi-dimensional histograms. The implementation of X-MIFS includes the binarization of the features, so $|\mathcal{D}_{\mathcal{S}}| = 2^{|\mathcal{S}|}$. For each subset, the procedure estimates the joint and marginal probability distributions through a linear traversal on all the points. The summation in (3) is also linear in $2^{|\mathcal{S}|}N_c$. Therefore, the worst-case time complexity of X-MIFS for selecting s features is:

$$\mathcal{C}_{\text{X-MIFS}} = O\left(\sum_{i=0}^{s-1}(D-i)[N(i+1) + 2^{i+1}N_c]\right). \tag{14}$$

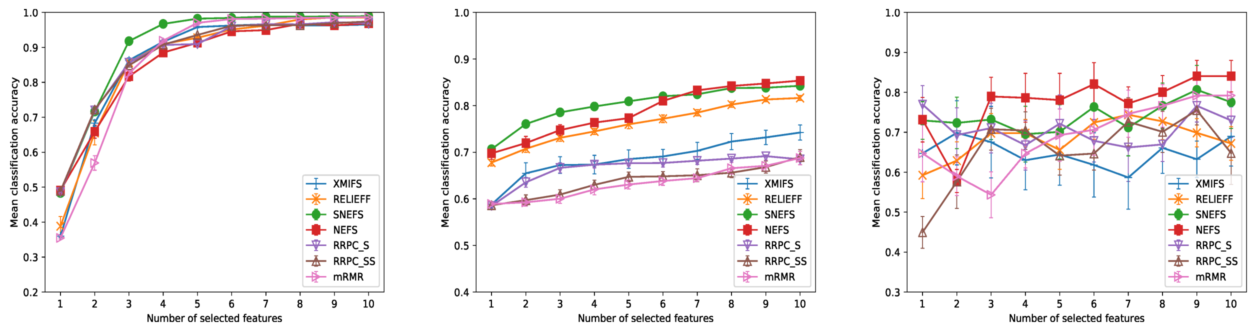
By using appropriate data structures to exploit the sparsity of the bins, as reported in the original work, and under the same hypotheses made in Section IV, the time spent on the traversal of the histograms can be reduced significantly, and the cost of X-MIFS becomes $O(Ds^2N)$.

For RRPC one can compute the correlation in linear time with respect to the number of samples. The worst case is the supervised scenario, in which the cost of computing the correlation is always $O(N)$. At each iteration, one needs $O(|\mathcal{S}|)$ operations to compute (7) and update the maximum. The complexity is then:

$$\mathcal{C}_{\text{RRPC\_S}} = O\left(\sum_{i=0}^{s-1}(D-i)iN\right) = O(Ds^2N). \tag{15}$$

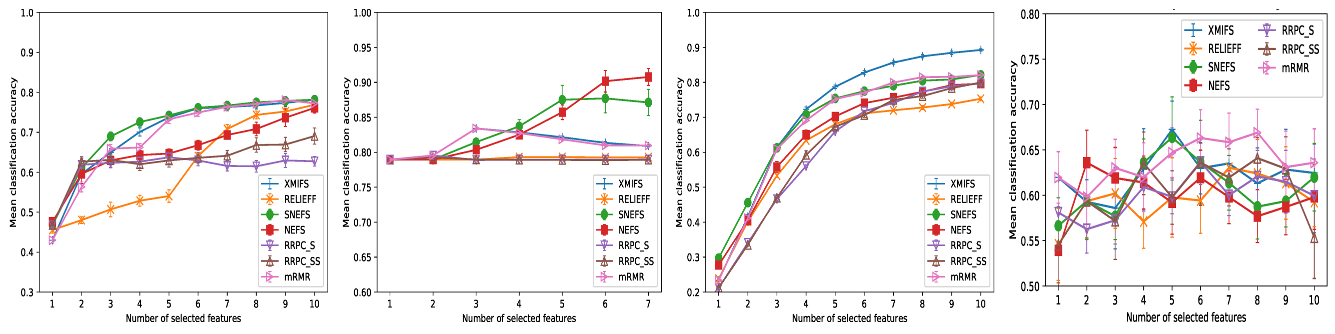
By using similar arguments, it can also be shown that the complexity of mRMR-MIFS is equal to that of RRPC. A summary of the asymptotic costs of all the algorithms is reported in Table III.

As concerns the scalability of the different implementations, in Table IV we report the least-square linear models of the running time with respect to the size of the dataset DN. s is fixed to 10, T to $N/4$ and ϵ to 15. It is evident that X-MIFS is the fastest. This can be explained if one considers that X-MIFS is applied on binary features and the current implementation is very efficient and uses bitwise operators. RELIEFF, RRPC



(a) (b) (c)

Fig. 4. The classification accuracy for the MLP on GAS (a), and for the RF on SPAMBASE_N (b) and on a small random sample of SPAMBASE_N (c).



(a) (b) (c) (d)

Fig. 5. The classification accuracy for the SVM on SEGMENT_N (a) and on HYPERSPHERES (b), for the RF on USPS (c), and for the kNN on SMK (d).

TABLE IV
CPU TIME LINEAR MODELS.

Method	Coefficient	Intercept
X-MIFS	4.8×10^{-7}	-0.144
RELIEFF	1.14×10^{-6}	1.21
SNEFS	1.02×10^{-5}	16.6
NEFS	2.61×10^{-4}	257
RRPC_S	3.8×10^{-6}	-0.521
RRPC_SS	3.2×10^{-6}	-0.489
mRMR	1.57×10^{-6}	-0.152

and mRMR-MIFS are also fast. NEFS is the slowest because of the cost paid for the construction of multiple indexes as well as the high frequency of nearest neighbors searches. RELIEFF instead updates the entire weights vector for each iteration, thus reducing the number of searches for the nearest neighbors. SNEFS is slower than most of the other methods but significantly faster than NEFS. Therefore, when robustness to noise and to class imbalance is a key requirement, SNEFS is a valid choice because it tends to provide better results in a reasonable amount of time.

VI. Conclusion

In this work we proposed a novel approach for feature selection based on the minimization of the Neighborhood Entropy, which corresponds to the maximization of the Mutual Information between the features and the output variable. The locally-optimal subset of features is selected by using a greedy procedure and the LSH index for nearest neighbors. We compared our algorithm with some of the most effective methods for selecting features based on MI, nearest neighbors and correlation. The experimental results show that our technique, in particular the SNEFS implementation, usually leads to superior classification accuracy, at the expense of more computation time. As an example, in the case of spam detection our techniques achieve an accuracy $\approx 25\%$ higher than that of recent algorithms based on the correlation coefficient. A better robustness to noise and to class imbalance is also empirically demonstrated. Moreover, in the context of non-linearities between the features and the output variable, SNEFS tends to select features with a better order, leading to a better classification accuracy for fewer features.

Acknowledgment

This work was supported by the University of Trento. The work of R. Battiti was supported by the Russian Science Foundation through the Project entitled Global Optimization, Supercomputing Computations, and Applications under Grant 15-11-30022.

References

[1] J. Kittler, “Feature selection and extraction,” *Handbook of pattern recognition and image processing*, pp. 59–83, 1986.

[2] R. Battiti and M. Brunato, *The LION way. Machine Learning* plus *Intelligent Optimization*. LIONlab, University of Trento, Italy, December 2017. [Online]. Available: http://intelligent-optimization.org/LIONbook/

[3] R. Wang, F. Nie, R. Hong, X. Chang, X. Yang, and W. Yu, "Fast and orthogonal locality preserving projections for dimensionality reduction," *IEEE Trans. Image Process.*, vol. 26, no. 10, pp. 5019–5030, 2017.

[4] R. Wang, F. Nie, X. Yang, F. Gao, and M. Yao, "Robust 2DPCA with non-greedy l1-norm maximization for image analysis," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1108–1112, 2015.

[5] D. Bouzas, N. Arvanitopoulos, and A. Tefas, "Graph embedded non-parametric mutual information for supervised dimensionality reduction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 951–963, 2015.

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

[7] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. Series B Stat. Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

[8] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint l2,1-norms minimization," in *Adv. Neural Inf. Process. Syst. 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1813–1821.

[9] C. Hou, F. Nie, X. Li, D. Yi, and Y. Wu, "Joint embedding learning and sparse regression: A framework for unsupervised feature selection," *IEEE Trans. Cybern.*, vol. 44, no. 6, pp. 793–804, 2014.

[10] S. Xiang, F. Nie, G. Meng, C. Pan, and C. Zhang, "Discriminative least squares regression for multiclass classification and feature selection," *IEEE Trans. Neural Netw.*, vol. 23, no. 11, pp. 1738–1754, 2012.

[11] M. A. Hall and L. A. Smith, "Feature subset selection: a correlation based filter approach," in *Proc. Intl. Conf. Neural Inform. Processing Intell. Inform. Syst.* Springer, 1997, pp. 855–858.

[12] J. Xu, B. Tang, H. He, and H. Man, "Semisupervised feature selection based on relevance and redundancy criteria," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 9, pp. 1974–1984, 2017.

[13] P. Jafari and F. Azuaje, "An assessment of recently published gene expression data analyses: reporting experimental design and statistical factors," *BMC Med. Inform. Decis. Mak.*, vol. 6, no. 1, p. 27, 2006.

[14] H. Tao, C. Hou, F. Nie, Y. Jiao, and D. Yi, "Effective discriminative feature selection with nontrivial solution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 796–808, 2016.

[15] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proc. 9th Intl. Workshop on Machine learning*, 1992, pp. 249–256.

[16] I. Kononenko, E. Šimec, and M. Robnik-Šikonja, "Overcoming the myopia of inductive learning algorithms with relieff," *Appl. Intell.*, vol. 7, no. 1, pp. 39–55, 1997.

[17] F. Nie, S. Xiang, Y. Jia, C. Zhang, and S. Yan, "Trace ratio criterion for feature selection." in *AAAI*, vol. 2, 2008, pp. 671–676.

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

[19] M. Brunato and R. Battiti, "X-mifs: Exact mutual information for feature selection," in *Proc. Intl. Joint Conf. on Neural Netw.*, 2016, pp. 3469–3476.

[20] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 189–201, 2009.

[21] K. Torkkola, "Feature extraction by non-parametric mutual information maximization," *J. Mach. Learn. Res.*, vol. 3, no. 3, pp. 1415–1438, 2003.

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

[23] R. Fano and D. Hawkins, "Transmission of information," *Am. J. Phys.*, vol. 29, no. 11, pp. 793–794, 1961.

[24] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, 2005.

[25] N. Kwak and C.-H. Choi, "Input feature selection by mutual information based on parzen window," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 12, pp. 1667–1671, 2002.

[26] L. Lefakis and F. Fleuret, "Jointly informative feature selection." in *AISTATS*, 2014, pp. 567–575.

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

[28] K. L. Clarkson, "Fast algorithms for the all nearest neighbors problem," in *IEEE Symp. on Foundations of Computer Science*, 1983, pp. 226–232.

[29] P. M. Vaidya, "An o(nlogn) algorithm for the all-nearest-neighbors problem," *Disc. Comp. Geom.*, vol. 4, no. 2, pp. 101–115, 1989.

[30] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," *Proc. 25th VLDB Conf.*, 1999.

[31] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, 2014.

[32] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *CVPR IEEE Conf.*, 2008, pp. 1–8.

[33] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *J. ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[34] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th annual symposium on Computational geometry*. ACM, 2004, pp. 253–262.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, no. Oct, pp. 2825–2830, 2011, http://scikit-learn.org.

[36] R. Battiti, "Accelerated backpropagation learning: Two optimization methods," *Complex systems*, vol. 3, no. 4, pp. 331–342, 1989.

[37] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[38] J. Alcalá, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Logic Soft Comput.*, vol. 17, no. 2-3, pp. 255–287, 2010, http://sci2s.ugr.es/keel/datasets.php [accessed Jan. 2018].

[39] X. Zhu, X. Wu, and Y. Yang, "Error detection and impact-sensitive instance ranking in noisy datasets," in *AAAI*, 2004, pp. 378–384.

[40] G. H. John, R. Kohavi, K. Pfleger *et al.*, "Irrelevant features and the subset selection problem," in *Machine learning: Proc. of the 11th Intl. Conf.*, 1994, pp. 121–129.

[41] A. Spira, J. E. Beane, V. Shah, K. Steiling, G. Liu, F. Schembri, S. Gilman, Y.-M. Dumas, P. Calner, P. Sebastiani *et al.*, "Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer," *Nature medicine*, vol. 13, no. 3, pp. 361–366, 2007.

[42] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, 1994.

[43] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Am. Stat. Assoc.*, vol. 32, no. 200, pp. 675–701, 1937.

Andrea Mariello (M'13) received an MSc in Computer Engineering cum laude from the University of Salento, Italy, in 2013, and he is currently pursuing a PhD at the ICT Doctoral School of the University of Trento, Italy. He worked as a computer scientist at the Euro-Mediterranean Center on Climate Change (CMCC) until 2015. In 2008, he was honored with the title of *Alfiere del Lavoro* by the President of the Italian Republic, being the first among the top 25 high school students in Italy in 2004-2008. He is a member of the LION Laboratory at the University of Trento and his research activity primarily focuses on machine learning and optimization, and in particular on feature selection and hyper-parameter optimization.

Roberto Battiti (F'09) received the Laurea degree in physics from the University of Trento, Italy, and the Ph.D. degree from the California Institute of Technology, Pasadena, CA, USA, in 1990. He is currently a Full Professor of Computer Science and the Director of the Machine Learning and Intelligent Optimization Laboratory with the University of Trento. He is intrigued by issues at the boundary between optimization and machine learning. He is the creator of Reactive Search Optimization, advocating the integration of machine learning techniques into search heuristics for solving complex optimization problems. He has authored over 100 scientific publications, including the recent book entitled *The LION Way*, which led to hundreds of applications in widely different fields.