An Efficient Mutual Information Based Feature Selection Method

Lari Lampén

Abstract

This paper investigates the application of the mutual information (MI) metric into feature subset selection in classification problems. MI has some significant advantages such as invariance under any injective transformation of variables. Several MI based feature selection schemes have been suggested in the literature, but they suffer either from an inability to account for complex feature interaction or from computational intractability in some conditions.

A new method is defined whereby computation of MI values for groups of features can be significantly speeded up by manipulating composite feature values, making it possible to compute groupwise MI values of sizes that were intractable with earlier methods. The new method is compared to previously published models and applied to a number of classification problems to illustrate its performance.

Keywords: Feature selection, feature ranking, mutual information.

1 Introduction

Feature selection is an important technique in many classification tasks[1] in pattern recognition. In feature selection, a subset of all available features is chosen so that the number of features left is as small as possible without degrading classification performance.

There are many reasons for wanting to do this: obviously, fewer features imply less need for storage space, faster processing and learning of data, and potentially the use of simpler classifier models. Some features may be tainted with noise to such an extent that they are actually detrimental to classification, and thus removal of such features increases accuracy.

The general problem of minimising the size of the data is called *dimensionality reduction*. Besides feature selection, the other main class of dimensionality reduction methods is *feature transformation*, also called feature extraction. As the name implies, such methods map the set of initial features into a space of lower dimension.

Feature transformation methods include singular value decomposition and principal component analysis [1], in which the data are assumed to lie on a nearly flat manifold; locally linear embedding [2], which considers curved manifolds locally flat around each data point; spectral domain methods such

[1]Feature selection is also useful for regression tasks, but they are beyond the scope of this paper.

as wavelet transforms and spectral decomposition [3]; and genetic algorithms [4]. They are most useful when it is known that the data points actually lie on a lower-dimensional manifold in the feature space, since the coordinates of the data points on the manifold can be computed and used as a new, smaller set of features, both to reduce storage space and in the hope that the shape of the manifold will offer some insight into the nature of the problem.

However, feature selectors have certain advantages over the more intricate feature transformation methods: the addition of new data points does not require reprocessing of all data, and if an initial set of data points is used to assess the usefulness of features before the whole data set is collected, one can decide to record only the features that are actually required for classification, saving time and effort that would otherwise have been used collecting useless measurements. For example in the field of medical diagnosis, collecting the data by making the necessary measurements and observations may be costly or risky and cause the patient discomfort; thereby, not collecting unnecessary features translates directly to cost efficiency and positive user experience.

In principle, feature selection and discretisation of continuous variables can be viewed as dimensionality reduction methods operating in orthogonal dimensions [5].

1.1 Feature Selection Methods

Feature selection is a complex endeavour due to interactions between features, which make it difficult to find a maximally useful subset of features (assuming that, as in most realistic problems, an exhaustive evaluation of all combinations is impractical). Feature ranking—the ordering of features by their salience (importance to classification)—is an even more complicated problem because of the non-monotonicity of the ranking: generally the ordering is not unique, and the best subset of n variables does not always contain the best subset of m variables, where $m < n$.

Suppose, for example, that one wishes to construct a system to predict the movements of the foreign exchange market based on historical currency rate data, and the three currency exchange rates GBP/EUR, USD/EUR, and GBP/USD occur as features in the data. It is clear that out of the three rates, one is redundant since it can be inferred from the others (assuming the data are noiseless). The choice of which one to discard is arbitrary, resulting in three possible equally valid solutions to the feature selection problem.

In more realistic cases, the relationship between features may be unknown. Continuing the previous example, consider the movements of the South Korean won (KRW) and the Japanese yen (JPY) against the US dollar. If the yen, say, suddenly falls by a significant amount, the won is likely

to eventually follow it, since the export industries of the two countries are direct competitors and a currency depreciation in one country will give its export industry an unsustainable edge in international trade. Thus there is a degree of interdependency in the rates of the two currencies, but it is highly nonlinear, unpredictable and noisy due to the extreme complexity of the generating process. Determining the salience of the USD/JPY rate for predicting the price of the won is therefore a highly nontrivial task.

The feature selection process has three components: a feature evaluation criterion for assessing the importance of feature subsets, a search algorithm for exploring the space of possible feature combinations, and a stopping criterion. Numerous search algorithms have been proposed in the literature for feature selection and ranking. *Exhaustive* methods are impossible to realise for problems of realistic size. *Heuristic* search algorithms can be divided into two main groups: *randomised* algorithms incorporating a nondeterministic element, for example genetic algorithms; and *sequential* algorithms.

A brief historical review of the development of feature selection methods is contained in [6]. Kudo and Sklansky [7] provide an experimental comparison of a variety of feature selection techniques.

Since a feature space of k features contains $2^k - 1$ non-empty subsets, exhaustive search of the feature space becomes intractable very quickly. The

branch and bound (B&B) method [8] reduces the search for monotonic feature evaluation criteria, but the monotonicity assumption is not satisfied in many practical scenarios. Genetic algorithms (GA) have been applied to the problem and found to reach semi-optimal performance in many cases without making assumptions of monotonicity [9]. However, Ferri et al. [10] note a number of problems in some current GA feature selection models, such as the introduction of extra parameters which have to be tuned to each problem, and the difficulty of determining a convergence criterion.

Sequential selection algorithms consider one feature at a time (or, in the case of extended methods, n features at a time). Forward sequential selection (FSS) starts with an empty set and adds features one by one, choosing the one that, together with the others already chosen, results in the best classification performance. Backward sequential selection (BSS) starts with the full set of features and eliminates features one at a time. A generalisation of forward and backward methods is the "plus l – take away r" method, which alternates l forward selections with r backward deletions. Sequential methods are greedy algorithms that can fail to find the optimal solution because of the non-monotonicity of the ranking of features [11].

Floating search [12] adds to sequential search the flexibility of reconsidering already selected or deleted features. It is similar to the "plus l – take away r" method, except that instead of having the fixed parameters l and

r, the sequence of selections and deletions is adaptive. Sequential floating forward selection (SFFS) and sequential floating backward selection (SFBS) are natural extensions of the basic sequential methods. In an experimental comparison, Kudo and Sklansky [7] found that floating search represents a dramatic improvement over simple sequential selection: its accuracy on par with methods based on genetic algorithms, and its complexity is still on a manageable level.

In terms of feature evaluation criteria, a natural starting point is of course classification accuracy (or, more accurately, probability of correct classification), which is what should be maximised in the end anyway. This can be done either by actually evaluating the classifier output with each feature set or by deriving simpler methods that are essentially dependent on the classifier model used. Methods that use the actual classifier to evaluate subsets are called *wrapper* methods, while algorithms that are classifier-independent are called *filter* methods [6].

A typical wrapper technique is clamping [13], where the idea is that a single feature can be "clamped" by resetting feature values to their mean, and then the modified data can be used to train and test the classifier system. Since the value of the clamped feature is equal for all data points, it has no discriminatory power and will have no effect on classification, and

thus its salience can be determined by the resultant decrease in classification accuracy.

In the form it was originally suggested in, clamping was only applied in connection with a sequential search using multilayer perceptron (MLP) classifiers. The ranking it produces was found to be largely independent of the parameters of the MLP. Thus a simple sub-optimal MLP can be used to rank the features before applying the "main" classifier. Extension to other search algorithms and classifier types is problematic for reasons of complexity, in particular when the classifier is computationally intensive (for example, a multiple classifier system).

1.2 Pairwise Mutual Information Methods

Filter methods have the advantage of generality, and often also speed, while still generally achieving good results. Methods based on mutual information are particularly interesting. See section 2.1 for a mathematical definition of mutual information. Intuitively, mutual information is a measure similar to correlation but extended to nonlinear cases. The mutual information of random variables U and V can be interpreted as a measure of the decrease in uncertainty about V once U is known. A key advantage of mutual information is its invariance under a variety of variable transformations.

Battiti [14] developed MIFS, a feature selection algorithm that uses mutual information as the feature evaluation criterion based on the intuitive idea that features that exhibit a high degree of mutual information contain redundancy. MIFS uses a greedy pairwise evaluation of features, which unfortunately severely limits its usefulness in cases with complex feature interaction.

MIFS uses the mutual information between a feature and a vector of correct class labels to evaluate salience of individual features, and the mutual information between two features to assess the amount of redundancy they contain. However, the limits of such pairwise computation soon become apparent. For example, in the case mentioned earlier where the three currency rates GBP/EUR, USD/EUR, and GBP/USD occur as features, a pairwise evaluation of redundancy, no matter how intricate, will fail to detect the complete redundancy of one feature, since no pair of these contains a significant amount of redundancy.

Battiti attempted to compensate for this deficiency by the inclusion of a correcting term in the equation: in MIFS, the sum of all pairwise feature redundancies multiplied by the parameter β is subtracted from the class-feature mutual information to obtain a heuristic approximation of usefulness. However, this approach is too simple for most classification problems, and since Battiti provided no systematic procedure for choosing β, the parameter

may need to be tweaked for each problem, complicating the feature selection process.

MIFS-U [15] is an improved version of MIFS that works well when information is evenly distributed across the features, but being based on pairwise feature evaluation, the algorithm suffers from the same basic limitations as the original MIFS.

MIEF [16] is a more intricate extension of MIFS that makes use of mutual information between a set of two features and the classification vector—essentially computing mutual information in triplets. This alleviates the problem of handling complex feature interactions but does not solve it. Moreover, MIEF exacerbates MIFS's problem of the β parameter by including a total of three free parameters with no systematic procedure to determine their values (although the authors suggest values that are "appropriate for many classification tasks").

To account for complex feature redundancies, an algorithm must be able to consider interaction between groups of features, but such schemes have received little attention because they have been seen as impractical. For example, Kwak and Choi [15] outline such a method in passing, only to brush it away as impossible to realise in practice.

1.3 MIFSFS and MIBSFS

Partridge and Cang [17] introduced greedy search methods (called MIFSFS and MIBSFS for forward and backward search, respectively) that extend Battiti's technique with full-blown mutual information computation between sets of features. They found that their extension greatly improved on Battiti's technique, producing semi-optimal results in most cases.

The MIFSFS algorithm is presented below. It operates with two sets: the features are moved one by one from the initial set F to the set of selected features S. The ascending order of transfer to set S produces a ranking of the features. Selection of the optimal feature space is then easy by choosing the top-ranked subset $F^{(m)}$ that maximises $I(C; F^{(m)})$, the mutual information with the class vector. A subset of specific size can also be chosen if required (for example due to storage space considerations).

1. $F \leftarrow$ initial set of features

   $S \leftarrow \emptyset$

2. For each $f \in F$, compute $I(C; f)$

3. Choose the f the maximises $I(C; f)$

   $F \leftarrow F \setminus f$

   $S \leftarrow f$

4. Repeat until $F = \emptyset$

   · For each $f \in F$, compute $I(C; S \cup f)$

   · Choose the f the maximises $I(C; S \cup f)$

   · $F \leftarrow F \setminus f$

   · $S \leftarrow S \cup f$

Applying both the forward and the backward search variants of the algorithm to a data set is suggested as a good way to detect feature interactions, which are evinced by a difference in ranking or subset selection between the two search types, which can then be used as a basis for further analysis of such interactions.

If the contribution of a single feature to the group-class mutual information is very small (i.e. $I(C; S) \approx I(C; S \cup f)$), the feature can be considered to be signalled as irrelevant or fully redundant by the feature selector.

Note that the algorithm makes use of $I(C; S \cup f)$, a mutual information value between the class vector and a set of features. While some authors (such as Al-Ani et al. [16]) define mutual information directly for more than two variables, Partridge and Cang instead compute it using a process called feature composition, a method of combining several features into one, whereby groupwise mutual information calculation can be reduced to the two-variable case. The two methods are equivalent in terms of complexity. Feature com-

position is further discussed in section 2.2.

The main shortcoming of MIFSFS is its limited scalability to problems with feature spaces of nontrivial size: the feature composition method used is susceptible to an exponential explosion of complexity, since the number of possible combinations grows exponentially as the number of features being considered increases. While Partridge and Cang note that combinations that do not actually occur in the data can obviously be omitted, the way they perform the omission still leaves the algorithm impractical for large feature spaces.

The difficulty arises in computing the mutual information of large groups of features, a problem that is discussed in detail in the next section.

2 Mathematical Preliminaries

This section defines some key concepts mathematically.

A classification task can be seen as a search for a mapping from data to a set of class labels. For a problem with n data points and k features, the set of all feature values can be written as a matrix M of size $n \times k$, where the value v_{ij} measures the jth feature for the ith data point. A classification then takes the form of a mapping from M to a vector of class labels:

$$\begin{pmatrix} v_{11} & v_{12} & \dots & v_{1k} \\ v_{21} & v_{22} & \dots & v_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ v_{n1} & v_{n2} & \dots & v_{nk} \end{pmatrix} \mapsto \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

In matrix M, the ith row vector $d_i = (v_{i1}, v_{i2}, \dots, v_{ik})$ contains all measurements related to the ith data point. Similarly, the jth column vector $f_j = (v_{1j}, v_{2j}, \dots, v_{nj})^T$ corresponds to the jth feature.

2.1 Mutual Information

All mutual information based feature selection schemes call for the computation of mutual information values using probability density functions, which in real-world problems have to be derived from data. Since approximation of these density functions using continuous functions is nontrivial (involving an integral which cannot be solved explicitly), the chosen approach is to

first discretise the feature values using any of a variety of existing discretisation techniques for conversion of continuous feature values. In the following, discussion is therefore confined in the domain of discrete random variables.[2]

The mutual information $I(U, V)$ between two discrete random variables U and V is defined as

$$I(U,V) = \sum_u \sum_v p(u,v) \log \frac{p(u,v)}{p(u)p(v)},$$

where $p(u), p(v)$ are density functions and $p(u, v)$ is a joint density function.

As mentioned above, a key advantage of mutual information is its invariance under many transformations. The following theorem states this property in more exact terms:

> **Theorem.** Assume that g is an injective[3] function. Then for any random variables U and V, it holds that $I(g(U), V) = I(U, V)$.

To outline a proof, note that the mutual information value is affected only by the frequency of occurrence of each value in U and not by the values themselves. Since g is an injection, it cannot change these occurrence counts.

[2]These methods can also be applied to nominal-valued features, which can be encoded numerically.

[3]An injective function, also called one-to-one function, maps different values in the domain to different values in the range, i.e. the function g is an injection if and only if $\forall a, b : (g(a) = g(b) \Rightarrow a = b)$.

2.2 Feature Composition

The above definition only covers mutual information between two random variables. The mutual information between two groups of features can be computed by creating a composite feature for each group, reducing the problem to that of two features. The central problem is then one of efficiently constructing a composite feature from a subset of the original feature set.

A composite feature basically consists of combinations of values of the simple features in it. The following matrix shows features f_1 and f_2 on the left side and their composite feature, which shall be denoted $F(f_1, f_2)$, on the right:

$$\left(\begin{array}{cc|c} v_{11} & v_{12} & (v_{11}, v_{12}) \\ v_{21} & v_{22} & (v_{21}, v_{22}) \\ \vdots & \vdots & \vdots \\ v_{n1} & v_{n2} & (v_{n1}, v_{n2}) \end{array}\right)$$

Thus a composite feature made up of k simple features consists of k-tuples of values corresponding to those of the component features; in the above case, $k = 2$ and the composite values are ordered pairs. For the purposes of composition, nested pairings can be considered equivalent to the corresponding "unrolled" list, i.e. $((a, b), c) = (a, (b, c)) = (a, b, c)$. It can easily be seen that with these definitions, the composition operation F is associative; that is, the composition of three features a, b, c can be done in

any order since $F(F(a,b),c) = F(a,F(b,c))$.

2.3 Complexity of the Feature Composition Process

Note that in the feature composition procedure outlined above, if f_1 has b_1 possible different values ("bins") and f_2 has b_2, then in principle there are $b_1 \cdot b_2$ possible values of $F(f_1, f_2)$. In general, for a composite feature made up of z features $b_1, b_2, \ldots, b_z$, the number of possible values is

$$B_c = \prod_{i=1}^{z} b_i,$$

and this number grows extremely fast (if the number of bins is the same for all features, B_c is an exponential function of n).

This very large number of possible values is the source of the intractability of MIFSFS and MIBSFS in large feature spaces. To compute the mutual information $I(U,V)$, the term $p(u,v)$ is estimated from the number of times each value occurs, and an obvious way to do this is by constructing an array where each element, corresponding to a possible value, counts the number of times it occurs. In the case when one of the features is a composite feature, the size of this array will be proportional to B_c, implying that storing the array or looping through it is impossible except in impractically small feature spaces.

Nevertheless, the composite feature of any number of features is still essentially a list of length n, as discussed above, and therefore it can contain at most n unique values. This indicates that as the number of simple features contained within a composite feature grows, the number of possible values and the size of the histogram array grow very fast, while the number of feature combinations that actually occur never exceeds a fixed ceiling. In short, the more features one combines, the sparser and more wasteful of memory the histogram array becomes.

3 The Proposed Method

The computation of mutual information of composite features can be greatly simplified by manipulating the value sets. The procedure is conceptually simple: a series of injective functions will be applied to the composite feature before constructing a histogram for computation of mutual information. Since mutual information is invariant under injections, the result values can be used to compute the mutual information of the original composite feature. The injections used are chosen specifically to ease the computational complexity and are defined as follows:

Assume that U is a composite feature made up of r simple features $f_1, f_2, \ldots, f_r$, and f_j has B_j bins. Define the function $b : \mathbb{R} \mapsto \mathbb{I}$ as a numbering of bins; that is, $b(x) = 0$ if and only if x is equal to the lowest value among the bins (for the feature in question), $b(y) = 1$ if y is equal to the second lowest, etc. Note that multiple occurrences of the same value map to the same number.

The composite feature U is a list of ordered r-tuplets, for which it is defined that $b(U) = b(F(f_1, f_2, \ldots, f_r) = F(b(f_1), b(f_2), \ldots, b(f_r))$, i.e. applying the function b to U means simply applying it to each of the simple features in it.

To illustrate this with an example, consider the following small example

matrix, which shows two features, their composite feature, and a composite of their labels according to the above-defined function b:

$$\left(\begin{array}{cc|c|c} f_1 & f_2 & F(f_1,f_2) & b(F(f_1,f_2)) \\ 3.0 & 6.0 & (3.0,6.0) & (3,2) \\ 1.0 & 1.0 & (1.0,1.0) & (1,1) \\ 2.5 & 6.5 & (2.5,6.5) & (2,3) \\ 1.0 & 1.0 & (1.0,1.0) & (1,1) \\ 0.5 & 0.5 & (0.5,0.5) & (0,0) \\ 3.0 & 9.0 & (3.0,9.0) & (3,4) \end{array}\right)$$

The features in the example contain multiple occurrences of values to emphasise the fact that these are discrete features with a limited number of possible values, or bins.

Define $B(f_1, f_2, \ldots, f_r) = \max_{1 \leq i \leq r} B_i$, i.e. the function B picks the largest number of bins from the given set of features. Moreover, define the function c

$$c : \mathbb{R}^r \mapsto \mathbb{N} : c(v_1, v_2, \ldots, v_r) = \sum_{i=1}^{r} b(v_i) \cdot B(v_1, v_2, \ldots, v_r)^i.$$

This function c is an injective mapping that associates a unique natural number with each possible value combination. In practice the resulting values are very sparse.

The function b is used again on the result of the previous stage, so that the resulting vector contains only nonnegative integers without any gaps.

Continuing the earlier example, the following matrix completes the transition from individual features to the simplified composite feature step by step:

$$\left(\begin{array}{cc|c|c|c} f_1 & f_2 & F(b(f_1), b(f_2)) & c(b(f_1), b(f_2)) & b(c(b(f_1), b(f_2))) \\ 3.0 & 6.0 & (3,2) & 3 \cdot 5^0 + 2 \cdot 5^1 = 13 & 3 \\ 1.0 & 1.0 & (1,1) & 1 \cdot 5^0 + 1 \cdot 5^1 = 6 & 2 \\ 2.5 & 6.5 & (2,3) & 2 \cdot 5^0 + 3 \cdot 5^1 = 17 & 4 \\ 1.0 & 1.0 & (1,1) & 1 \cdot 5^0 + 1 \cdot 5^1 = 6 & 2 \\ 0.5 & 0.5 & (0,0) & 0 \cdot 5^0 + 0 \cdot 5^1 = 0 & 1 \\ 3.0 & 9.0 & (3,4) & 3 \cdot 5^0 + 4 \cdot 5^1 = 23 & 5 \end{array}\right)$$

The advantage of the method can be seen from the above example: while in theory there are $B_1 \cdot B_2 = 4 \cdot 5 = 20$ possible values for the composite feature, there are actually only five unique values, and a histogram for the simplified composite feature will only need to contain five elements. The difference quickly becomes much larger as the number of features composed increases.

Fig. 1 shows another example, illustrating the transformation one step at a time.

Using the functions defined above, let $U' = d(c(U))$; thus, each value combination in U is mapped to a single natural number in U', as illustrated by the matrix above. The mutual information values will remain unchanged, i.e. $I(U', V) = I(U, V)$ by the Theorem in section 2.1, but building a histogram of the values in U' is fairly trivial, since the values are numbered using consecutive integers, and the largest value is equal to the number of

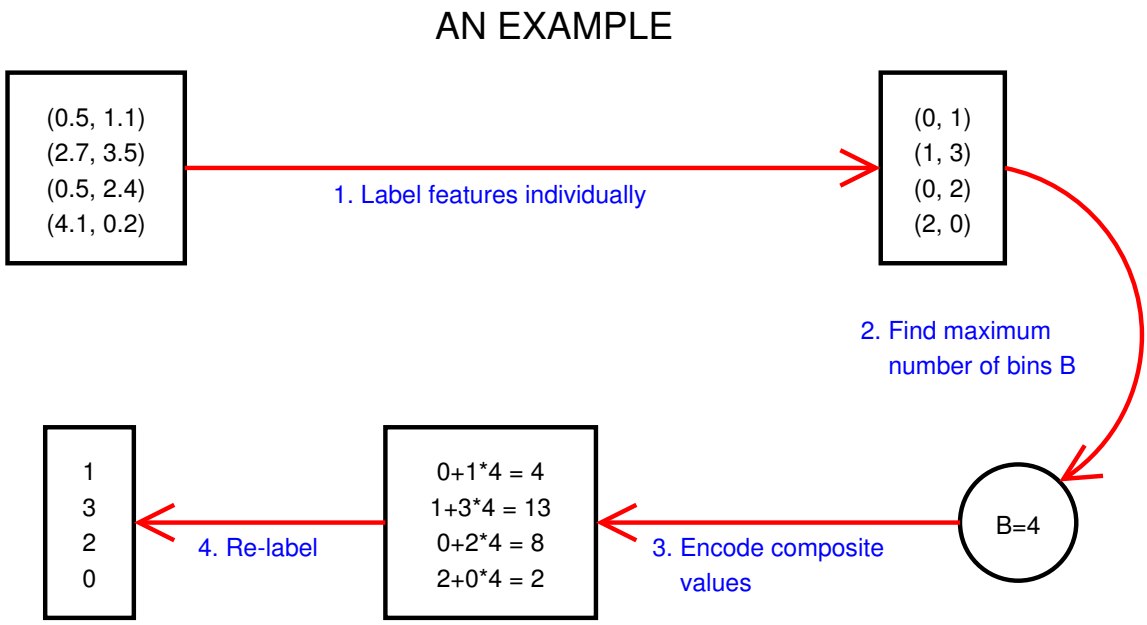

Figure 1: A step-by-step example of the method.

unique value combinations that actually occur in the training data. An upper limit for this is the number of data points n, which, especially in large feature spaces, is only a tiny fraction of the B_c combinations required by the naive histogram approach.

3.1 Complexity of the Proposed Method

The key question here is of course how the complexity of the method described above compares with that of the basic feature composition process when faced with the practical task of computing mutual information values for groups of features.

Assume that one wishes to compute the mutual information between a total of r features (since mutual information is a symmetric measure, the specific way in which the features are grouped does not affect the result) in a problem with n data points. To simplify the problem somewhat, assume that all features are binned with q bins; in other words, q is the number of unique values occurring in each feature.

The complexity of the basic composition process was discussed above in section 2.3. The total cost of computing the mutual information using it is $O(n + q^r)$, where, as r grows, n quickly becomes insignificant and the complexity function becomes exponential.

As for the new method, the bin labelling function b is computed separately for each of the r features and involves a sorting operation which can be implemented in $O(n \log n)$ time, resulting in a total complexity of $O(r \cdot n \log n)$. Finding the maximum number of bins B and the encoding function c both require time proportional to $O(rn)$, and finally, actually computing the mutual information requires a pass through the histogram, which has size not exceeding n, implying a complexity of $O(n)$.

Thus, the total complexity of the operation of computing the mutual information is of the order $O(r \cdot n \log n)$, which is sub-quadratic in terms of the number of data points, but more significantly, is linear in relation to the number of features being considered. Thus, this procedure is not a mere speed-up of existing methods; it makes it possible to apply the mutual information metric on a group evaluation basis to any feature space of realistic size, whereas earlier, feature spaces of any but impractically small size were intractable with such methods.[4]

[4]For example, the largest example Partridge and Cang [17] discuss involves just six features.

4 Results

This section presents various experimental results grouped by data set. These classification problems have been widely discussed in the literature. All of these data sets are available from the UC Irvine machine learning data repository.

4.1 The Monk3 Data Set

Since the algorithm described above is a new way to implement MIFSFS, not a heuristic approximation of it, it should produce the same results as MIFSFS implemented using the straightforward histogram method. Indeed, to verify that this is the case, an implementation of the proposed algorithm was applied to rank features in the Monk3 data set from the UCI data repository, since this data set was used as an example by Partridge & Cang [17].

The Monk3 data represent a binary classification task with six discrete-valued features (denoted $F_1, F_2, \ldots, F_6$), where the class label determines whether the logical expression $(F_5 = 3 \wedge F_4 = 1) \vee (F_5 \neq 4 \wedge F_2 \neq 3)$ holds. Note that features F_1, F_3 and F_6 do not appear in the expression and are thus completely irrelevant to the classification task. There are 122 data points in the set.

To summarise the results, the algorithm picks F_2 followed by F_5 and then

F_4 as the most salient features, as could be expected. After them, it picks features F_1, F_3 and F_6, all flagged as irrelevant or redundant. (In comparison, MIFS incorrectly chooses F_3 as the third most salient feature, whereas MIFS-U chooses the same ordering as MIFSFS.)

The results, including exact mutual information values, were identical to those reported by Partridge & Cang [17] for their implementation of MIFSFS. This serves as empirical confirmation of the correctness of the proposed method.

4.2 Ionosphere Data

The most interesting problems are those with a large feature space, since MIFSFS could not be applied to them before the present method. Any feature count in the double digits is already large in this sense.

In this experiment, MIFSFS was applied to the Johns Hopkins University's ionosphere data set from the UCI data repository. The data set contains radar measurements of free electrons in the ionosphere. There are 34 continuous features to be classified between two classes, and the total number of instances is 351. This data set is used because it has a feature space of suitable size and a "canonical" partitioning of data into training and testing sets. The latter is useful for reproducibility of results, as different partitions

tend to produce different feature rankings.

The methodology is as follows: the features were first ranked using MIFSFS, and then a single multilayer perceptron (MLP) classifier was trained and tested with each top subset, i.e. first with no features (resulting in chance level performance), then with the most salient feature, then with the two most salient features, etc. The MLP was trained and tested a total of ten times for each feature subspace, and the average classification accuracy with each subset was recorded.

The MLP had 50 hidden nodes, was trained for 50 cycles, and had learning rate and momentum both set to 0.0005. The features are already normalised in the original data set to have values in the range $[-1, 1]$; they were discretised using ten bins of equal width. The first 200 instances were used for training, the rest for testing. The results are shown in fig. 2. For comparison with pairwise methods, the result of MIFS (with $\beta = 1.0$) for the same data with the same parameters is also shown.

This type of plot should usually show a monotonic increase in accuracy, as features are added in, possibly followed by some decline, if some of the features are so noisy they degrade classification performance. The absence of a noticeable decline in this graph indicates that no features are so noisy that they would significantly confuse the MLP. On the other hand, a large number of features are redundant, since the accuracy $94.2\% \pm 2.5$ reached

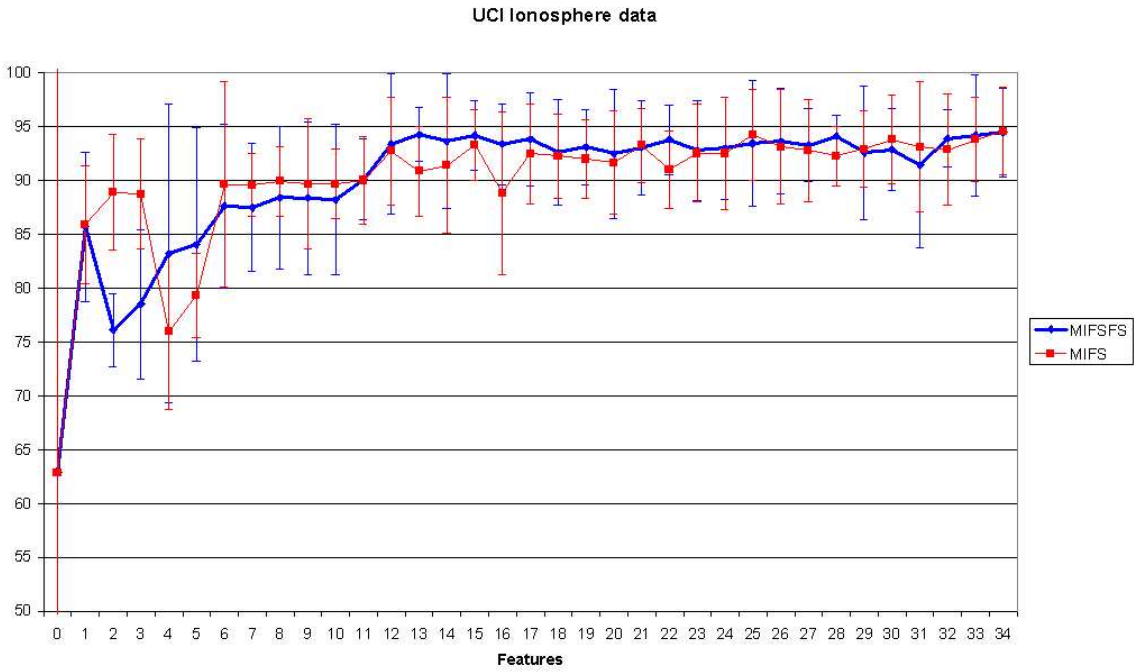

Figure 2: Results for the UCI ionosphere data set.

Ranking	1.	2.	3.	4.	5.	6.	7.
MIFSFS	F_5	F_6	F_8	F_9	F_{20}	F_{19}	F_{24}
MIFS	F_5	F_1	F_2	F_{12}	F_3	F_6	F_{27}

Table 1: The highest-ranking features in the ionosphere data as chosen by MIFSFS and MIFS.

with 13 features with MIFSFS is practically identical with the $94.4\% \pm 4.1$ with the full data set, suggesting that nearly two thirds of the features can be discarded without loss of accuracy.

Table 1 shows the features that are ranked the most salient by the two algorithms. While the algorithms agree that F_5 is the most salient feature (the algorithms always pick the same feature first, since in that case, the

setwise mutual information reduces to the pairwise case), not much agreement can be seen otherwise. This indicates that there is interaction in the data that is not picked up by pairwise evaluation. MIFSFS indicates just five features as a sufficient subset, tagging the rest as redundant, but clearly the MLP classifier benefits from the inclusion of some more features. Ruiz et al. [18] tested feature selection algorithms based on projection and correlation on this data set, and in their results, the number of features selected varied between 12–32, showing how comparatively aggressive MIFSFS is in marking features as irrelevant.

The classification accuracy using subsets chosen by MIFSFS essentially overlaps that of MIFS throughout, except for the smallest feature subsets. The choice of subset of size 2 is particularly surprising, as it shows an unusual dip that breaks the normal pattern of monotonic increase. On the other hand, there is a similar dip in the performance of MIFSFS at the subset of size 4. However, as table 1 shows, these dips do not correspond to the same feature. The cause of this phenomenon is unclear.

In the literature, Cantú-Paz [19] has used a genetic algorithm to select features in this data set. His algorithm chose 11.5 ± 2.95 features, for which a Naive Bayes classifier reached an accuracy of $91.50\% \pm 0.79$ in five-fold cross-validation. Yang and Honavar [9] used another genetic algorithm for feature selection and an artificial neural network based on a constructive learning

algorithm for classification. Their algorithm was able to choose a subset of 17.3 ± 3.5 features on which the classifier achieved an accuracy of $98.6\% \pm 2.4$ in 10-fold cross-validation. Actual feature rankings were not reported. These results are not directly comparable due to differences in data partitioning and classifier systems, but serve to give an idea of what has been achieved.

4.3 Wisconsin Diagnostic Breast Cancer Data

The Wisconsin Diagnostic Breast Cancer (WDBC) data set contains features computed from a digitised image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. There are 30 continuous features and two classes corresponding to diagnosis of the tumour as either benign or malignant. The number of instances is 569. The data were normalised for the experiment so that $\mu = 0.0$ and $\sigma = 1.0$ for each feature.

The basic set-up of the experiment is similar to the one with the ionosphere data: features were ranked with MIFSFS (and MIFS with $\beta = 1.0$, for comparison) and then a single MLP classifier was trained and tested with each feature subspace. The MLP parameters were the same as in the previous section. The features were discretised using ten bins of equal width. Partitioning was done using 10-fold cross-validation so that for each parti-

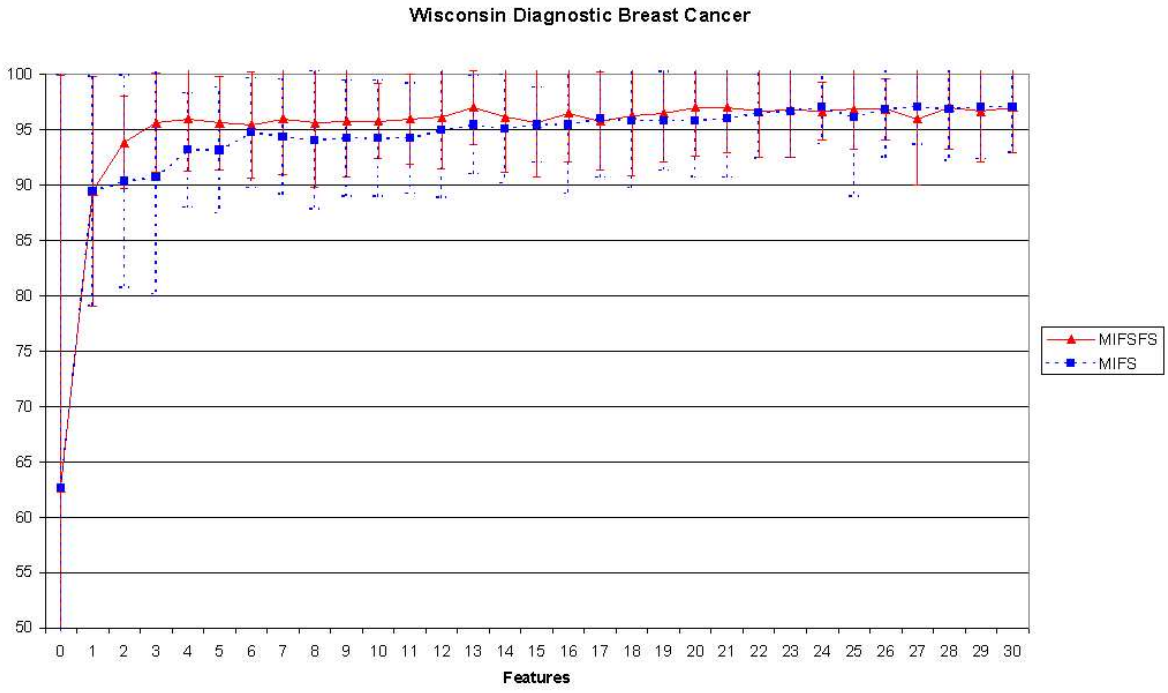

Figure 3: Results for the breast cancer diagnosis data set.

tion, only the training set was used to rank features. The results are shown in fig. 3.

Again, the difference between MIFSFS and MIFS is rather small. This data set contains a remarkable amount of redundancy: for example, MIFSFS reaches a classification accuracy of $95.6\% \pm 4.2$ with just three features, compared to an accuracy of $97.0\% \pm 4.1$ with the full data set. In other words, just one tenth of the original set of features produces results that are nearly on par with those with the full feature space. The redundancy is well documented: for example, Street, Wolberg and Mangasarian [20], using feature

transformation methods, found that a set of three features gave the best results.

Scherf and Brauer [21] developed a filter-type feature selection method that performs feature weighting based on Euclidean distances between instances and applied it to this problem. Even for the most aggressive parameter setting, their algorithm selected a subset of 20 features, on which an adaptive radial basis function network classifier achieved the same 98% accuracy as with the entire data set; no standard deviations were reported. Considering the amount of redundancy apparent in the data, their feature selector appears rather lax about accepting features as relevant.

4.4 Sonar Target Data

The sonar target classification data set contains measurements related to radar signals bounced off objects at various angles and under various conditions. It is a binary classification task: the object in question must be identified as either a rock or a metal cylinder (intended to simulate a mine). There are 111 "mine" patterns and 97 "rock" patterns. The number of features is 60. An even division into training and testing material is provided. The data is normalised in the range $[0.0, 1.0]$.

Kwak and Choi [15] report a comparison of feature selection methods

for this data set. Unfortunately, analysis of their results shows that their experiment has used the full data set for feature selection before partitioning the data. Since the test set has been available in the feature selection phase, the independence of the test set is compromised and the reliability of results is called into question.

Even accounting for this problem, their results were initially impossible to reproduce. However, in a later paper [22], the same authors report results that are identical for the relevant part, but a key parameter is reported differently (in [15], the learning rate of the classifier is stated to be 2.0, while in [22] it is reported as 0.2), while all other parameters are unchanged. It was therefore assumed that a there has been a typing mistake or other error, and that the latter value is correct. Of course, the reason of the discrepancy could be something else.

With these caveats in mind, the experiment was nevertheless modelled after that of Kwak and Choi to achieve comparability of results. The features were ranked using the *entire* data set. Optimal subsets of certain sizes were picked from this ranking, and an MLP classifier was trained and tested ten times for that feature subspace, with average performance recorded. The MLP had three hidden nodes and a learning rate of 0.2 with no momentum. It was trained for a maximum of 300 iterations.

Table 2 shows a comparison of the results and those reported by Kwak

Features	MIFSFS	MIFS	MIFS-U
3	**76.35** (14.95)	51.71 (2.1)	65.23 (1.6)
6	**78.27** (6.32)	74.81 (1.4)	77.03 (0.4)
9	74.04 (8.21)	76.45 (2.4)	**78.98** (0.7)
12	76.44 (8.70)	78.12 (1.8)	**81.51** (0.4)
All 60	87.92 (0.2)		

Table 2: Performance of MLP classifier on selected subsets on the Sonar data. Standard deviations given in parentheses. The highest accuracy on each row is highlighted. Figures for MIFS, MIFS-U and whole set extracted from [22].

Ranking	MIFSFS	MIFS ($\beta = 0.0$)	MIFS ($\beta = 1.0$)	MIFS-U ($\beta = 1.0$)
1.	F_{12}	F_{12}	F_{12}	F_{12}
2.	F_{16}	F_{11}	F_{51}	F_{11}
3.	F_{26}	F_{10}	F_{4}	F_{49}
4.	F_{22}	F_{13}	F_{40}	F_{36}
5.	F_{28}	F_{49}	F_{60}	F_{51}

Table 3: Five top-ranked features as selected by different feature selection methods for the sonar target data set.

and Choi. MIFSFS is strong when very few features are selected, but its performance seems to peak early on. MIFS performs particularly poorly with small feature subspaces. The standard deviation values reported by Kwak and Choi seem suspiciously low.

Table 3 tabulates the highest-ranking features. These are interesting in the context of fig. 4, which is a mutual information diagram for the data set, i.e. it shows the mutual information of each feature with the class vector. Since all of these feature selection algorithms are mutual information based forward selectors, it is natural that they all choose F_{12} as the most salient

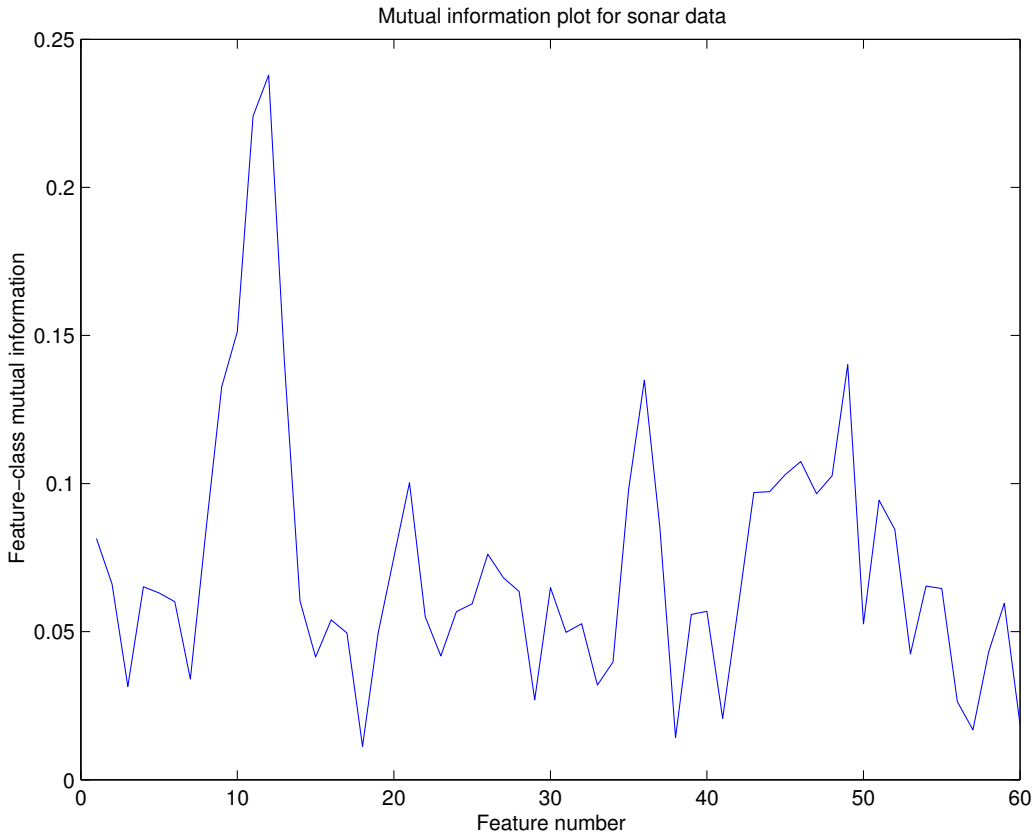

Figure 4: Mutual information diagram for the sonar target data set.

feature since it has maximal mutual information with the class. It is instructive to look at the rankings of MIFS with $\beta = 0.0$, which completely ignores feature redundancy: the four top-ranked features represent adjacent values near the peak in the graph. Since they correspond to a very short interval of time, any property of the target manifested in a different portion of the sonar signal is practically impossible to pick up based on these features.

In contrast, the other algorithms account for redundancy and pick values from a wider distribution in the interval. MIFS with $\beta = 1.0$ seems to overdo this at the expense of salience for classification, while MIFSFS and MIFS-U seek more realistic compromises between redundancy and salience.

These results suggest that MIFSFS can outperform its rivals for small feature subspaces, but as the feature subspace grows, differences in accuracy between different mutual information feature selectors become slim.

5 Discussion

This chapter notes problems with the current method and proposes a possible solution.

5.1 Limitations

Perhaps the most significant limitation of the MIFSFS feature selection method is its dependence on binning. Since the method is based on frequency counts of occurrences, it will not work well with features where values rarely occur repeatedly—typically when the number of possible values is large in comparison to the actual number of values (e.g. if it happens that the data consist of a thousand integers in the range 1...10,000 and all of them are different, all frequency counts will be either 0 or 1, producing poor results). Thus, it is not sufficient that the input data are discrete; they must in fact be binned using an appropriate number of bins.

Partridge & Cang [17] point out that increasing the number of bins used for discretising a single feature in relation to other features introduces a bias towards that feature. The cause of this phenomenon is easy to see in the context of the method proposed here: mutual information between groups of features is fundamentally a measure of their ability to discriminate between data points. The process of progressively including more and more features

in a set to discriminate between more data points is analogous to refining a set of decimal numbers with more and more significant digits to be able to tell apart the numbers: to discriminate two numbers that are equal up to the ith position, i significant digits must be used. If the number of possible values for the ith digit (or feature) is larger than for the others, there will be a natural bias towards it in the selection process as it is likely to have more discriminatory power.

It is natural that features are chosen that can be used to tell data points apart, but the bias is still an artefact of the discretisation process rather than a genuine property of the data. Probably the best simple approach is to use the same number of bins for all features, but this aspect of feature discretisation needs further research.

An additional problem with histogram-based estimation of probability densities is its susceptibility to the curse of dimensionality in large feature spaces. In high dimensions, binning is inaccurate [23]. However, the proposed method is not directly compatible with other probability density estimation methods, such as kernel density estimators.

5.2 Future Research

Choosing an appropriate number of bins is not all there is to discretisation. Numerous discretisation methods exist, including the standard equi-width histogram (where the bins correspond to intervals of equal size), equal frequency binning (where the bins have an equal number of data points), the MaxDiff method (which seeks to place bin boundaries in spots where there is a large difference between successive data points), and various fuzzy and iterative methods [24]. Which one produces the best classification results often depends both on the data and the classifier used.

There is initial evidence to suggest that mutual information could be used to assess discretisation methods. Consider fig. 5 and compare it with fig. 2 back on page 28. It plots the group mutual information with the class vector $I(C, S)$ as a function of the size of S for the ionosphere data set. A graph of this kind first displays a quick rising trend, as the most important features are added in, and then reaches a plateau, as subsequent features contain less and less additional information about the classification. Due to the monotonicity of mutual information, no decreases are possible in such graphs.

Preliminary results suggest that when the rate of increase of information contained in the subset is quick, i.e. the graph plateaus early on, information about the class is well "condensed" in the most important features, which

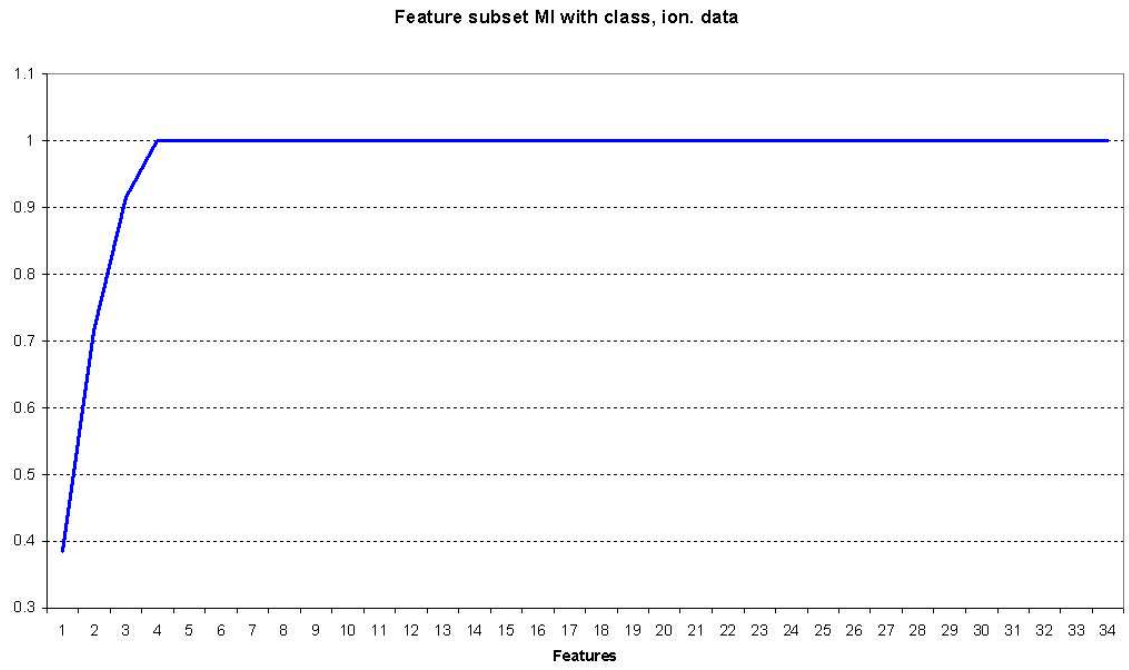

Figure 5: Subset information content plot for UCI Ionosphere data.

leads to good results in feature selection and classification.

These results, if confirmed by additional experiments, could lead to an algorithm that autonomically chooses an appropriate binning strategy and binning parameters. This would largely eliminate the main shortcoming of the method, its dependence on correct choice of binning.

6 Conclusions

This paper has introduced a new method for quickly calculating exact mutual information values for large groups of features and used it to improve the MIFSFS feature selection algorithm. The results show an improvement over methods based on pairwise mutual information evaluation in some cases and no improvement in others. It seems that real-world problems seldom contain complex group redundancies to such an extent that considering them is crucial to successful feature selection.

Nevertheless, MIFSFS has certain advantages over most pairwise methods, having no parameters that need problem-specific tuning, and being able to identify irrelevant or redundant features. The new method brings MIFSFS's computational complexity near to that of pairwise methods like MIFS-U, making it a good choice not only in cases where complex feature interaction is suspected, but in all feature selection tasks.

References

[1] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. Singular value decomposition and principal component analysis. In D. P. Berrar, W. Dubitzky, and M. Granzow, editors, *A Practical Approach to Microarray Data Analysis*, pages 91–109. Kluwer, Norwell, MA, 2003.

[2] M. Polito and P. Perona. Grouping and dimensionality reduction by locally linear embedding. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1255–1262, Cambridge, MA, 2002. MIT Press.

[3] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[4] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and A. K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4, 2000.

[5] H. Liu and R. Setiono. Dimensionality reduction via discretization. *Knowledge-Based Systems*, 9:67–72, 1996.

[6] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine*

Learning, pages 121–129, 1994. Journal version in AIJ, available at http://citeseer.nj.nec.com/13663.html.

[7] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33(1):25–41, 2000.

[8] M. Vlach. Branch and bound method for the three-index assignment problem. *Ekonomicko-Matematicky Obzor*, 12:181–191, 1967.

[9] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13:44–49, 1998.

[10] F. J. Ferri, V. Kadirkamanathan, and J. Kittler. Feature subset search using genetic algorithms. In *Proceedings of the IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, 1993.

[11] D. W. Aha and R. L. Bankert. A comparative evaluation of sequential feature selection algorithms. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: AI and Statistics*, pages 199–206. Springer-Verlag, 1996.

[12] P. Pudil, F. J. Ferri, J. Novovičová, and J. Kittler. Floating search methods for feature selection with nonmonotonic criterion functions. In *12th International Conference on Pattern Recognition*, pages 279–283, 1994.

[13] W. Wang, P. Jones, and D. Partridge. Assessing the impact of input features in a feedforward network. *Neural Computing and Applications*, 19:101–112, 2000.

[14] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

[15] N. Kwak and C.-H. Choi. Input feature selection for classification problems. *IEEE Transactions on Neural Networks*, 5:143–159, 2002.

[16] A. Al-Ani, M. Deriche, and J. Chebil. A new mutual information based measure for feature selection. *Intelligent Data Analysis*, 7(1):43–57, 2003.

[17] D. Partridge and S. Cang. Revealing feature interactions in classification tasks. In A. Abraham, J. Ruiz-del-Solar, and M. Köppen, editors, *Soft Computing Systems: Design, Management and Applications*, Frontiers in Artificial Intelligence and Applications Vol. 87, pages 394–403. IOS Press, 2002.

[18] R. Ruiz, J. S. Aguilar-Ruiz, and J. C. Riquelme. SOAP: Efficient feature selection of numeric attributes. In *Proceedings of the 8th Ibero-American*

Conference on AI: Advances in Artificial Intelligence, pages 233–242. Springer-Verlag, 2002.

[19] E. Cantú-Paz. Feature subset selection by estimation of distribution algorithms. In W. B. Langdon et al., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 303–310, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[20] W. Street, W. Wolberg, and O. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology*, volume 1905, pages 861–870, 1993.

[21] M. Scherf and W. Brauer. Feature selection by means of a feature weighting approach. Technical Report No. FKI-221-97. Institut für Informatik, Technische Universität München, Munich, 1997.

[22] N. Kwak and C.-H. Choi. Input feature selection by mutual information based on Parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1667–1671, 2002.

[23] Y.-I. Moon, B. Rajagobalan, and U. Lall. Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3):2318–2321, 1995.

[24] Ying Yang and Geoffrey I. Webb. A comparative study of discretization methods for Naive-Bayes classifiers. In *Proceedings of the Pacific Rim Knowledge Acquisition Workshop*, 2002.