TNN A768

Input Feature Selection for Classification Problems

Nojun Kwak and Chong-Ho Choi

Nojun Kwak is a Ph.D course student in the School of Electrical Eng. Seoul National Univ.,Seoul,Korea.
Chong-Ho Choi is with the School of Electrical Eng., ERC-ACI, and ASRI in Seoul National Univ., Seoul, Korea.
This research was partially supported by Brain Science and Engineering Program of Korea Ministry of Science and Technology and Brain Korea 21 program of Korea Ministry of Education.

July 5, 2001

Abstract

Feature selection plays an important role in classifying systems such as neural networks. We use a set of attributes which are relevant, irrelevant or redundant, and from the viewpoint of managing a dataset which can be huge, reducing the number of attributes by selecting only the relevant ones is desirable. In doing so, higher performances with lower computational effort is expected. In this paper, we propose two feature selection algorithms. The limitation of MIFS [1] is analyzed and a method to overcome this limitation is studied. One of the proposed algorithms makes more considered use of mutual information between input attributes and output classes than the MIFS. What is demonstrated is that the proposed method can provide the performance of the ideal greedy selection algorithm when information is distributed uniformly. The computational load for this algorithm is nearly the same as that of MIFS. In addition, another feature selection algorithm using the Taguchi method is proposed. This is advanced as a solution to the question as to how to identify good features with as few experiments as possible. The proposed algorithms are applied to several classification problems and compared with MIFS. These two algorithms can be combined to complement each other's limitations. The combined algorithm performed well in several experiments and should prove to be a useful method in selecting features for classification problems.

Keywords

Feature selection, mutual information, orthogonal array, Taguchi method, neural network.

I. Introduction

Feature selection plays an important role in classifying systems such as neural networks. For the purpose of classification problems, the classifying system has usually been implemented with rules using $if - then$ clauses, which state the conditions of certain attributes and resulting rules [2] [3]. However, it has proven to be a difficult and time consuming method. From the viewpoint of managing large quantities of data, it would still be most useful if irrelevant or redundant attributes could be segregated from relevant and important ones, although the exact governing rules may not be known. In this case, the process of extracting useful information from a large dataset can be greatly facilitated. In this paper, the problem of selecting relevant attributes among the attributes available for the purpose of classification is dealt with.

This problem of feature selection has been tackled by several researchers [1], [4]-[10]. One of the most popular methods for dealing with this problem is the PCA (principal component analysis) method [4]. This method transforms the existing attributes into new ones considered to be crucial in classification. However from the viewpoint of maintaining data, this method is not desirable, as it needs to process all the data when new data is added. The main drawback of this method is that it is not immune from distortion under transformation. Simply scaling some of the attributes can cause serious changes to the results. Recently, the feature selection problem has been dealt with intensely and some solutions have been proposed. One of the most important contributions has been made using the decision tree method. This uncovers relevant attributes one by one iteratively [9]-[12]. Setiono and Lui proposed a feature selection algorithm based on a decision tree by excluding the input features of the neural network one by one and retraining the network repeatedly [9]. It has many attractive attributes but it basically requires the process of retraining for almost every combination of input features. To overcome this shortcoming, a fast training algorithm other than the BP (back-propagation) is used, but nevertheless requires a considerable amount of time. The CDP (classifier with dynamic pruning) of Agrawal et. al. and his colleagues is also based on the decision tree which makes use of the mutual information between inputs and outputs [10]. It is very efficient in finding rules which map inputs to outputs but as a downside, requires a great deal of memory, because it generates and counts all the possible input-output pairs. Battiti's MIFS(mutual information feature selector) [1] uses mutual information between inputs and outputs like the CDP. He demonstrated that mutual information can be very useful in feature selection problems and the MIFS can be used in any classifying systems for its simplicity whatever the learning algorithm may be. But the performance can be degraded as a result of large errors in estimating the mutual information.

Regarding the topic of selecting appropriate number of features, stepwise regression [13] and Winston's Best-first search [14] and considered as a standard technique. The former uses a statistical partial F-test in deciding whether to add a new feature or to stop regression. The latter searches the space of attribute subsets by greedy hillclimbing augmented with backtracking facility. Since it does not care how the performance of subsets are evaluated, the sucess of the algorithm usually depends on the subset evaluation scheme.

This paper investigates the limitation of MIFS using a simple example and proposes an algorithm which can overcome this limitation and improve performance. The Taguchi method [15] [16] to the feature selection problem is also applied. The Taguchi method was devised for robust design of complex systems and has been successfully applied in many manufacturing problems. Recently, Peterson et al. used the Taguchi method to find the neural network structure that best fits the given data [17].

In the following section, the basics of information theory and the Taguchi method are briefly presented with concepts such as entropy, mutual information, and orthogonal array. In Section III, the limitation of MIFS is analyzed and an improved version of MIFS is proposed. In Section IV, we show the limitation of the selection algorithms based on mutual information and how the Taguchi method can be incorporated in such cases. In Section V, the proposed algorithms are applied to several classification problems to show their effectiveness. And finally, conclusions follow in Section VI. In implementing classifying systems, we use neural networks, but the proposed methods can be equally well used in other applications.

II. Preliminaries

In this section we briefly introduce some basic concepts and notations of the information theory and the Taguchi method which are used in the development of the proposed algorithms.

A. Entropy and Mutual Information

A classifying system such as neural networks (NN) maps input features onto output classes. In this process, there are relevant features that have important information regarding output, whereas irrelevant ones contain little information regarding output. In solving feature selection problems, we try to find inputs that contain as much information about the output as possible and need tools for measuring the information. Fortunately, Shannon's information theory provides a way to measure the information of random variables with entropy and mutual information [19] [20].

The entropy is a measure of uncertainty of random variables. If a discrete random variable X has $\mathcal{X}$ alphabets and the probability density function (pdf) is $p(x) = \Pr\{X = x\}$, $x \in \mathcal{X}$, the entropy of X is defined as

$$H(X) = -\sum_{x\in\mathcal{X}} p(x) \log p(x). \tag{1}$$

Here the base of log is 2 and the unit of entropy is the bit. For two discrete random variables X and Y with their joint pdf $p(x, y)$, the joint entropy of X and Y is defined as

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\ \log p(x,y). \tag{2}$$

When certain variables are known and others are not, the remaining uncertainty is measured by the conditional entropy:

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathcal{X}} p(x)\, H(Y|X=x) \\ &= -\sum_{x\in\mathcal{X}} p(x) \sum_{y\in\mathcal{Y}} p(y|x)\ \log p(y|x) \\ &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\ \log p(y|x). \end{aligned} \tag{3}$$

The joint entropy and the conditional entropy has the following relation:

$$\begin{aligned} H(X,Y) &= H(X) + H(Y|X) \\ &= H(Y) + H(X|Y). \end{aligned} \tag{4}$$

This, known as the "chain-rule", implies that the total entropy of random variables X and Y is the entropy of X plus the remaining entropy of Y for a given X.

The information found commonly in two random variables is of importance in our work and this is defined as the mutual information between two variables:

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\ \log \frac{p(x,y)}{p(x)p(y)}. \tag{5}$$

If the mutual information between two random variables is large (small), it means two variables are closely (not closely) related. If the mutual information becomes zero, the two random variables are totally unrelated or the two variables are independent. The mutual information and the entropy have the following relation, as shown in Fig. 1.

$$\begin{aligned}
I(X;Y) &= H(X) - H(X|Y) \\
I(X;Y) &= H(Y) - H(Y|X) \\
I(X;Y) &= H(X) + H(Y) - H(X,Y) \\
I(X;Y) &= I(Y;X) \\
I(X;X) &= H(X). \qquad (6)
\end{aligned}$$

Until now, definitions of the entropy and the mutual information of discrete random variables have been presented. For many classifying systems the output class C can be represented with a discrete variable, while in general terms, the input features are continuous. For continuous random variables, though the differential entropy and mutual information are defined as

$$\begin{aligned}
H(X) &= -\int p(x) \, \log p(x) dx \\
I(X,Y) &= \int p(x,y) \, \log \frac{p(x,y)}{p(x)p(y)} dxdy, \qquad (7)
\end{aligned}$$

it is practically impossible to find pdfs $(p(x), p(y), p(x,y))$ exactly and to perform integration. Therefore we divide the continuous input feature space into several discrete partitions and calculate the entropy and mutual information using the definitions for discrete cases. The inherent error that exists in the process of conversion from continuous variables to discrete ones is bounded by some constant value which depends only on the number of partitions that divide the continuous space [21].

B. The Taguchi Method

The Taguchi method, based in part on the Fisher's experimental methods [22], was applied for the robust design of products by Taguchi in the early 1950's. It developed in popularity first in Japan and later on in the U.S. and Europe [15] [16]. However, relatively few applications of this method in neural networks exist [17]. Therefore we introduce this method in more detail.

One basic ingredient of the Taguchi method is the orthogonal array (OA) which is used to find important control variables that influence the performance of a product among many candidate variables based on experiments and to assign them appropriate values. Firstly, some control variables that are suspected of influencing the performance of a product are selected, and then experiments are performed by changing the values of the control variables systematically. Then, the best combination of the values of the control variables are found.

Suppose the experimental result v can be represented as a function of some discretized control variables $u_1, u_2, \cdots, u_n$, i.e., $v = f(u_1, \cdots, u_n)$. The purpose is to control these variables and to make the result v as close as possible to the desired value. If there are N variables and for each control variable u_i there are l_i levels that can be changed, the full factorial method will disclose the best combination of the variables.

However, to do this, it requires $\Pi_{i=1}^{N} l_i$ experiments. To reduce the time consumed conducting experiments while taking advantage of the performance of the full factorial method, the orthogonal array method was introduced. It is a method of setting up experiments that only requires a fraction of the full factorial combinations. The treatment combinations are chosen to provide sufficient information to determine the factor effects. The orthogonal array ordains the order of experiments in a specific way. An example of the orthogonal array is shown in Table I. In Table I there are seven variables (A to G) and each variable has two levels (1,2). If we use the full factorial method to discover the optimal combination of variables, we need to experiment with 128 ($= 2^7$) runs, whereas the orthogonal array allows us to experiment with only eight runs. The procedure of using the orthogonal array is quite simple. As shown in the table, in the first run, all the variables are set to *level-1*, and in the second run variables A to C are set to *level-1* and variables D to G are set to *level-2*, and so on.

We can compose numerous OAs by superimposing various Latin squares [1] (see [18]) and generally, the OA is represented in $La(b^c)$ form. Here a represents the number of runs to be performed, b is the number of levels of variables, and c is the number of variables. The OA in Table I is represented as $L8(2^7)$ or simply as $L8$. The way to construct an OA and some useful OAs can be found in [16].

In general, to determine the importance of a variable using the experimental results of OA, we use the analysis of the mean (ANOM) method [16]. It simply averages the results performed according to the OA to find the various factor effects. In the OA the orthogonality guarantees that all the levels are tested equally and the influences from other variables are assumed to be almost equal. Therefore the means of different levels of a variable are compared to determine the appropriate level of that variable. If the variation between the averages for different levels of a certain variable is small, it strongly suggestes that changing the level of that variable has little influence on performance and we can consider that it has a weak relation with the output. If the variation is large, we can assume that the variable influences the output greatly.

The OA with ANOM performs ideally when the output can be represented as a linear combination of input variables. It also performs well when the interaction between variables is not so strong. It is widely recognized that if the output is roughly a monotonic function of each inputs, the OA with ANOM can still be successfully used to analyze the performance [16].

III. Mutual Information Feature Selector under Uniform Information Distribution (MIFS-U)

In this section a new algorithm for input feature selection using mutual information is presented. At first, the problem under consideration will be presented.

[1]Each row and column of a Latin square has no duplicate element with equal sum.

A. *The FRn-k Problem*

In the process of selecting input features, it is desirable to reduce the number of input features by excluding irrelevant or redundant features among the ones that are extracted from raw data. This concept is formalized as selecting the most relevant k features from a set of n features, and Battiti named it as a "feature reduction" problem [1]:

[FRn-k] : Given an initial set of n features, find the subset with $k < n$ features that is "maximally informative" about the class.

As reviewed in the preceding section, the mutual information between two random variables measures the amount of information commonly found in these variables. The problem of selecting input features which contain the relevant information about the output can be solved by computing the mutual information between input features and output classes. If the mutual information between input features and output classes could be obtained accurately, the *FRn-k* problem could be reformulated as follows:

[FRn-k] : Given an initial set F with n features and C set of all output classes, find the subset $S \subset F$ with k features that minimizes $H(C|S)$, i.e., that maximizes the mutual information $I(C;S)$.

Three key strategies for solving this *FRn-k* problem can be presented for consideration. Firstly, the "generate and test" strategy. All the feature subsets S are generated and their $H(C|S)$ are compared. Potentially, this can find the optimal subset but it is almost impossible due to the large number of combinations. Secondly, there is the "backward elimination" strategy. In this strategy, from the full feature set F that contains n elements, we eliminate the worst feature one-by-one until k elements remain. This method also has many drawbacks in computing $H(C|S)$ [2]. The final strategy is "greedy selection". In this method, starting from the empty set of selected features, we add the best available input feature to the selected feature set one by one till the size of the set reaches k. This ideal greedy selection algorithm using mutual information is realized as follows:

1. (Initialization) set $F \longleftarrow$ 'initial set of n features', $S \longleftarrow$ 'empty set.'
2. (Computation of the MI with the output class) $\forall f_i \in F$, compute $I(C; f_i)$.
3. (Selection of the first feature) find the feature that maximizes $I(C; f_i)$, set $F \longleftarrow F \setminus \{f_i\}$, $S \longleftarrow \{f_i\}$.
4. (Greedy selection) repeat until desired number of features are selected.
   (a) (Computation of the joint MI between variables) $\forall f_i \in F$, compute $I(C; f_i, S)$.
   (b) (Selection of the next feature) choose the feature $f_i \in F$ that maximizes $I(C; f_i, S)$, and set $F \longleftarrow F \setminus \{f_i\}$, $S \longleftarrow \{f_i\}$.
5. Output the set S containing the selected features.

[2]The number of memory cells needed to compute $H(C|S)$ is $K_c \times \Pi_{i=1}^{m} P_i$ which can be very huge, where K_c is the number of classes, P_i is the number of partitions for *i-th* input feature space, and m is the number or elements in S

To compute the mutual information we must know the *pdf*s of variables, but this is difficult in practice, so the best we can do is to use a histogram of the data.

In selecting k features, if the output classes are composed of K_c classes and we divide the $j-th$ input feature space into P_j partitions to get the histogram, there must be $K_c \times \Pi_{j=1}^{k} P_j$ cells to compute $I(C; f_i, S)$. In this case, even for a simple problem of selecting ten important features, $K_c \times 10^{10}$ memories are needed if each feature space is divided into ten partitions. Therefore realization of the ideal greedy selection algorithm is practically impossible. To overcome this practical obstacle an alternative method of computing $I(C; f_i, S)$ has to be devised.

B. *MIFS and its Limitation*

The MIFS algorithm is the same as the ideal greedy selection algorithm except for Step 4. Instead of calculating $I(C; f_i, S)$, the mutual information between a *candidate for newly selected feature* f_i plus *already selected features in* S and output classes in C, Battiti [1] used only $I(C; f_i)$ and $I(f_i; f_j)$. To be selected, a feature which cannot be predictable from the already selected features in S, must be informative regarding the class. In the MIFS, Step 4 in ideal greedy selection algorithm was replaced as follows [1]:

4. (Greedy selection) repeat until desired number of features are selected.

   (a) (Computation of the MI between variables) for all couples of variables (f_i, f_s) with $f_i \in F, f_s \in S$ compute $I(f_i; f_s)$, if it is not yet available.

   (b) (Selection of the next feature) choose the feature $f_i \in F$ that maximizes $I(C; f_i) - \beta \sum_{f_s \in S} I(f_i; f_s)$; set $F \longleftarrow F \backslash \{f_i\}$, $S \longleftarrow \{f_i\}$.

Here β is the redundancy parameter which is used in considering the redundancy among input features. If $\beta = 0$, the mutual information among input features is not taken into consideration and the algorithm selects features in the order of the mutual information between input features and output classes, the redundancy between input features is never reflected. As β grows, the mutual informations between input features begin to influence the selection procedure and the redundancy becomes reduced. But in the case where β is too large, the algorithm only considers the relation between inputs and does not reflect the input-output relation well.

The relation between input features and output classes can be represented as shown in Fig. 2. The ideal greedy feature selection algorithm using the mutual information chooses the feature f_i that maximizes joint mutual information $I(C; f_i, f_s)$ which is the area 2,3, and 4, represented by the dashed area in Fig. 2. Because $I(C; f_s)$ (area 2 and 4) is common for all the unselected features f_i in computing the joint mutual information $I(C; f_i, f_s)$, the ideal greedy algorithm selects the feature f_i that maximizes the area 3 in Fig. 2. On the other hand, the MIFS selects the feature that maximizes $I(C; f_i) - \beta I(f_i; f_s)$. For $\beta = 1$, it corresponds to area 3 subtracted by area 1 in Fig. 2.

Therefore if a feature is closely related to the already selected feature f_s, the area 1 in Fig. 2 is large and this can degrade the performance of MIFS. For this reason, the MIFS does not work well in nonlinear problems such as the following example.

Example 1: Each of the random variables X and Y is uniformly distributed on [-0.5,0.5], and assume that there are 3 input features $X, X-Y$ and Y^2. The output belongs to class Z

$$Z = \begin{cases} 0 & \text{if } X + 0.2Y < 0 \\ 1 & \text{if } X + 0.2Y \geq 0. \end{cases}$$

When we take 1,000 samples and partition each input feature space into ten, the mutual information between each input feature and the output classes and those between input features are shown in Table II. The order of selection by the MIFS($\beta = 1$) is X, Y^2, and $X - Y$ in that order.

As shown in Table II(c) the MIFS selects Y^2 rather than the more important feature $X - Y$ as the second choice [3].

This is due to the relatively large β, and is a good example showing a case where the relations between inputs are weighted too much. The MIFS handles redundancy at the expense of classifying performance.

C. *Proposed Algorithm (MIFS-U)*

A feature selection algorithm that is closer to the ideal one than the MIFS is now proposed. The ideal greedy algorithm tries to maximize $I(C; f_i, f_s)$ (area 2, 3, and 4 in Fig. 2) and this can be rewritten as

$$I(C; f_i, f_s) = I(C; f_s) + I(C; f_i | f_s). \tag{8}$$

Here $I(C; f_i|f_s)$ represents the remaining mutual information between the output class C and the feature f_i for a given f_s. This is shown as area 3 in Fig. 2, whereas the area 2 plus area 4 represents $I(C; f_s)$. Since $I(C; f_s)$ is common for all the candidate features to be selected in the ideal feature selection algorithm, there is no need to compute this. So the ideal greedy algorithm now tries to find the feature that maximizes $I(C; f_i|f_s)$ (area 3 in Fig. 2). However, calculating $I(C; f_i|f_s)$ requires as much work as calculating $H(f_i, f_s, C)$.

So we will approximate $I(C; f_i|f_s)$ with $I(f_s; f_i)$ and $I(C; f_i)$, which are relatively easy to calculate. The conditional mutual information $I(C; f_i|f_s)$ can be represented as

$$I(C; f_i|f_s) = I(C; f_i) - \{I(f_s; f_i) - I(f_s; f_i|C)\}. \tag{9}$$

Here $I(f_s; f_i)$ corresponds to area 1 and 4 and $I(f_s; f_i|C)$ corresponds to area 1. So the term $I(f_s; f_i) - I(f_s; f_i|C)$ corresponds to area 4 in Fig. 2. The term $I(f_s; f_i|C)$ means the mutual information between

[3] Y can be calculated exactly by a linear combination of X and $X - Y$. Because the output class Z can be computed exactly by X and $X - Y$, we can say $X - Y$ rather than Y^2 is more informative about the Z for a given X. We trained neural networks and compared the classification rates later in section V. As expected, the results are 99.8% when X and $X - Y$ are selected, and 93.4% when X and Y^2 are selected.

the already selected feature f_s and the candidate feature f_i for a given class C. If conditioning by the class C does not change the ratio of the entropy of f_s and the mutual information between f_s and f_i, i.e., if the following relation holds,

$$\frac{H(f_s|C)}{H(f_s)} = \frac{I(f_s;f_i|C)}{I(f_s;f_i)}, \tag{10}$$

$I(f_s;f_i|C)$ can be represented as

$$I(f_s;f_i|C) = \frac{H(f_s|C)}{H(f_s)} I(f_s;f_i). \tag{11}$$

Using the equation above and (9)

$$\begin{aligned} I(C;f_i|f_s) &= I(C;f_i) - (1 - \frac{H(f_s|C)}{H(f_s)}) I(f_s;f_i) \\ &= I(C;f_i) - \frac{I(C;f_s)}{H(f_s)} I(f_s;f_i). \end{aligned} \tag{12}$$

If we assume that each region in Fig. 2 corresponds to its corresponding information, condition (10) is hard to satisfied when information is concentrated on one of the four regions in Fig. 2, i.e., $H(f_s|f_i,C)$, $I(f_s;f_i|C)$, $I(C;f_s|f_i)$, or $I(C;f_s;f_i)$. It is more likely that the condition (10) holds when information is distributed uniformly throughout the region of $H(f_s)$ in Fig. 2. Because of this, we will refer to the algorithm, which will be proposed shortly, as the MIFS-U. We computed (10) for *Example 1* and the values of several pieces of mutual information are shown in Table III. It shows that the relation (10) holds with less than 10% of error.

With this formula, we revise Step 4 in the ideal greedy selection algorithm as follows:

4. (Greedy selection) repeat until desired number of features are selected.

(a) (Computation of entropy) $\forall f_s \in S$, compute $H(f_s)$ if it is not already available.

(b) (Computation of the MI between variables) for all couples of variables (f_i, f_s) with $f_i \in F, f_s \in S$, compute $I(f_s;f_i)$, if it is not yet available.

(c) (Selection of the next feature) choose a feature $f \in F$ that maximizes $I(C;f_i) - \beta \sum_{f_s \in S} \frac{I(C;f_s)}{H(f_s)} I(f_i;f_s)$; set $F \longleftarrow F \backslash \{f_i\}$, $S \longleftarrow \{f_i\}$.

Here the entropy $H(f_s)$ can be computed in the process of computing the mutual information with output class C, so there is little change in computational load with respect to the MIFS. In the calculation of mutual informations and entropies, there are two mainly used approaches of partitioning the continuous feature space: equi-distance partitioning [1] and equi-probable partitioning [21]. In this paper, we used equi-probable partitioning method as in [1] [4].

Parameter β offers flexibility to the algorithm as in the MIFS. If we set β zero, the proposed algorithm chooses features in the order of the mutual information with the output. As β grows, it excludes the

[4] If the distribution of the values in a variable f_i is not known *a priori*, we computed its mean μ and standard deviation σ and cut the interval $[\mu - 2\sigma, \mu + 2\sigma]$ into p_i equally spaced segments. The points falling outside are assigned to the extreme left (right) segment.

redundant features more efficiently. In general we can set $\beta = 1$ in compliance with (12). For all the experiments to be discussed later we set $\beta = 1$ if there is no comment.

In computing mutual information $I(f_s; f_i)$, a second order joint probability distribution which can be computed from a joint histogram of variables f_s and f_i is required. Therefore, if there are n features and each feature space is divided into p partitions to get a histogram, we need p^2 memories for each of $\binom{n}{2}$ histograms to use MIFS-U. The computational effort therefore increases in the order of n^2 as the number of features increases for given numbers of examples and partitions. This implies that MIFS-U can be applied to large problems without excessive computational efforts.

IV. Taguchi Method in Feature Selection

Including the algorithm proposed in the previous section, the greedy algorithms using mutual information for input feature selection problems always select the feature that has the largest mutual information as the most important one. This method generally works well. However, if the algorithm selects a poor feature as the first candidate, the final feature set may give poor performance. This situation may occur if two or more combined features, instead of a dominant one, influence the classification procedure. To cope with this problem, we propose an algorithm using the Taguchi method which can be used together with the greedy algorithms described in the previous section.

A. Limitation of the Greedy Selection Algorithms using Mutual Information

Consider the following example for the case where the greedy selection algorithms using mutual information does not work properly.

Example 2: Each of the random variables X and Y are uniformly distributed on [-1,1] and assume that there are three input features X, Y and X^2Y. The output belongs to class Z

$$Z = \begin{cases} 0 & \text{if } XY < 0 \\ 1 & \text{if } XY \geq 0. \end{cases}$$

As can be seen, this is a variation of the typical XOR problem. When we take 1,000 samples and partition each input feature space into ten, the mutual information between the input and the output as well as the order of selection using greedy algorithms are shown in Table IV.

As in Table IV, both the MIFS and the MIFS-U select X^2Y as the most important feature rather than X or Y. This phenomenon occurs frequently when we employ the greedy selection algorithms with mutual information. This is difficult to avoid as long as only mutual information is used. In the following subsection, the Taguchi method is applied to deal with this kind of limitation.

B. Input Feature Selection by the Taguchi Method

In selecting input features, methods such as training neural networks for many different combinations of input features can be contrived, in addition to the methods using mutual information. This kind of method such as Setiono's [9], however, needs many different runs of training.

To reduce the number of trainings, we adopt the orthogonal array as an input feature selection method, which is initially motivated from the idea of reducing the number of experiments in experimental design. The orthogonal array, as mentioned in Section II.B, is used to find the sub-optimal solution by changing control variables (input features in this problem) systemically. To adopt the orthogonal array in selecting input features, we must first think of a way to determine the levels of each variable. The input feature selection problem can be considered as a problem of finding the combination of input features that performs best in classification. If there are N input features and the levels of each input feature are determined by whether the feature is included in the feature set (*level-1*) or not (*level-2*), then there are two levels for each variable. Thus, there are 2^N different points in the search space, and the optimal solution will be one of these points.

The next issue to be considered is how close the solution obtained by the orthogonal array method can be to the optimal one. To obtain an optimal solution for this case, the output should be in the form of a linear combination of input features. This is unrealistic. In general, the Taguchi method provides better results when the interactions between control variables are relatively small. We can mitigate this condition if the output, with other input variables fixed, is a monotonic function of each input variable, and in this case we can get quite good analysis on the system [16]. In the training of neural networks, if a feature is a salient one, in general terms, the performance of training using it is supposed to be better than when it is not used. That is, for a given selected feature set, the inclusion of one of the remaining features would increase the performance of the classifying system if it were a salient one. Therefore, we can consider the output as a monotonic function of each salient feature, with other variables fixed. Consequently, if the orthogonal array is applied to this problem, good performance with a relatively small number of trainings can be expected.

The following is a short explanation of how to apply the Taguchi method to the input feature selection problem (Table V). First we make an orthogonal array and let each column correspond to each input feature. Each row corresponds to one training of the neural network with input features set to *level-1* or *level-2* in that row. After all of the rows are trained, we compare the average performance with a specific feature in the input vector and that without the feature. Then, we choose features which gives better average improvement in performance . In this algorithm when there are N inputs, we need $2^{\lceil \log_2 N+1 \rceil}$ trainings [5]. As we need $O(N)$ experiments, we conclude that if we choose inputs in this way, we can get good input features with a relatively small number of trainings. The computational effort increases linearly with the number of features. Therefore the method can be applied with effect to large problems without excessive computational effort as in MIFS-U. A demonstration as to how to use the orthogonal array for the input feature selection problem will be given with the example of Table V. There are three input features ($F1 \sim F3$) and we used the OA of $L4$ as shown in Table V(a).

[5] $\lceil \cdot \rceil$ denotes smallest integer greater than or equal to the number enclosed within.

In Table V(b), the first row of OA represents that all the three features are selected as the input features. The neural network with these features are trained to give 90% of the classification rate (in this case the performance measure is the classification rate). The second row of the table shows that the network trained solely with $F1$ gives 30% of the classification rate. In this way after all the four trainings listed in the OA has been finished, the average performance using each feature and without it is calculated. For example, for $F1$ the average performance using this feature is 60% ($(90+30)/2$), and that without it is 50% ($(40+60)/2$), and so on. With this average performance, the improvement of the average performance for each feature is evaluated. It is 10% for $F1$, 20% for $F2$, while $F3$ shows a 40% improvement. This means that $F3$ has the most influence on the performance of the network, and can be regarded as the most salient feature. So we selected features $F3, F2$, and $F1$ in that order.

For most orthogonal arrays, the first row has a series of 1s and the others have 1s less than half the number of columns. If we use *level-1* for the inclusion and *level-2* for the exclusion of features, the training corresponding to the first row will always include all the features. Such an unbalance in the number of included features between the runs can degrade the performance of the selection procedure.

To avoid this problem, we train the neural network all over again with input vectors replacing *level-1* (*level-2*) by *level-2* (*level-1*) in Table V(b). With this additional training, we can select better features.

A relavant question to ask is what if there are hundreds of features that can make the size of OA extremely large. In such cases, we can use the Taguchi method after reducing the features by other selection algorithms such as the greedy selection algorithms described in the previous section, and in doing so, better performance can be expected. The feature selection algorithm using the Taguchi method is summarized as follows:

Taguchi Method in Feature Selection (TMFS)

1. (Filtering) If there are too many features, reduce the number of features to twice the number we want to select by using some algorithm such as the MIFS-U.
2. (Obtaining the orthogonal array) Obtain the orthogonal array corresponding to the number of features which are filtered in Step 1.
3. Repeat the following steps with $i = 1, 2$.

   (a) (Form an input feature vector for NN) For each row of the OA, form an input feature vector with features whose values are i in the OA.

   (b) (Training the NN) For each row, train the NN with the training data and store the performance for the test data.

   (c) (Analysis of the Mean) Calculate the average performance of each feature for its inclusion and exclusion in the input feature vector. Then, for each feature, evaluate the performance increment for its inclusion case over the exclusion case.
4. (Selecting the input features) For each feature, average out the two increments in performance for

$i = 1$ and $i = 2$ cases in Step 3, and select input features by the order of these averaged terms.

V. Experimental Results

In this section we will present some experimental results which show the characteristics of the proposed algorithms.

In using the TMFS, to clarify the terminologies, we use the term 'method I' when we compose the input feature set with elements corresponding to *level-1*, and 'method II' for that with elements corresponding to *level-2*. The multilayer perceptrons (MLP) used have one hidden layer and the BP (back-propagation) algorithm is used to train the network. If the target value for an input pattern is 0 (1), the classification is assumed to be correct when the network's output is below 0.3 (above 0.7). Otherwise, the classification is considered incorrect. In the training of NN, all the classification rates are for the test data except for the experiment *A.3)*, and the meta-parameters of the NNs were set appropriately by performing several experiments. All the inputs were normalized on [0,1].

In the Process of TMFS, we could have used different network structures as the number of features vary, but we kept them fixed because most number of features in each run is about half the number of columns for a given OA as described in Section II.

A. Simple Problems

1) MIFS-U vs. MIFS

For *Example 1* in Section III.*B*, we compared the MIFS-U with the MIFS for several different values of β, and these results are shown in Table VI. The data consisted of 1,000 patterns and the entropies and the mutual information are calculated by dividing each input feature space into ten partitions. To verify that $X - Y$ is a more important feature than Y^2, we trained neural networks with (X,$X - Y$) and (X,Y^2) as input features respectively. The correct classification rates on the test data set were 99.8% for the first, and 93.4% for the second network.b The neural networks were trained with sets of 200 training data and the classification rates are on the test data of 800 patterns. Two hidden nodes were used with a learning rate of 2.0 and momentum of 0.1. The number of epochs at the time of termination was 200.

The result shows that when using the MIFS with $\beta = 0.5 \sim 1$ as Battiti suggested [1], the results are unsatisfactory. The reason can be found in the characteristic of the problem. After the first selection of the feature X that has the greatest mutual information with the output, the mutual information of other inputs with X influences the selection of the second feature too much, as shown in Table II(c). The MIFS-U performs well for all values of β including the suggested value 1.

2) TMFS vs. Greedy Selection Algorithms (Case 1)

For *Example 2* in Section IV, we compared the greedy selection algorithms and the TMFS. The result of applying the TMFS to *Example 2* is illustrated in Table VII.

In each training of TMFS, the MLP has two hidden nodes, the learning rate and the momentum are 2.0, 0.1 respectively. We divided the data into sets of 200 training data and 800 test data. The number of training epochs was set to 300. One simulation was conducted for each row in Table VII to output TMFS result. From Table VII we select input features in the order of X, Y, and X^2Y. As shown in *Example 2* the greedy selection algorithms, like the MIFS and the MIFS-U, cannot select the correct input features in this problem, while the TMFS solves this problem correctly.

3) TMFS vs. Greedy Selection Algorithms (Case 2)

The example we are going to consider was constructed by Elder [23] as a counterexample to the CART [12], one of the greedy optimization algorithms. The dataset is given as follows [24]:

a	b	c	y
0	0	0	0
1	0	0	0
0	0	1	1
1	0	1	1
0	1	0	1
1	1	0	1
1	1	1	0
1	1	1	0

Here the output y is a function of inputs a, b, and c. As one can see from the data, the important features are b and c, because the output y can be exactly represented as $y = b \oplus c$.

The mutual information between each input feature and output are $I(a; y) = 0.0488$, $I(b; y) = 0$, and $I(c; y) = 0$ respectively. The MIFS and the MIFS-U select a as the most important feature, because the mutual information $I(a; y)$ is greater than $I(b; y)$ and $I(c; y)$.

When we apply the TMFS to this dataset, it identifies b and c as the salient features. The order of selected features is shown in Table VIII.

The MLP has two hidden nodes and the learning rate and the momentum are 2.0 and 0.1 respectively. The terminating number of epochs was 200. One simulation was conducted for each row in Table VIII. In the training and testing, we used whole eight examples.

4) Monk3 dataset

The monk3 dataset [25] is an artificial one in which robots are defined by different attributes. It has binary classes and six discrete attributes ($F1, \cdots, F6$) among which only the second, fourth and fifth are relevant to the target concept: $(F5 = 3 \cap F4 = 1) \cup (F5 \neq 4 \cap F2 \neq 3)$ where $\cap$ and $\cup$ denote logical *and* and *or*, respectively.

We used 1,000 patterns in which the attributes are generated randomly and the output classes are decided according to the target concept. For a more realistic situation, we applied the feature selection algorithms to three monk3 datasets, where none, 5% and 10% of data are noise (misclassified) respectively.

In the process of TMFS, we divided the data into 200 training data set and 800 test data set. The MLP has 4 hidden nodes as in [25] and the learning rate and the momentum are 0.01 and 0.9 respectively. The number of trainings was set to 10,000 epochs. One simulation was performed for TMFS. Filtering was not used and a $L8$ orthogonal array was used. In Table IX we showed the result of feature selection. The bold faced features are relevant ones.

The results show that the three algorithms performed well when the monk3 dataset had zero and 5% of noise. For cases with 10% of noise, the MIFS-U performed best while the MIFS selected $F4$ last and the TMFS selected it fourth.

B. IBM datasets

These datasets were generated by Agrawal *et al.* to test their data mining algorithm *CDP* [10], and Setiono *et al.* also used it for testing the performance of their feature selector [9]. Each of the datasets has nine attributes, which are *salary, commission, age, education level, make of the car, zipcode of the town, value of the house, years house owned,* and *total amount of the loan.* The three classification functions are shown in Table X.

We generated 1,000 input-output patterns and each input space was divided into 10 partitions to compute the entropies and the mutual information. For convenience, we will refer to three datasets generated by using each function in Table X as IBM1, IBM2, IBM3 and nine input features as $F1, F2, \cdots, F9$ respectively.

- MIFS-U

Fig. 3 shows the mutual information between each input feature and the output classes for IBM1, IBM2, and IBM3 datasets. In Table XI, we compared the selected features by the MIFS-U and the MIFS for the three datasets. For MIFS-U with $\beta = 0$, the order of selection was exactly the same as the order of the mutual information with the output shown in Fig. 3. The features used in the classification functions are written in bold face in Table XI.

As shown in Table XI, both the MIFS and the MIFS-U selected the desired features for the IBM1 and IBM2 datasets. Note that for IBM2, when $\beta = 0$, $F4$ was selected as the eighth important one, while it was selected as the third for both the MIFS and the MIFS-U with $\beta = 1$. This shows that both the MIFS and the MIFS-U have the ability of effectively eliminating the redundant features.

For IBM3 dataset the classification is determined by four features, i.e., salary($F1$), commission($F2$), elevel($F4$), and loan($F9$) as shown in Table X. So these four features must be chosen as salient features by good feature selectors. Table XI shows that the MIFS selects $F1, F4$, and $F9$ in the first three selection, but $F2$ is selected at the ninth with $\beta = 1$. This is an example of a case where too much weight is put on the mutual information between input features. As noted in Section III the method of excluding redundant features by MIFS may result in poor performance. For IBM3 we see that the MIFS-U selects the four features successfully. Table XII shows the selection results of IBM3 for various values of β. Note the change in the order of selection for $F2$ over β in the MIFS case. Even for relatively small values of β, the MIFS regards $F2$ as an unimportant feature, while the MIFS-U does not. This shows that the MIFS-U performs consistently better than MIFS for different values of β.

- TMFS

For IBM datasets we also tested the performance of the TMFS. To show the effectiveness of the filtering process, we compared the TMFS with and without the filtering process. For the filtering, we used the MIFS-U with $\beta = 1.0$. We filtered four and six features out of nine for IBM1 and IBM2 respectively. An OA of $L8$ was used for these datasets. We did not conduct the experiment with filtering step for IBM3 dataset, because the number of the relavant features are four whose double is close to the number of the full features. For all the experiments without the filtering step, we used an OA of $L12$. All the input features were assigned to the corresponding columns of the orthogonal array.

The MLP has ten hidden nodes and the learning rate and the momentum are 0.01 and 0.9, respectively. The NNs were trained with 500 training data and the performances were evaluated on the remaining test data. The number of epochs in training is 10,000. One simulation is performed for each dataset.

Table XIII shows the process of feature selection for the IBM1 dataset with full features. The last two columns in the table denote the classification rates on the test data set using method I and method II, respectively, and the second row from the last shows the difference in the average classification rate between the inclusion and the exclusion of each feature in the input feature vector, namely the average improvement of the classification rate. The last row shows the order of selection.

Fig. 4 shows the average improvement of the classification rates for each input feature by using the TMFS without filtering process. The features are selected by the order of the magnitude of the graph. We show the selection orders of TMFS with and without filtering process in the bottom of the Table XI

For IBM1 dataset $F1$ and $F3$ are selected successfully. Also for IBM3, we can see that the features $F1, F2, F4$, and $F9$ are selected successfully. In IBM2 dataset, the important features are $F1, F3$, and $F4$, while the TMFS without filtering process selects $F1, F3$ and $F6$. In Fig. 4 we see that $F4$ is considered as the second least important feature. This result shows some resemblance to that of the greedy selection algorithm with $\beta = 0$ (see Fig. 3). We can see in Table XI that this failure is resolved through the filtering process in the algorithm. We can say that the chance of selecting important features by the TMFS are high if the average improvement of correct classification for the input features are high.

C. Sonar target dataset

This dataset was constructed to discriminate between the sonar returns bounced off a metal cylinder and those bounced off a rock, and it was used by Battiti to test the MIFS's performance [1]. The raw data is from UC-Irvine machine learning database [26]. It consists of 208 patterns including 104 training and testing patterns each. It has 60 input features and two output classes, *metal/rock*. As in [1], we normalized the input features to have the values in [0,1] and allotted one node per each output class for the classification. We divided each input feature space into ten partitions to calculate the entropies and mutual information. Unlike the IBM datasets, we do not know which features are important *a priori*, so we selected $3 \sim 12$ features (top $5\% \sim 20\%$) among the 60 features, and trained the neural network with the set of training patterns using these input features. We compared the classification rates of MIFS and MIFS-U. Then, we selected features by the TMFS. In the filtering process of the TMFS, we used both the MIFS and the MIFS-U and compared the results. Multilayer perceptrons with one hidden layer were used and the hidden layer had three nodes as in [1]. The conventional back-propagation learning algorithm was used with the momentum of 0.0 and learning rate of 2.0. We trained the network for 300 epochs in all cases as Battiti did [1].

Fig. 5 shows the results of selection by the MIFS and the MIFS-U. In the figure, the x-axis denotes 60 features and the y-axis corresponds to the selection order. The feature that has a value of 60 is selected first, the feature with 59 is selected second, and so on.

Using the TMFS, we also selected $3 \sim 9$ features from among twice the number of candidates produced through the filtering process. The orthogonal arrays used were $L8$ for the selection of three features, $L12$ for four, $L16$ for six, and $L24$ for nine respectively. In Table XIV, we compare the performances of MIFS, MIFS-U, and TMFS filtered by either MIFS or MIFS-U for the test set. The classification rates for various numbers of selected features are compared. For comparison, we also conducted the stepwise regression [13] and Best-first search algorithm [14] with Correlation-based subset evaluator [27]. In the stepwise regression, we set confidence level α to 0.1 and all the regression steps were set identical to those in [13]. It selected six features and the performance with these were 76.12%. The Best-first algorithm reported 75.96% classification rate with 19 features. In the table, all the resultant classification rates are the average values of ten experiments.

In the table, we can see that the proposed algorithms produced better performances than the Best-first algorithm with smaller number of features. In addition, the results show that the MIFS-U performed better than the MIFS by 10% in classification rate for the $3 \sim 4$ selected features. In the other cases, the MIFS-U also worked better. In all cases, the TMFS performed better than the greedy algorithms used in the filtering process. The best performance was achieved when we used the TMFS filtered with the MIFS-U. In all the cases, the TMFS/MIFS-U algorithm performed better than the TMFS/MIFS.

To show the statistical significance of the improvements, we have performed the two-tailed T-test using

Table XIV for the following four null hypotheses.

- There is no difference between the performances of MIFS and MIFS-U.
- There is no difference between the performances of MIFS and TMFS with MIFS.
- There is no difference between the performances of MIFS-U and TMFS with MIFS-U.
- There is no difference between the performances of TMFS with MIFS and TMFS with MIFS-U.

For each of the above hypotheses, the degree of freedom is $18(10 + 10 - 2)$, since the average is over ten experiments. For 18 degree of freedom, the significance levels corresponding 1% and 5% are 2.878 and 2.101 respectively. The threshold ratio t is computed as in Table XV. Since all the threshold ratios are well over 2.878, all the null hypotheses are rejected at the confidence level of 99%. Thus, we can say that MIFS-U is better than MIFS, TMFS with MIFS (MIFS-U) is better than MIFS (MIFS-U) and so on. This suggests that the combined TMFS and MIFS-U algorithm can be used effectively for feature selection problems.

D. Ionosphere dataset

This Johns Hopkins University Ionosphere database [26] consists of 34 continuous valued attributes with a binary class. There are 351 instances and we compared the performance of the proposed feature selection algorithms with the conventional MIFS with various βs,Best-first with Correlation-based subset evaluator, stepwise regression, and One-R feature selector [28].

In training the dataset, we used MLP with three hidden nodes, momentum of 0.1, learning rate of 1.0, and 300 terminating epochs. Table XVI is the result of the experiments. Among 34 features, we selected $3 \sim 20$ features. Ten-fold cross validation was performed ten times for each experiments. The classification rates are the average of the ten experiments and standard deviations are provided in the parentheses. In performing TMFS, we preselected (filtered) 15 features with MIFS-U and used $L16$ OA.

In the table, we can see that the full 34 features do not produce best performance. Using MIFS-U or TMFS with MIFS-U, we can get $3\% \sim 4\%$ performance improvement with only 10% of its features.

The result shows the proposed algorithm performs better than the conventional feature selection methods especially when the small number of features are selected.

VI. Conclusion

In this paper, we have proposed two input feature selection algorithms for classification problems. Algorithms, such as the MIFS, based on the information theory have strong advantages because they require relatively small computational effort. But these algorithms handle redundancy at the expense of classifying performance. We have analysed the limitations of the MIFS, and proposed the MIFS-U for overcoming this limitation. The MIFS-U can provide the performance of the ideal greedy selection algorithm when the information is distributed uniformly and its computational complexity is almost the same as that of the MIFS.

We have also applied the Taguchi method, which has been successfully used in many experimental design problems, to the feature selection problems and proposed the TMFS. Though it takes more computational effort than the algorithms based on the information theory, it can supplement the greedy selection algorithms. We have tested the TMFS with several examples and the results showed that it can be a valuable tool.

When the number of input features are large, combining the MIFS-U with TMFS are expected to deliver a significant improvement in performance, especially when the number of selected input features has to be kept small. Because these methods do not require excessive computational effort, we can also apply these methods to large problems. We have used neural networks as the classifying system, but these methods can be applied to other classifying systems as well.

References

[1] Roberto Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Trans. Neural Networks*, vol. 5, no. 4, July 1994.

[2] M.-S. Chen, J. Han, P.S. Yu, "Data mining: an overview from a database perspective," *IEEE Trans. Know. and Data Eng.*, vol. 8, no. 6, Dec. 1996.

[3] T.M. Anwar, H.W. Beck, and S.B. Navathe, "Knowledge mining by imprecise querying: a classification based approach," *IEEE 8th Int. Conf. on Data Eng.*, Phoenix, Arizona, Feb. 1992.

[4] Joliffe, I.T., *Principal Component Analysis*, New York: Springer-Verlag, 1986.

[5] K.L Priddy et al., "Bayesian selection of important features for feedforward neural networks," *Neurocomputing*, vol. 5, no. 2 and 3, 1993.

[6] L.M. Belue and K.W. Bauer, "Methods of determining input features for multilayer perceptrons," *Neur. Comput.*, vol. 7, no. 2, 1995.

[7] J.M. Steppe, K.W. Bauer Jr., and S.K. Rogers., "Integrated feature and architecture selection," *IEEE Trans. Neural Networks*, vol. 7, no 4, July 1996.

[8] Q. Li, and D.W. Tufts, "Principal feature classification," *IEEE Trans. Neural Networks*, vol. 8, no. 1, Jan. 1997.

[9] R. Setiono and H. Liu, "Neural network feature selector," *IEEE Trans. Neural Networks*, vol. 8, no. 3, May 1997.

[10] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: a performance perspective," *IEEE Trans. Know. and Data Eng.*, vol. 5, no. 6, Dec. 1993.

[11] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, CA., 1993.

[12] L. Breiman, J.H. Friedman, R.A.Olshen, and C.J. Stone, *Classification and Regression Trees*, Belmont,CA: Wadsworth, 1984.

[13] N.R. Draper and H. Smith, *Applied Regression Analysis*, 2nd ed., John Wiley & Sons, NY, 1981.

[14] P.H. Winston, *Artificial Intelligence*, Addison-Wesley, MA, 1992.

[15] G. Taguchi, *Taguchi on Robust Technology Development*, ASME, New York, NY, 1993.

[16] W.Y. Fowlkes and C.M.Creveling, *Engineering Methods for Robust Product Design*, Addison-Wesley, New York, 1995.

[17] G.E. Peterson et al., "Using Taguchi's method of experimental design to control errors in layered perceptrons," *IEEE Trans. Neural Networks*, vol. 6, no. 4, July 1995.

[18] N.J.A. Sloane, http://www.research.att.com/∼njas/oadir/index.html

[19] C.E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, Urbana, IL: University of Illinois Press, 1949.

[20] T.M. Cover, and J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991.

[21] A.M. Fraser and H. L. Swinney, "Independent coordinates for strange attractors from mutual information," *Physical Review A*, vol. 33, no. 2, 1986.

[22] R.A. Fisher, *The Design of Experiments*, Oliver and Boyd, 1935.

[23] J.F. Elder, "Assisting inductive modeling through visualization," *Joint Statistical Meeting*, San Fransisco, CA, 1993.

[24] V.S. Cherkassky and I.F. Mulier, *Learning from Data*, Chapter 5, John Wiley & Sons, 1998.

[25] S.B. Thrun *et al.*, " The MONK's Problem - a performance comparison of different learning algorithms," Dept. Computer Sci., Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-91-197, 1991.

[26] P. M. Murphy and D. W. Aha, UCI repository of machine learning databases, 1994. For information contact ml-repository@ics.uci.edu or http://www.cs.toronto.edu/∼delve/.

[27] M.A. Hall, *Correlation-based Feature Subset Selection for Machine Learning*, Ph. D. Dissertation, University of Waikato, NewZealand, 1999.

[28] G. Holmes and C.G. Nevill-Manning, "Feature selection via the discovery of simple classification rule," *Proc. International Symposium on Intelligent Data Analysis*, Baden-Baden, Germany, 1995.

TABLE I

Example of the Orthogonal Array (OA)

Run	A	B	C	D	E	F	G
1	1	1	1	1	1	1	1
2	1	1	1	2	2	2	2
3	1	2	2	1	1	2	2
4	1	2	2	2	2	1	1
5	2	1	2	1	2	1	2
6	2	1	2	2	1	2	1
7	2	2	1	1	2	2	1
8	2	2	1	2	1	1	2

TABLE II

Feature Selection by MIFS for *Example 1*

(a) MI between input and output classes ($I(f_i; Z)$)

X	$X-Y$	Y^2
0.8459	0.2621	0.0170

(b) MI between input features ($I(f_i; f_j)$)

	X	$X-Y$	Y^2
X	–	0.6168	0.0610
$X-Y$	0.6168	–	0.5624
Y^2	0.0610	0.5624	–

(c) $I(f_i; Z) - I(f_i; f_s)$

$X-Y$	$I(X-Y; Z) - I(X-Y; X) = -0.3537$
Y^2	$I(Y^2; Z) - I(Y^2; X) = -0.0439$

(d) Order of Selection

	X	$X-Y$	Y^2
Ideal Greedy	1	2	3
MIFS ($\beta = 1$)	1	3	2

TABLE III

Validation of (10) for *Example 1*

$H(f_s|C)/H(f_s)$

$H(X)$	3.3181
$H(X|Z)$	2.4723
$H(X|Z)/H(X)$	0.745

$I(f_s; f_i|C)/I(f_s; f_i)$

$I(X-Y; X)$	0.6168	$I(Y^2; X)$	0.0610
$I(X-Y; X|Z)$	0.4379	$I(Y^2; X|Z)$	0.0491
$I(X-Y; X|Z)/I(X-Y; X)$	0.709	$I(Y^2; X)/I(Y^2; X|Z)$	0.805

TABLE IV

Order of Selection for *Example 2* using Greedy Selection Algorithms

Input Feature	X	Y	X^2Y
MI with Output	0.0089	0.0073	0.0111
Order of Selection (MIFS($\beta = 1$))	3	2	1
Order of Selection (MIFS-U($\beta = 1$))	3	2	1

TABLE V

Example of Input feature Selection with OA

(a) Orthogonal Array ($L4$)

Run	F1	F2	F3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

(b) Input Feature Selection (O : select, – : do not select)

NN Training	F1	F2	F3	Performance(%)
1	O	O	O	90
2	O	–	–	30
3	–	O	–	40
4	–	–	O	60
Average Performance(select) (%)	60	65	75	–
Average Performance(not select) (%)	50	45	35	–
Avg. Improvement of Performance(%)	10	20	40	–
Order of Selection	3	2	1	–

TABLE VI

Comparison of MIFS and MIFS-U for *Example 1* (F1 = X, F2 = $X - Y$, F3 = Y^2 ; F1 and F2 are salient features)

(a) Results for MIFS

β	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4
First Selection	F1	F1	F1	F1	F1	F1	F1	F1
Second Selection	F2	F2	F2	F3	F3	F3	F3	F3
Third Selection	F3	F3	F3	F2	F2	F2	F2	F2

(b) Results for MIFS-U

β	0.0	0.2	0.4	0.6	0.8	1.0	1.2	1.4
First Selection	F1	F1	F1	F1	F1	F1	F1	F1
Second Selection	F2	F2	F2	F2	F2	F2	F2	F2
Third Selection	F3	F3	F3	F3	F3	F3	F3	F3

TABLE VII

Order of selection when TMFS is applied to the example of case 1

	Run	X	Y	X^2Y	Performance (%)
Method I	1	O	O	O	89.0
	2	O	–	–	0.0
	3	–	O	–	0.0
	4	–	–	O	0.0
	Improvement (%)	44.5	44.5	44.5	–
Method II	1	–	–	–	0.0
	2	–	O	O	1.8
	3	O	–	O	82.4
	4	O	O	–	91.7
	Improvement (%)	86.15	5.55	-3.75	–
Avg. Improv. of Performance		65.33	25.03	20.38	–
Order of Selection		1	2	3	–

TABLE VIII

Order of selection when TMFS is applied to the example of case 2

	Run	a	b	c	Performance (%)
Method I	1	O	O	O	100.0
	2	O	–	–	25.0
	3	–	O	–	0.0
	4	–	–	O	0.0
	Improvement (%)	62.5	37.5	37.5	–
Method II	1	–	–	–	0.0
	2	–	O	O	100.0
	3	O	–	O	37.5
	4	O	O	–	37.5
	Improvement (%)	-12.5	50.0	50.0	–
Avg. Improv. of Performance		25.0	43.75	43.75	–
Order of Selection		3	1	1	–

TABLE IX

Order of Selection for Monk3 dataset

	F1	**F2**	F3	**F4**	**F5**	F6
Noiseless case						
MIFS	6	**2**	4	**3**	**1**	5
MIFS-U	6	**2**	4	**3**	**1**	5
TMFS	6	**1**	4	**3**	**2**	5
5% of noise case						
MIFS	5	**1**	4	**3**	**2**	6
MIFS-U	6	**1**	4	**3**	**2**	5
TMFS	6	**1**	4	**3**	**2**	5
10% of noise case						
MIFS	5	**2**	4	**6**	**1**	3
MIFS-U	4	**2**	6	**3**	**1**	5
TMFS	6	**1**	3	**4**	**2**	5

TABLE X

IBM Classification Functions

Function 1	
Group A:	$((\text{age} < 40) \ \wedge \ (50000 \leq \text{salary} \leq 100000)) \vee$
	$((40 \leq \text{age} < 60) \ \wedge (75000 \leq \text{salary} \leq 125000)) \vee$
	$((\text{age} \geq 60) \ \wedge (25000 \leq \text{salary} \leq 75000))$.
Group B:	Otherwise.
Function 2	
Group A:	$((\text{age} < 40) \wedge$
$(((\text{elevel} \in [0 \ldots 2] \ ? \ (25000 \leq \text{salary} \leq 75000)) : (50000 \leq \text{salary} \leq 100000))))\vee$	
$((40 \leq \text{age} < 60) \wedge$	
$(((\text{elevel} \in [1 \ldots 3] \ ? \ (50000 \leq \text{salary} \leq 100000)) : (75000 \leq \text{salary} \leq 125000))))\vee$	
$((\text{age} \geq 60) \wedge$	
$(((\text{elevel} \in [2 \ldots 4] \ ? \ (50000 \leq \text{salary} \leq 100000)) : (25000 \leq \text{salary} \leq 75000))))$.	
Group B:	Otherwise.
Function 3	
Group A:	disposable > 0, where
disposable $= (0.67 \times (\text{salary} + \text{commission}) - 5000 \times \text{elevel} - 0.2 \times \text{loan} - 10000)$.	
Group B:	Otherwise.

TABLE XI

Feature Selection for IBM datasets. The bold faced features are the relevant ones in the classification.

IBM 1									
	F1	F2	**F3**	F4	F5	F6	F7	F8	F9
MIFS / MIFS-U ($\beta = 0$)	**1**	3	**2**	8	7	9	6	4	5
MIFS ($\beta = 1$)	**1**	9	**2**	3	5	4	8	6	7
MIFS-U ($\beta = 1$)	**1**	9	**2**	3	6	8	7	4	5
TMFS (without filtering)	**1**	5	**2**	8	7	3	4	9	6
TMFS (with filtering)	**1**	–	**2**	3	–	–	–	4	–
IBM 2									
	F1	F2	**F3**	**F4**	F5	F6	F7	F8	F9
MIFS / MIFS-U ($\beta = 0$)	**2**	3	**1**	**8**	5	6	7	9	4
MIFS ($\beta = 1$)	**2**	9	**1**	**3**	5	4	8	6	7
MIFS-U ($\beta = 1$)	**2**	9	**1**	**3**	5	6	7	8	4
TMFS (without filtering)	**1**	3	**2**	**8**	9	5	6	4	7
TMFS (with filtering)	**1**	–	**2**	**3**	5	4	–	–	6
IBM 3									
	F1	**F2**	F3	**F4**	F5	F6	F7	F8	**F9**
MIFS / MIFS-U ($\beta = 0$)	**2**	**3**	6	**4**	8	9	7	5	**1**
MIFS ($\beta = 1$)	**2**	**9**	7	**3**	5	4	8	6	**1**
MIFS-U ($\beta = 1$)	**2**	**3**	5	**4**	8	7	9	6	**1**
TMFS	**2**	**4**	8	**3**	7	9	5	6	**1**

TABLE XII

Order of Feature Selection for Various Values of β (IBM3)

MIFS									
β	Order of Selection								
0.0	F9	F1	**F2**	F4	F8	F3	F7	F5	F6
0.1	F9	F1	F4	F8	F3	F6	F5	**F2**	F7
0.2	F9	F1	F4	F6	F8	F5	F3	F7	**F2**
0.5	F9	F1	F4	F6	F5	F8	F3	F7	**F2**
0.7	F9	F1	F4	F6	F5	F8	F3	F7	**F2**
1.0	F9	F1	F4	F6	F5	F8	F3	F7	**F2**
1.2	F9	F1	F4	F6	F5	F8	F3	F7	**F2**
1.5	F9	F1	F4	F6	F5	F8	F3	F7	**F2**
MIFS-U									
β	Order of Selection								
0.0	F9	F1	**F2**	F4	F8	F3	F7	F5	F6
0.1	F9	F1	**F2**	F4	F8	F3	F7	F5	F6
0.2	F9	F1	**F2**	F4	F8	F3	F7	F5	F6
0.5	F9	F1	**F2**	F4	F8	F3	F6	F5	F7
0.7	F9	F1	**F2**	F4	F8	F3	F6	F5	F7
1.0	F9	F1	**F2**	F4	F3	F8	F6	F5	F7
1.2	F9	F1	**F2**	F4	F3	F8	F6	F5	F7
1.5	F9	F1	**F2**	F4	F3	F6	F8	F5	F7

TABLE XIII

Feature selection for IBM1 using TMFS ($L12$ is used).

Run	F1	F2	F3	F4	F5	F6	F7	F8	F9	–	–	Performance (Method I)	Performance (Method II)
1	1	1	1	1	1	1	1	1	1	1	1	99.8	0.0
2	1	1	1	1	1	2	2	2	2	2	2	98.3	74.6
3	1	1	2	2	2	1	1	1	2	2	2	69.5	65.2
4	1	2	1	2	2	1	2	2	1	1	2	98.0	65.8
5	1	2	2	1	2	2	1	2	1	2	1	71.2	76.1
6	1	2	2	2	1	2	2	1	2	1	1	70.4	78.8
7	2	1	2	2	1	1	2	2	1	2	1	71.3	99.0
8	2	1	2	1	2	2	2	1	1	1	2	67.4	98.8
9	2	1	1	2	2	2	1	2	2	1	1	75.2	78.0
10	2	2	2	1	1	1	1	2	2	1	2	76.0	98.5
11	2	2	1	2	1	2	1	1	1	2	2	65.0	70.9
12	2	2	1	1	2	1	2	1	2	2	1	63.4	71.8
Avg. Impro. of Performance	40.9	13.9	38.1	10.7	11.6	17.9	14.6	8.6	12.7	–	–	–	–
Order of Selection	1	5	2	8	7	3	4	9	6	–	–	–	–

TABLE XIV

Classification Rates with Different Numbers of Features for Sonar Dataset (%) (The numbers in the parentheses are the standard deviations of ten experiments each)

Number of features	MIFS	MIFS-U	TMFS with MIFS	TMFS with MIFS-U	Stepwise regression	Best-first
3	51.71 (2.1)	65.23 (1.6)	65.19 (0.7)	71.22 (1.3)	–	–
4	59.77 (1.3)	68.38 (3.1)	68.01 (1.7)	75.55 (2.4)	–	–
6	74.81 (1.4)	77.03 (0.4)	77.47 (1.9)	79.31 (0.3)	76.12(0.3)	–
9	76.45 (2.4)	78.98 (0.7)	78.65 (1.5)	81.70 (1.2)	–	–
12	78.12 (1.8)	81.51 (0.4)	–	–	–	–
19	–	–	–	–	–	75.96(1.2)
All (60)	87.92 (0.2)					

TABLE XV

Threshold Ratio t for the Null Hypotheses

Number of features	MIFS vs. MIFS-U	MIFS vs. TMFS with MIFS	MIFS-U vs. TMFS with MIFS-U	TMFS with MIFS vs. TMFS with MIFS-U
3	36.646	36.538	24.342	24.504
4	15.370	23.102	10.973	15.382
6	9.148	6.762	27.360	5.739
9	6.072	4.664	11.747	9.527
12	11.031	–	–	–

TABLE XVI

Classification Rates with Different Numbers of Features for Ionosphere Dataset (%) (The numbers in the parentheses are the standard deviations of ten experiments each)

Number of features	One-R	MIFS ($\beta = 0.0$)	MIFS ($\beta = 0.5$)	MIFS ($\beta = 0.7$)	MIFS ($\beta = 1.0$)	MIFS-U ($\beta = 1.0$)	TMFS with MIFS-U	Stepwise regression	Best-First
3	84.04(0.1)	89.74(0.5)	88.60(0.2)	88.88(0.5)	88.88(0.5)	91.45(0.3)	92.88(0.6)	–	–
5	85.75(0.5)	90.02(0.7)	89.17(0.3)	88.31(1.1)	88.60(0.3)	91.18(0.5)	92.88(0.3)	–	–
10	90.59(1.2)	91.73(0.2)	90.02(0.4)	88.31(1.3)	88.03(0.7)	92.02(0.4)	92.13(0.7)	–	–
12	–	–	–	–	–	–	–	88.60(0.9)	–
14	–	–	–	–	–	–	–	–	90.31(0.3)
15	90.59(0.2)	89.17(0.4)	89.45(0.7)	91.45(1.0)	89.45(0.3)	91.16(0.7)	–	–	–
20	89.17(0.3)	90.88(0.7)	90.31(0.4)	90.31(0.8)	90.31(0.1)	90.88(0.2)	–	–	–
All (34)	88.31 (0.2)								

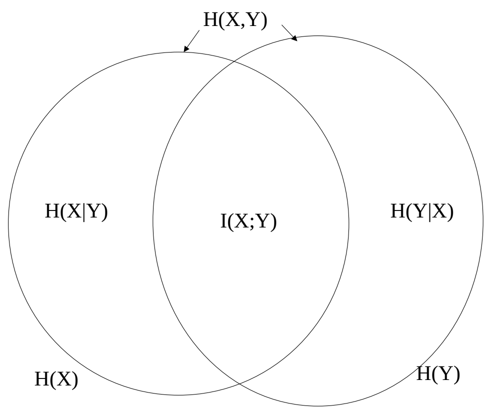


Fig. 1. The relation between the mutual information and the entropy

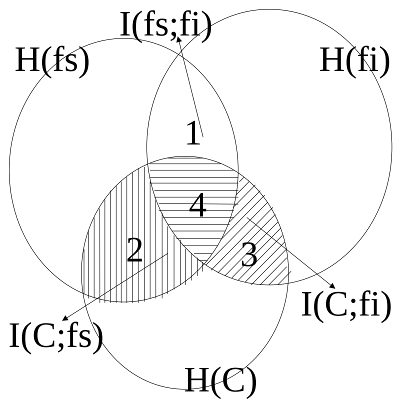
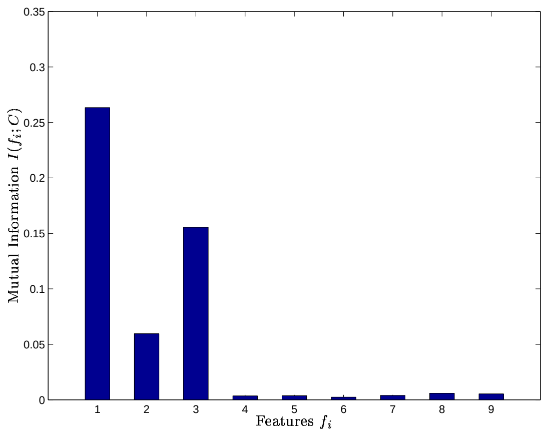
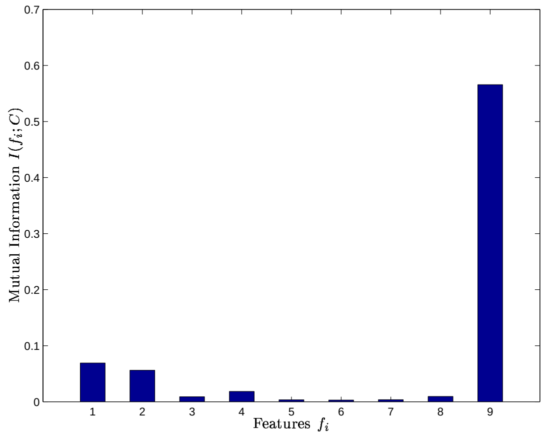


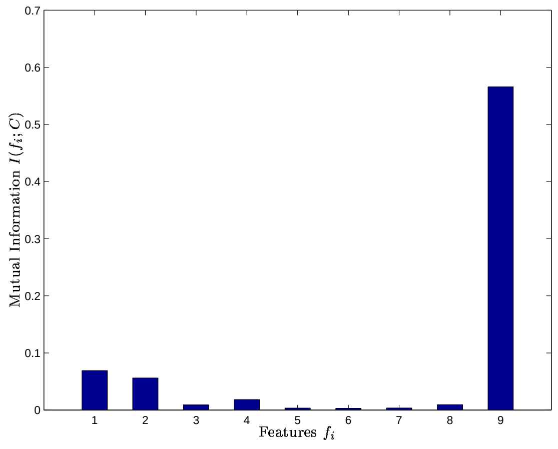
Fig. 2. The relation between input features and output classes



MI with the output (IBM1)

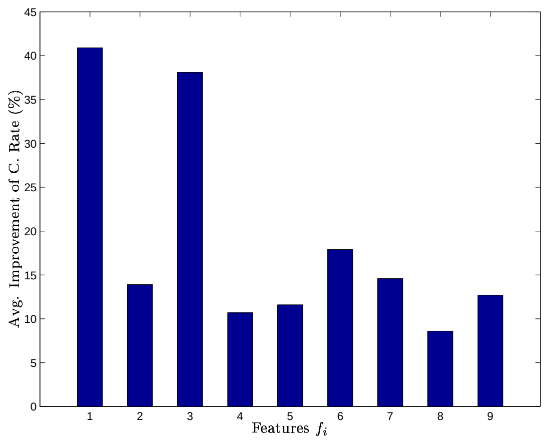


MI with the output (IBM2)

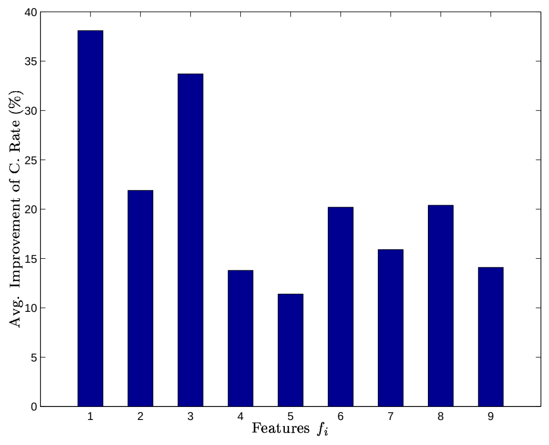


MI with the output (IBM3)

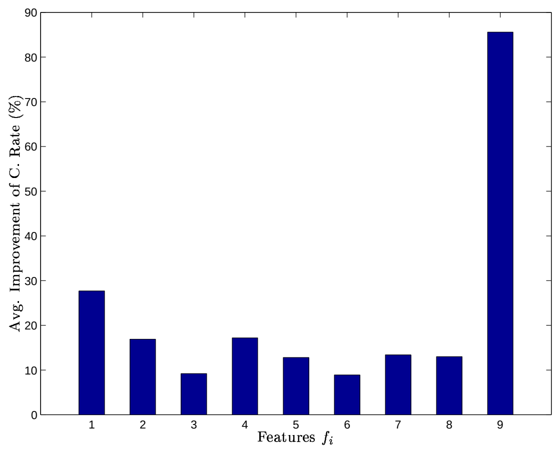
Fig. 3. Mutual information between each feature and the output class for IBM datasets



Avg. improvement of classification rate (IBM1)

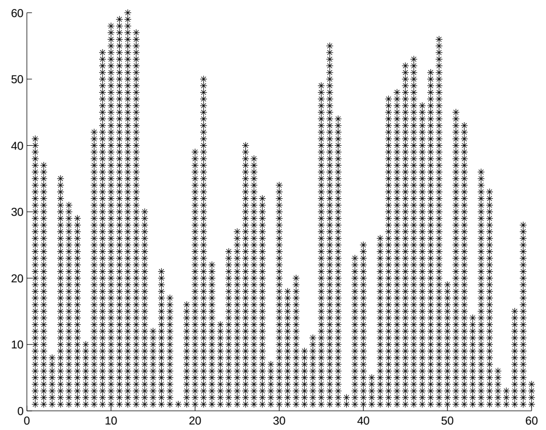


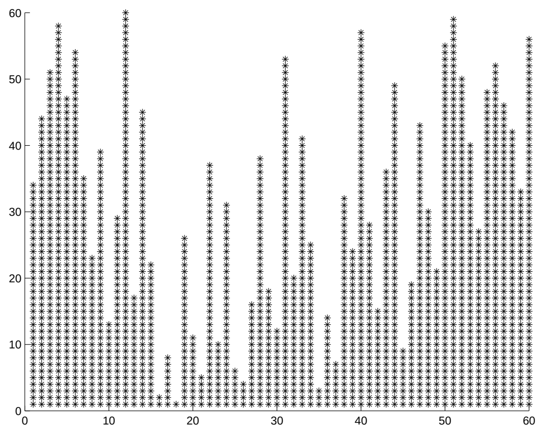
Avg. improvement of classification rate (IBM2)



Avg. improvement of classification rate (IBM3)

Fig. 4. Average improvement of the classification rates for IBM datasets by TMFS


MIFS/MIFS-U ($\beta = 0$)


MIFS ($\beta = 1$)

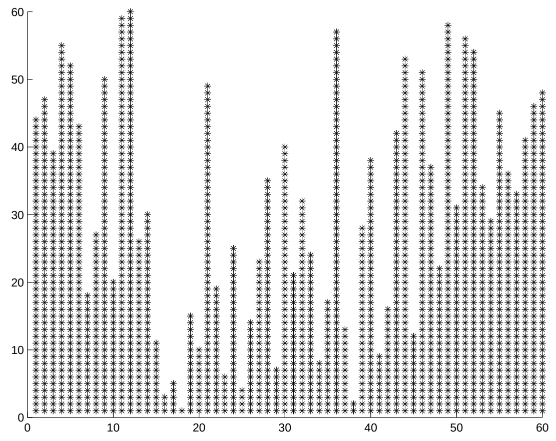

MIFS-U ($\beta = 1$)

Fig. 5. Order of Selection for Sonar dataset (MIFS and MIFS-U)