2009 21st IEEE International Conference on Tools with Artificial Intelligence

Real-valued Feature Selection by Mutual Information of Order 2

Rüdiger Brause
Department of Computer Science and Mathematics
Goethe-University,
60054 Frankfurt, Germany
r.brause (at) informatik.uni-frankfurt.de

***Abstract*—The selection of features for classification, clustering and approximation is an important task in pattern recognition, data mining and soft computing. For real-valued features, this contribution shows how feature selection for a high number of features can be implemented using mutual information. Especially, the common problem for mutual information computation of computing joint probabilities for many dimensions using only a few samples is treated by using the Rènyi mutual information of order two as computational base. The convex property is proved for ranked target samples. For real world applications like process modelling, the treatment of missing values is included.**

An example shows how the relevant features and their time lags are determined in time series even if the features determine nonlinearly the output.

By the computationally efficient implementation, mutual information becomes an attractive tool for feature selection even for a high number of real-valued features.

Keywords – mutual information; feature selection; Rènyi information; nonlinear output approximation

I. INTRODUCTION

For many applications the selection of proper input features is very important. Good features are essential for good diagnosis, prognosis, classification and approximation used in the medical, financial and industrial area. What are "good" features and how are they obtained? If we have many input features, how do we know which are the most salient ones? This is the classical task for feature selection and depends heavily on the application.

This paper shows how features can be selected using information as performance measure. We concentrate on features which contribute most of the information to the target of the application, e.g. a diagnosis or a prognosis. For information measure, we start with the traditional Shannon information using mutual information $I(\mathbf{X};Y)$ between the tuple of input features $\mathbf{X}$ and a target Y.

For categorical (symbolic) features, e.g. features exhibiting a final number of states or qualitative labels like "green", "red", "good", "sweet", the computation of mutual information between the features and the target variable is quite common for building decision trees, see [1], and are based on the probability evaluation of the states ("counting the states"). For real valued features, this is not possible any more, because we have an infinity of states. Here, other methods have to be considered.

There are many different ways of extracting relevant features for a real-valued target, like non-supervised linear transformations as the Principal Component Analysis (PCA) or the Independent Component Analysis (ICA). Both methods are based on a linear transformation of the basis vectors and need all features to compute the relevant ones. In many cases, measuring all possible features is not possible and acceptable, especially in the medical domain where each feature means an expensive and risky examination.

Instead, we use the *sequential forward selection* approach [1]. This is a greedy algorithm, and, like most greedy algorithms, not optimal: it tends to get stuck in local optima and are not to guarantee a global optimum, i.e. the optimal subset of features of all possible 2^N subsets. Certainly, there are other methods which are more efficient than greedy algorithms, like the *floating search* method [3] which includes features again already dropped, or the *branch and bound* method [4][5] which depends on the monotony of the performance criterion. Both methods are necessary when features are grouped and have to be selected as a full subset, i.e. if they are only valid together; single features do not contain much information. This is discussed in more detail in [6]. For our application here, we have no grouped features. Therefore, the sequential forward selection method is sufficient and used in this paper.

II. MUTUAL INFORMATION BASED FEATURE SELECTION

Let us use the concept of mutual information in selecting the input features. Given all input features, we have to select a proper subset of them. The concept of evaluating all possible subsets is prohibited by the combinatorial explosion of the number of subsets to be tested. Instead, we use a simple forward selection using mutual information as performance criterion.

Let us start with the observed n input features $X_1 \ldots X_n$ and the output Y. With the definition of the mutual information [7]

1082-3409/09 $26.00 © 2009 IEEE
DOI 10.1109/ICTAI.2009.127

$$I(X;Y) = H(Y) - H(Y|X) = H(Y) + H(X) - H(Y,X) \quad (1)$$

and the Shannon entropy [8]

$$H(X) = -\sum_{i=1}^{m} P_i \log(P(x_i)) = -\langle \log p(x_i) \rangle_i$$

we know that for random variables X and Y we have $H(Y) \geq H(Y|X)$ or

$$I(X;Y) \geq 0 \quad (2)$$

which becomes zero only for independent variables, see [7]. On the other hand, the more *Y* depends on feature x_i the amount of mutual information increases independently of the kind of input-output function. On this observation we build our selection procedure. The base for the greedy forward selection algorithm is the *chain rule for mutual information*, see [7], Theorem 2.5.2:

$$I(X_1,X_2,\ldots,X_n;Y)$$
$$= I(X_1;Y) + I(X_2;Y|X_1) + I(X_3;Y|X_1,X_2) + \ldots + \ldots.$$
$$= \sum_{i=1}^{n} I(X_i; Y \mid X_{i-1}, X_{i-2}, \ldots, X_1)$$

Thus, the mutual information between the *n* random variables and target *Y* can be obtained by adding *n* terms of mutual information between the target and a feature on the condition of a sequentially growing feature set. In every step, mutual information is conditional upon the joint distribution of the already selected features (random variables). Thus, we only count the additional information a feature gives us about the target in the light of the already included features, not the full information of the feature including redundancy.

If we choose the *normalized version* of mutual information

$$\tilde{I}(X;Y) = \frac{H(Y) - H(Y \mid X)}{H(Y)} = 1 - \frac{H(Y \mid X)}{H(Y)}$$

we have the property

$$0 \leq \tilde{I}(X;Y) \leq 1 \quad (3)$$

If not denoted otherwise, we use the normalized version of mutual information in the rest of the paper.

For the subsequent examples, we choose the mutual information forward selection algorithm as follows:

Forward Selection Algorithm

i = 1. As first input feature X_{k1}, select the feature with the highest mutual information I(.)

$$X_{k1} = \arg\max_{X_j} I(X_j;Y)$$

FOR i :=2 TO n DO

Select the next variable X_{ki} giving the highest mutual information

$$X_{ki} = \arg\max_{X_j} I(X_j; Y \mid X_{k1},\ldots,X_{ki-1}) \quad (4)$$

i := i+1

ENDFOR

The algorithm gives us a ranked list of variables $\{X_{k1},\ldots,X_{kn}\}$. For a set of most relevant features, we might stop the algorithm as soon as possible, e.g.

- we have reached the number of predefined input variables,
- or the amount I(.) gets no significant information increase any more,
- or the amount I(.) surpasses a predefined threshold θ,
- or the computation time surpasses a predefined threshold.

III. COMPUTING MUTUAL INFORMATION

The main reason why information based real valued feature selection is not commonly used, is the lacking of an efficient procedure for computing the mutual information. The computation of mutual information I(.) is based on the computation of the entropies H(Y), H(**X**) and H(**X**,Y) using the joint distributions P(**X**) = $P(X_{k1},\ldots,X_{kn})$ and P(**X**,Y). The computation of an entropy H(.) is impeded by the fact that in the standard case we do not have the necessary number of input samples for the computation to avoid the *curse of dimensionality* for a high number *n* of dimensions.

There are several possibilities for estimating the probability density. One idea is to circumvent the problem by approximating the mutual information between the feature set and the target output by a weighted sum of the pairwise mutual information values of the feature set, see for example Battini [9]. Certainly, for more complicated interactions this does not replace the real conditional mutual information.

Another approach is based on the Parzen window approach [10]: Each sample is taken as the center of a Gaussian distribution of variance 1; the superposition of all Gaussians is taken as the desired density function. For mutual information, this approach was introduced by Principe et al. [11] and used for learning e.g. by Torkkola [12].

Let us follow another approach. We try to circumvent the problem that we do not have sufficient samples per histogram interval by taking also the samples in the neighboured intervals into account, i.e. by averaging the number of samples. For the Shannon entropy the average is taken after computing the probability and its logarithm, not before. So, averaging the number of samples is not possible. If we could inverse the operational sequence, i.e. first compute the average of the probability and then take the logarithm, we can profit not only from the neighbours, but also avoid costly computations of the logarithm. This is performed by the following approach.

By Jensen's inequality for the convex function f(Z) (see theorem 2.6.2 in [7])

$$\langle f(z_i)\rangle_i \geq f\left(\langle z_i\rangle_i\right)$$

which is true for random variables Z, we know that for the negation of the relation

$$H(X) = -\langle \log P_i\rangle_i \leq -\log\langle P_i\rangle_i \equiv H_2(X)$$

holds. Here, the average can be computed very efficiently, see section IV. The function

$$H_2(X) = -\log\langle P(x_i)\rangle_i = -\log\sum_{i=1}^{m} P_i P(x_i)$$

$$= -\log\sum_{i=1}^{m} P_i^2$$

is called the *Rényi entropy H_α of order $\alpha = 2$* (see [13]) which is a generalization of the Shannon entropy.

Formally, the Rényi entropy is defined as follows. Let X be a discrete random variable with the probability distribution

$$P(X=x_i) \equiv \{P_i\},\ i = 1..m \quad \text{and} \quad \sum_{i=1}^{m} P_i = 1$$

Then, the Rényi-Entropy of order $\alpha \in \Re$, $\alpha \geq 0$ is defined by

$$H_\alpha(X) = \begin{cases} \dfrac{1}{1-\alpha}\log_2\sum_{i=1}^{m} P_i^\alpha & : \alpha \geq 0, \alpha \neq 1 \\ -\sum_{i=1}^{m} P_i \log_2 P_i & : \alpha = 1 \end{cases}$$

with $0^0 := 0$ and $0\cdot\log_2(0) := 0$. Here, for $\alpha = 1$ we get the Shannon entropy.

For independent random variables *X*, *Y* we know that the mutual information I(X;Y) in eq.(1) becomes zero, which is also valid for $I_2(X;Y)$. Nevertheless, in our case $I_2(X;Y)$ may also become negative, and from $I_2(X;Y) = 0$ we can not deduce the independence of *X* and *Y* because relation (2) does not hold any more. Therefore, our selection procedure may no longer be accurate. This problem is illustrated in [6] by an example.

Therefore, we need the following theorem for ensuring proportion (2):

THEOREM: For a uniformly distributed random variable *Y* with $P_i = 1/T$ and random variable *X* of unknown distribution we have

$$I_\alpha(X;Y) \geq 0 \quad \text{for } \alpha = 2 \tag{5}$$

Only iff the random variables *X* and *Y* become independent, this becomes zero.

The theorem is proved in [6]. Therefore, we have

$$\min\{H_2(X), H_2(Y)\} \geq I_2(X;Y) \geq 0$$

with I_2 equal to zero only in the case where X and Y are independent distributions. So, our forward selection procedure still holds for uniform target distributions.

This is the reason why we transfer the target time series Y to uniform distribution before we use it for computing $I_2(X;Y)$. The resulting time series y(t) is changed, but it still reflects the time series dynamics.

IV. COMPUTING MUTUAL INFORMATION OF ORDER TWO

Most of the work in using mutual information feature selection is bound to the implementation. There are several problems and their solutions which are described here. Let us start with the basic computation procedure and then consider the acceleration of the computation afterwards.

A. The basic computation

Let us assume that our samples $\{\mathbf{x}\}$ are from the *n*-dimensional space $\Re^n$. For a time series of length *T*, the k-th sample $\mathbf{x}(t_k)$ is taken at time point t_k. Given a sample $\mathbf{x}(t_1)$, the number *c* of samples $\{\mathbf{x}\}$ in its neighbourhood can be obtained by counting all samples within a hypercube of length ε, i.e. within the interval $[x_i(t_1) - \varepsilon/2,\ x_i(t_1) + \varepsilon/2]$ for all dimensions *i*.

We have

$$\begin{aligned} c(t_1) &= \left|\left\{(t_1,t_2) \mid \bigwedge_{i=1}^{n}\left\|x_i(t_1) - x_i(t_2)\right\| < \tfrac{\varepsilon}{2}, t_2 = 1,...,T\right\}\right| \\ &= \left|\left\{(t_1,t_2) \mid B(t_1,t_2) = \text{TRUE}, t_2 = 1,...,T\right\}\right| \\ &\text{with } B(t_1,t_2) = \bigwedge_{i=1}^{n}\left\|x_i(t_1) - x_i(t_2)\right\| < \tfrac{\varepsilon}{2} \\ &= \sum_{t_2} b(t_1,t_2) \\ &\text{and } b(t_1,t_2) = \begin{cases} 0 & B(t_1,t_2) = \text{FALSE} \\ 1 & B(t_1,t_2) = \text{TRUE} \end{cases} \end{aligned} \tag{6}$$

All decisions $\{b(t_1,t_2)\}$ can be represented by a binary matrix $\mathbf{b} = [\ b(i,j)\ ]$ containing them.

The number $c(t_1)$ is the number of co-occurrences of the sample events within the ε-hypercube. The average number over all T possible values for t_1 is

$$C = \sum_{i=1}^{T} p(c_i)c_i = \frac{1}{T}\sum_{i=1}^{T}\sum_{j=1}^{T} b(i,j)$$

This is referenced as the "correlation integral" introduced by Takens [14] and Grassberger & Procaccia [15] and used there for probability estimation in chaotic systems.

The relative number of samples per ε-hypercube, an estimation for the average probability within a cube, becomes

$$\langle P\rangle = C/T = \sum_{i=1}^{T} p(c_i)\frac{c_i}{T} = \sum_{i=1}^{T} p(c_i)^2$$

and the negative logarithm of it becomes

$$H(\mathbf{x}) = -\log\langle P(\mathbf{x})\rangle = -\log\left(\sum_{i=1}^{T} p(c_i)^2\right)$$
$$= -\log\frac{1}{T^2}\sum_{i=1}^{T}\sum_{j=1}^{T} b(i,j) \qquad (7)$$

B. Accelerating the computations

By eq. (7) the final procedure for computing the multi-dimensional entropy is reduced to counting samples b(i,j) within ε-hypercubes. Since we have *T* samples, we have to compute $T\cdot T = T^2$ decisions to cover all possible (t_1,t_2) tuples; the runtime complexity is $O(T^2)$.

1) Acceleration by symmetry: Since we have symmetric decisions $b(t_1,t_2) = b(t_2,t_1)$ we might facilitate the task by computing only half of the decisions, i.e. the upper triangular part of the matrix . The T^2 decisions are separated into the *T* trivial elements b(i,i) = 1 and the T(T−1) non-trivial elements b(i,j), $i \neq j$

$$\langle P(\mathbf{x})\rangle = \frac{1}{T^2}\sum_{i=1}^{T}\sum_{j=1}^{T} b(i,j)$$
$$= \frac{1}{T^2}\left(\sum_{i=1}^{T} b(i,i) + \sum_{i=1}^{T}\sum_{\substack{j=1\\ j\neq i}}^{T} b(i,j)\right) = \frac{T+2C_x}{T^2} \qquad (8)$$

The number C_x of comparisons in the upper triangular matrix of decisions (b_{ij}) is doubled, since they are symmetric, and complemented by the value of the main diagonal b_{ii}. The relation to the number of all possible decisions is the sample average probability.
Therefore, we have

$$H(\mathbf{x}) = -\log\frac{1}{T}\left(1+\frac{2C_x}{T}\right),\quad C_x = \sum_{i=1}^{T}\sum_{j=i+1}^{T} b(i,j) \qquad (9)$$

2) Acceleration by ranking: Further acceleration can be found by decreasing the number of decisions. Equation (9) still suggests that we have to compare all *T* samples with the rest of the samples. Since we have to rank at least the target samples before computing the information, we might as well profit by the already existing index array of the ranking. The main idea is illustrated by the drawing of the probability density function p(f) of a time series f(t).

Each function value f(t) of the time series is represented in the histogram (*pdf*) on the left hand side. Similar values, even if they occur at very different time instances, are neighbours here. This neighbourhood is reflected by the ranking index field: two neighboured samples y_i, y_j have also a small distance in their ranking index

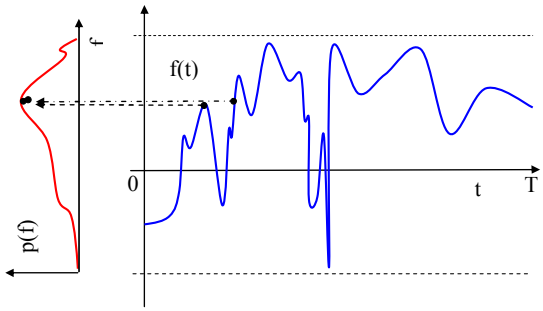

Fig. 1 The probability density function of a time series

$$|\,\mathrm{index}(y_i) - \mathrm{index}(y_j)\,| < \delta \qquad (10)$$

For instance, for the uniform distribution of the ranked output *y* we know that the *T* samples are transformed into the ranked time series with indices 0,1,2,…,T−1. Replacing the original samples by their index values and scaling them to 0.0, 1/(T−1), 2/(T−1), …, (T−1)/(T−1) = 1.0 in the interval [0.0,1.0] produces a uniformly distributed time series. Within this time series, the linearly changing index value also limits the maximal value difference of the samples. In our case, only all δ samples from index $t_1 = i+1$ to $t_2 = i+\delta$ within the index array fulfil relation (10), because

$$|y_i - y_j| < \varepsilon/2 = \varepsilon' \Leftrightarrow \left|\frac{i}{T-1} - \frac{j}{T-1}\right| < \varepsilon'$$
$$\Leftrightarrow |\,i-j\,| < \varepsilon'(T-1) \equiv \delta$$

Therefore, it is sufficient to check only for those δ samples of time series *y* if the corresponding compound input samples **x**(k) also fulfil relation (6). If YES, the comparison is counted for C_{xy}. If NO, it is omitted.
By this limitation, we are able to lower the complexity of the computation algorithm from $O(T^2)$ to $O(T\delta)$ which is much lower.

C. The missing value problem

There is one problem often found in real world data: the data are not complete. This might due to acquisition errors like broken or stuck sensors, or because the measurements are too expensive or not available, e.g. x-ray data of humans or laboratory data like tissue analysis which have not a high sampling frequency. For all these missing values, the comparison with the other time series samples at the same time point can not be done. This has some implications on the entropy computation:

- The uniform distribution can not be normalized by the number of all samples (length of the ranked array), but only by the number of valid ones.

- The number of possible comparisons which is necessary for computing the entropy in eq.(9) is reduced

by the number of missing values. This has to be taken into the computation formula.

Let us investigate this in detail.

1) Treating missing values: Let us assume that we have marked *k* missing values in the involved *n* time series. We know that for a given time point *t*, if there is only one missing value of one of the *n* series, all other samples at time *t* can not be used for this time point, too. Therefore, the number of possible comparisons b(i,j) is restricted by the *k* missing values, not by dimension *n*.

For the main diagonal of matrix **b** = [b(i,j)] (see Fig. 2) we have only T−k valuable comparisons. Each missing value, denoted by a filled circle in the T×T matrix **b**, causes also other comparisons to be invalid. Each missing value is involved in the *T* comparisons on the horizontal line and on the vertical line in Fig. 2 and therefore invalidates them. To be exact, including the missing value on the main diagonal, the first missing value invalidates T+(T−1) = 2T−1 comparisons. The next one also invalidates T+(T−1) comparisons, but the two comparison at the crossings of the vertical and horizontal lines with those of the first missing value are counted double. So, we have only T+(T−3) = 2T−3 additional invalidations for the 2nd missing value.

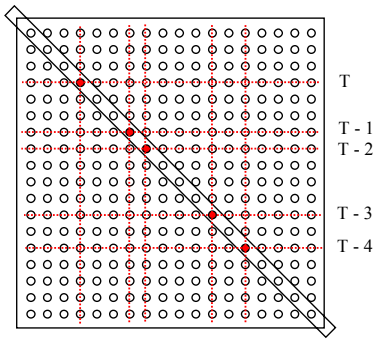

Fig. 2 The invalidations by missing values

This is also valid for the 3rd missing value which invalidates 2T−5 comparisons. In conclusion, we have

$$m = (2T-1) + (2T-3) + \ldots + (2T-2k+1)$$
$$= \sum_{i=1}^{k}(2T-2i+1) = k(2T+1) - 2\sum_{i=1}^{k} i$$

invalidations. With

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$$

we have

$$m = k(2T+1) - 2\frac{k(k+1)}{2}$$

$$= 2kT+k-k^2-k = k(2T-k)$$

invalidations. Therefore, in eq. (8)

$$\langle P(\mathbf{x})\rangle = \frac{2C_x + T}{T^2}$$

the number T of main diagonal elements with b(i,i) = 1 becomes (T−k) for *k* missing values. Additionally, the total number of valid comparisons becomes T^2−m. Therefore, we get for the estimated average probability having *k* missing values

$$\langle P(\mathbf{x})\rangle = \frac{2C_x + T - k}{T^2 - m} = \frac{2C_x + T - k}{T^2 - k(2T-k)} \qquad (11)$$

2) Treating equal values: Another problem is the appearance of equal values within a time series. Sometimes, they are due to a broken sensor and should therefore be treated as missing values. In other cases, the accuracy of the sensors is limited to a restricted number of discrete values. In the latter case, we have to treat the input differently, especially if this occurs in the target time series.

For equal values, the order of the ranked samples will be very arbitrary, depending on the sorting algorithm. For two different sorting algorithms the order in the ranking might be different, resulting in two different ranked and scaled time series and therefore in two different numerical values of the mutual information for the same input data. There are several ideas how to treat this problem:

- we might do nothing special, since the differences are small in our example
- we always use only the same sorting algorithms
- we change the input data by adding a small increment to all equal valued samples
- we treat all equal valued samples as discrete states, compute the mutual information by the conventional symbolic approach of counting states and integrate it to the I_2-estimation

The first two ideas are very arbitrary and not very appealing. The third one is also arbitrary, but provides the means for consistent results in different software environments. The fourth one is the best, but most complicated one: How should we integrate two different kinds of probability estimation into one schema?
In this paper, we use the third approach, leaving the fourth for further research.

D. The hypercube size

In this paper, for each computation of the mutual information a hypercube size ε for each counting is used. What size should it have to give optimal performance? It should not be too small, giving no result, or too big, giving an imprecise result. Here, we use a simple adaptation algorithm, an interval nesting, based on the target value of a

certain percentage *s* of samples which should be contained in the hypercube. The interval nesting should choose a cube size ε such that the entropy $H_{xy}(\varepsilon)$ is equal to the entropy $H_s = -\log(s)$. The algorithm is described in [6].

V. Approximating an Industrial Process Output

The feature selection method introduced so far is now used to select the best features for the approximation of a input-output mapping of a industrial chemical process. The approximation benchmark was introduced as a competition within the European NISIS network at 2006. It consists of a data base of 5867 samples of 14 input lines and one output ("catalyst activity") for training and four periods for prediction and retraining, see Fig. 3.

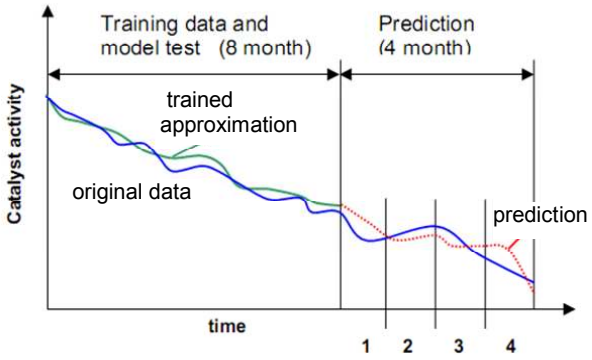

Fig. 3 The training period and the test/retraining intervals

The input/output data of each period was provided only after a prognosis was given for that interval. The prognosis was based on all data available before that period. The resulting error between the prognosis and the real data was accumulated. After the end of the contest, the best solution was marked.

The solution quality within the contest was defined by the relative error E_A of the approximation

$$E_A = \sum_{j=1}^{4} \frac{100}{N} \sum_{i=1}^{N} \frac{|y_i - L_i|}{|y_i|} \qquad (12)$$

as sum of the 4 periods containing N=15 selected samples each.

Now let us regard the task a bit closer. The chemical process can be described as follows.

A. The process

The chemical process to be modelled consists of a reactor containing some 1000 tubes filled with catalyst, used to oxidize a gaseous feed (ethane is taken as example). All measurable influences are considered as input variables for a mathematical multi-input-single-output-model describing relevant process variables (model outputs) representative for chemical processes:

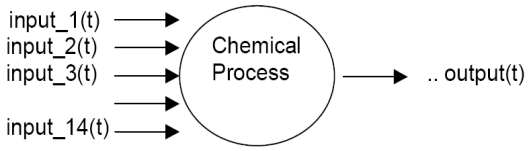

Fig. 4 The input-output modelling situation

The Input time series consists of measured flow of air, combustible gas, combustible component in combustible gas feed in mass fraction, total feed temperature, cooling temperature, product concentration of oxygen in mass fraction and product concentration of combustible component in mass fraction. The output is the catalyst activity.

B. Selecting relevant inputs

First, the input feature selection process can be applied to the 14 input lines of the process example described above. In a first attempt, we just set up a ranking according to the mutual information I_2 between the last 3800 samples of the output time series and the corresponding input time series (s=2%). This gives us Table 1 .

TABLE 1 *RANKING OF INPUTS ON I_2 BASIS*

rank	Name of var	I_2
1	QI X ORG	0,6138
2	QI EX C*	0,4829
3	FI ORG	0,3994
4	TI ROR 7	0,3296
5	TI ROR11	0,2504
6	QI EX O2	0,2250
7	TI ROR16	0,2154
8	TI ROR 4	0,2074
9	TCOOL	0,1801
10	TI ROR20	0,1763
11	FI AIR	0,1674
12	TI FEED	0,1399
13	TI ROR 1	0,1311
14	TI ROR 2	0,1266

The ranking shows a peculiarity: the input "QI X ORG" is nearly constant due to a sensor failure. Since the activity of the catalyst degrades slowly, it seems to have some information in common with a constant value. This is also partially true for other input values which can be labelled as "missing values". Therefore, we weight the input sources by their amount of missing values and select only the most reliable ones. This eliminates the input lines "QI X ORG" and "QI EX C*".

Additionally, for each time series the normalized mutual information with the output time series was computed. As we discussed in section II, this takes not the mutual dependencies into account. Therefore, we revise the list: we apply the forward selection procedure described in section II and rank the inputs according to their conditional normalized mutual information. The results are shown in Table 2.

TABLE 2 RANKING OF INPUTS ON CONDITIONAL I2 BASIS

rank	Name of var	I_2 incr	I_2
1	FI ORG	0,3994	0,3994
2	TI ROR 7	0,2118	0,6111
3	TI ROR11	0,0828	0,6939
4	FI AIR	0,0602	0,7541
5	TI ROR16	0,0562	0,8103
6	TI ROR 4	0,0142	0,8245
7	TCOOL	0,0127	0,8373
8	QI EX O2	0,0043	0,8415
9	TI ROR20	0,0044	0,8459
10	TI ROR 2	-0,0179	0,8280
11	TI ROR 1	-0,0297	0,7984
12	TI FEED	-0,0136	0,7848

We also remark that for high dimensional cells (10, 11 and 12 variables) which have to be enlarged in order to hold still 2% of the samples, the I_2 computation precision decreases and gives lower absolute I_2 values leading to negative I_2 differences: Those inputs may also be cancelled.

Now we pose the question: Does the activity depend on input values of the past? Are there time delays in the system which makes the output in a time step depend on input values of prior time steps? We test this by introducing as new inputs the old inputs delayed by different time delays, from 0 to 2000 samples in steps of 100 samples. This gives us the ranking in Table 3.

TABLE 3 RANKING OF INPUTS ON CONDITIONAL I2 BASIS INCLUDING THEIR DELAYED VERSIONS

rank	Name of var	delay	I_2 incr	I_2
1	FI ORG	700	0,5075	0,5075
2	FI ORG	1700	0,2951	0,8026
3	FI ORG	1000	0,0933	0,8959
4	FI ORG	100	0,0622	0,9581
5	FI ORG	800	0,0100	0,9681
6	FI ORG	1300	0,0046	0,9727
7	FI ORG	0	0,0145	0,9871
8	FI ORG	400	0,0051	0,9923
9	FI ORG	1200	0,0003	0,9926
10	FI ORG	0	0,0000	0,9926
11	FI ORG	0	0,0000	0,9926
12	FI ORG	0	0,0000	0,9926

We remark a stupefying result: It suffices to take the delayed versions of just one input for completely determining the output. The last three variables in the list reflect the fact that no other variable gives a contribution which increases the I_2 any more; the best variable is one which contributes zero.

The ranking of forward selection of section II can be visualized in Fig. 5. The contributions are shown as bars while the resulting I_2 is shown as function plot over the bars. The contributing variable names, composed of their origin and time delay, are shown under the bars.

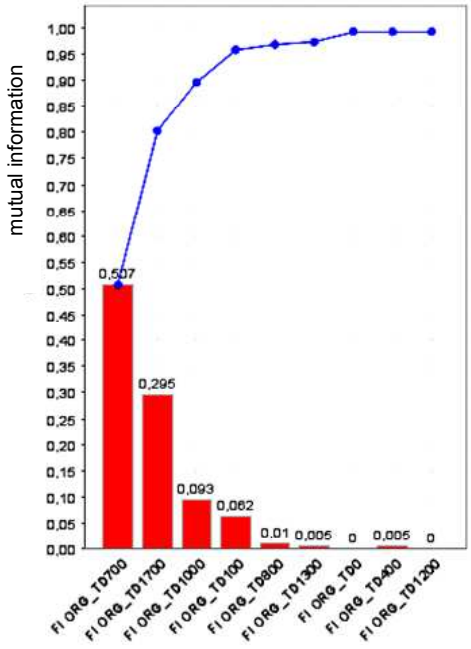

Fig. 5 *Approximating the maximal I_2 by forward input selection*

After selecting the best features we trained two kind of neural networks: A standard two-layer RBF network and a standard two-layer perceptron network with backpropagation learning.

C. Results

The original contest had a variety of system architectures. The best architecture was a set of multilayer-backpropagation networks learning by genetic algorithms. Each network was characterized by a parameter set and adapted using genetic operations.

In the following table, the errors of the best three architectures of the competition (cont1, cont2, cont3) are shown. Additionally on the last lines, we show the best results of our standard RBF and Multilayer-Perceptron approximations.

TABLE 4 PREDICTION ERROR OF THE BEST COMPETITION NETWORKS

#	Test 1	Test 2	Test 3	Test 4	**Sum**
cont1	21.01	12.87	19.14	20.13	**73.06**
cont2	43.41	18.38	52.31	24.14	**138.26**
cont3	63.15	17.83	33.89	28.91	**143.79**
RBF	1.76	6.10	5.57	15.47	**28.90**
ML	3.12	16.86	17.59	18.54	**56.11**

We see that our two standard approaches using only the selected features of maximal mutual information give much better predictions than all other specialized, more complex approaches.

VI. Conclusion

In this contribution we have shown how the selection of real valued features for mutual information can be approximated using only a few samples. This enables us to drop all irritating, irrelevant features in the process of approximating an input-output function, even if we do not know this function explicitly. Thus, the approximation process becomes much more efficient: the convergence is accelerated and the resulting error is decreased. It was shown that even the influence on the input-output mapping of time delays of the time series can be included in the feature selection process.

For the implementation of the density estimation process, the approach using the Rènyi information measure was introduced. A theorem is provided which guarantees that the minimal mutual information stays positive as long as we use at least one uniformly distributed random variable. Several algorithmic acceleration procedures were proposed and the influence of "missing values" were discussed and included in the computation scheme. All computation procedures are documented in [6].

An example shows the usefulness of feature selection for real world applications. An industrial chemical process was approximated using delayed versions of even only one input variable of 14. The results are better than that of the best solution in an international contest.

In summary, real valued feature selection shows to be a promising tool for facilitating the building of all approximation and diagnostic tools. All program code is available in [16].

Acknowledgements

We thank Sven Förster for performing the simulations on RBF and Multilayer network approximation.

References

[1] Dash M., Liu H.: "Feature Selection for Classification". Intelligent Data Analysis - An International Journal, Elsevier, Vol. 1, No. 3, pages 131 - 156, 1997

[2] Theodoridis S., Koutroumbas K.: Pattern Recognition, 2nd ed., Elsevier Academic Press, London 2003

[3] Pudil P., Novovicova J., Kittler J.: "Floating search methods in feature selection", Pattern Recognition Letters, Vol.15, pp. 1119-1125, 1994

[4] Yu B.,Yuan B.: "A more efficient branch and bound algorithm for feature selection", Pattern Recognition Vol 26 (6), pp. 883-889, 1993

[5] P. Somol, P. Pudil, F.J. Ferri and J. Kittler. "Fast branch and bound algorithm in feature selection". Proc. of SCI/ISAS 2000. Volume VII, (B. Sanchez, J.M. Pineda, J. Wolfmann, Z. Bellahsene, y F.J. Ferri, eds.), 646-651, 2000

[6] Heister F, Brause R: "Real-valued Feature Selection for process approximation and prediction", Technical Report Nr. 1/09, Computer Science Dep., Goethe University, Frankfurt, Germany 2009. Also available at http://www.informatik.uni-frankfurt.de/asa/papers /ASA_Report_1-09.pdf

[7] Cover T, Thomas J: Elements of Information Theory. John Wiley& Sons, New York 1991

[8] Shannon CE, Weaver W: The mathematical theory of information. University of Illinois Press, Urbana, 1949

[9] Battiti R.: "Using mutual information for selecting features in supervised neural net learning", IEEE Transactions on Neural Networks, vol.5, pp.537-550 (1994)

[10] Parzen E.: "On the estimation of probability density function and the mode". The Annals of Mathematical Statistics, 33, 1065. (1962).

[11] Principe J., Fisher III J., Xu, D.: "Information theoretic learning". In S. Haykin (Ed.), Unsupervised adaptive filtering. Wiley, New York, NY (2000).

[12] Torkkola K., Campell W.: "Mutual Information in Learning Feature Transformations", Proc. of the 17th Int. Conf. on Machine Learning, Morgan Kaufmann, pp.1015-1022, 2000

[13] Rényi A.: Probability Theory, Noth-Holland, Amsterdam 1970.

[14] Takens, F.: "Invariants related to dimension and entropy". In: Atas do 13. Coloquio Brasileiro de Mathematica, Rio de Janeiro (1983)

[15] Grassberger P., Procaccia I.: "Estimation of the Kolmogorov entropy from a chaotic signal". Phys. Rev. A 28, 2591-2593 (1983)

[16] Mutual Information Software: http://www.informatik.uni-frankfurt.de/asa/software/MI2/