KYBERNETIKA — VOLUME 43 (2007), NUMBER 5, PAGES 713 – 730

NOTES ON THE EVOLUTION OF FEATURE SELECTION METHODOLOGY

Petr Somol, Jana Novovičová and Pavel Pudil

The paper gives an overview of feature selection techniques in statistical pattern recognition with particular emphasis on methods developed within the Institute of Information Theory and Automation research team throughout recent years. Besides discussing the advances in methodology since times of Perez's pioneering work the paper attempts to put the methods into a taxonomical framework. The methods discussed include the latest variants of the optimal algorithms, enhanced sub-optimal techniques and the simultaneous semi-parametric probability density function modelling and feature space selection method. Some related issues are illustrated on real data by means of the Feature Selection Toolbox software.

Keywords: feature selection, branch & bound, sequential search, mixture model

AMS Subject Classification: 62H30, 62G05, 68T10

1. INTRODUCTION

The application of pattern recognition (PR) and machine learning (ML) in the real-world domain often encounters data problems caused by the high dimensionality of the input space. Dimensionality reduction refers to the task of finding low dimensional representations for high-dimensional data. More formally, given a set of n real D-dimensional vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^D$, we define dimensionality reduction as the task of finding a set of corresponding low-dimensional representatives $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$, $\boldsymbol{y}_i \in \mathbb{R}^d$, that preserve the similarity structure in the input data as well as possible. Dimensionality reduction (DR) is an important step in data preprocessing in PR and ML applications. It is sometimes the case that such tasks as classification or approximation of the data represented by so called feature vectors, can be carried out in the reduced space more accurately than in the original space.

There are two main ways of doing DR depending on the resulting features: DR by *feature selection* (FS) and DR by *feature extraction* (FE). FS approach does not attempt to generate new features, but to try to select the "best" ones from the original set of features. FE approach defines a new feature vector space in which each new feature is obtained by combinations or transformations of the original features. FS leads to savings in measurements cost since some of the features are discarded and the selected features retain their original physical interpretation. In addition, the retained features may be important for understanding the physical process that

generates the feature vectors. On the other hand, transformed features generated by feature extraction may provide a better discriminative ability than the best subset of given features, but these new features may not have a clear physical meaning.

We are honored to acknowledge that the pioneer research in this direction was carried out already as far back as in 1967 by Albert Perez and his co-workers. In the framework of a broader research project "Decision processes" (in medical care) they started to investigate problems of DR. At that time they used the term "reduction of symptom complex". Perez has shown that an intuitive reduction can lead to substantial loss of information and proposed to use mathematical methods and computers to optimize the solution. The theoretical background was formed mainly by Perez's own theoretical results like α-information values. As a result he and his co-workers developed several variants of algorithm TIBIS (Theory of Information in BIological Systems). The algorithms used either the forward (including) strategy or the backward (eliminating) strategy. The first series of PC codes was implemented in years 1967 – 1971 on MINSK 22 mainframe, in 1972 – 1973 a more advanced and sophisticated version was implemented on IBM 370 in Fortran [40]. Obviously, the implementation was restricted by hardware limitation of those early years, however, for its time it was a remarkable achievement. All the algorithms yielded subsets of symptoms from original symptom complex – thus in the present terminology carried out FS. TIBIS software was used in several medical fields, e. g., [7, 40, 41]. Further development of information theoretic approaches to FS was published in [42].

This paper is limited to the FS approach to DR. We introduce concepts and algorithms of FS that represent some of the key advances in the field since the sixties. To emphasise the continuity with Perez's work we focus mainly on the contribution of the Institute of Information Theory and Automation (ÚTIA) research team. Nevertheless, general trends in FS research are discussed and some guidelines are provided regarding the applicability of various FS concepts.

2. FEATURE SUBSET SELECTION

Let X_D be the set that comprises all the D original features, and $\mathcal{X}_d$ the set of all possible subsets of size d, where d represents the desired number of features. Let X be a subset of X_D, with d features. $J(X)$ is a function that determines how good the subset X is, by a certain criterion. Without any loss of generality, let us assume that a higher value of J indicates a better feature subset. The problem of feature selection is defined by searching the subset $\tilde{X}_d \subset X_D$ of d features, that satisfies:

$$J(\tilde{X}_d) = \max_{X \in \mathcal{X}_d} J(X) \,. \tag{1}$$

An optimal subset is always relative to a certain evaluation criterion (i. e., an optimal subset chosen using one evaluation function may not be the same as that using another evaluation function). It is searched for in a process consisting usually of these four steps: *subset generation*, *subset evaluation, stopping criterion*, and *result validation* [14]. Selection strategies and evaluation criteria are two dominating factors in designing a feature selection algorithm. For overview of various related problem aspects see, e. g., Liu and Yu [14].

2.1. Categorization of feature selection methods

Different methods have been developed and used for feature subset selection in PR and ML, using different search strategies and evaluation functions. The basic approach to subset generation is to build up a subset of required number of features starting with the empty set and successively add features (*forward* search), or to start with a full set and remove features successively (*backward* search), or to start with both ends and add and remove features simultaneously (*bi-directional* search) to optimize a chosen evaluation function. Search may also start with a randomly selected subset. Alternatively, feature selection may be an inherent part of complex classification approaches (e. g., of mixture-based discrimination, see Section 5).

2.1.1. Categorization with respect to search strategies

To balance the trade-off of result optimality and computational efficiency, different search strategies such as exhaustive, complete, heuristic, and random search have been studied to generate candidate feature subsets for evaluation.

Exhaustive Search. An exhaustive approach to FS problem requires examining all possible subsets of size $d < D$ of the set X_D and selecting the subset with the largest value of J. Although the exhaustive search guarantees the optimality of the solution according to the evaluation criterion used, it is computationally prohibitive for problems of high dimensionality.

Complete Search. The alternative to exhaustive search is the *Branch & Bound* (B&B) algorithm, proposed first in [20], and ancestor algorithms based on a similar principle. All B&B algorithms rely on the *monotonicity property* of the FS criterion: given two subsets of the feature set X_D, A and B, if $A \subset B$, then $J(A) < J(B)$. By a straightforward application of this property many feature subset evaluations may be omitted. Therefore these algorithms guarantee to select an optimal feature subset of size d without involving explicit evaluation of all the possible combinations of d features. This assumption precludes the use of the error rate as the criterion in the classification problem. This is an important drawback as the error rate can be considered superior to other criteria [11, 28]. The time complexity is exponential in terms of data dimensionality, therefore the B&B algorithm is still prohibitive in terms of computation requirements for several recent applications, particularly those in data mining and document classification involving hundreds to tens of thousands of features. Since searching for an optimal subset requires prohibitive computation, many sub-optimal search methods have been proposed, namely heuristic (also known as sequential) and random searching methods.

Heuristic Search. A number of heuristic (non-exhaustive) selection methods have been proposed which essentially trade-off the guarantee of optimality of the selected subset for computational complexity. This category includes, e. g., Best Individual Features, Sequential Forward and Backward Selection, Plus-l-Take Away-r, their generalized versions, Genetic algorithms, and particularly the Floating and Oscillating Search. A comprehensive method list can be found, e. g., in books [3, 5, 10, 37, 35]. A comparative taxonomy can be found, e. g., in [4, 9] or [12].

The time complexity of these methods is quadratic or less in terms of data dimensionality.

Random Search. It starts with a feature set containing all randomly selected features, add or remove features randomly selected one at a time and stop after a given number of iterations. The time complexity can be linear to the number of iterations. Various heuristic methods attempt to take use of the random search principle, hoping to avoid local extremes in the search space.

2.1.2. Categorization with respect to selection criteria

Based on the *selection criterion* choice, feature selection methods may roughly be divided into three types: the *filter* [2, 39], the *wrapper* [11] and the *hybrid* [1, 27].

Filter methods are based on performance evaluation functions calculated directly from the training data such as *distance, information, dependency, and consistency,* and select feature subsets without involving any learning algorithm.

Wrapper approach requires one predetermined learning algorithm and uses its performance as the evaluation criterion. It attempts to find features better suited to the learning algorithm aiming to improve its performance. In the wrapper category, *classifier error rate* is used for classification and *cluster goodness* for clustering. Generally, the wrapper method achieves better performance than the filter method, but tends to be more computationally expensive than the filter approach. Also, the wrappers yield feature subsets optimized for the given learning algorithm only – the same subset may thus be bad in another context.

Hybrid approach combines the advantages of both approaches. Hybrid algorithms have recently been proposed to deal with high dimensional data. In these algorithms, first, a goodness measure of feature subsets based on data characteristics is used to choose best subsets for a given cardinality, and then, cross validation is exploited to decide a final best subset across different cardinalities. These algorithms mainly focus on combining filter and wrapper algorithms to achieve best possible performance with a particular learning algorithm with the time complexity comparable to that of the filter algorithms.

Filters as all other FS methods may be of *local* or *global* type, depending on whether the task (e. g., classification) is performed locally for each individual class or globally under all classes.

3. ADVANCED BRANCH & BOUND TYPE ALGORITHMS

In the field of complete search we have proposed some of the currently best-performing methods. First, let us briefly summarize the essential Branch & Bound principle [3, 5]. The algorithm constructs a search tree where the root represents the set of all D features, X_D, and leaves represent target subsets of d features. While tracking the tree down to leaves the algorithm successively removes single features from the current set of "candidates" ($\tilde{X}_k$ in kth level). In leaves the algorithm verifies/updates the information about the till-now best subset of d features and the

corresponding criterion value – denoted the *bound*. Anytime a criterion value in some internal node is found to be lower than the current *bound*, due to the required monotonicity property of the adopted criterion function the whole sub-tree may be cut-off and many computations may be omitted without affecting the optimality of result. Note that it is possible to impose additional restrictions on the *bound* value in form of a pre-specified limit to transform B&B to faster but no longer optimal search.

3.1. Fast Branch & Bound

B&B algorithms do not guarantee to be always faster than exhaustive search. This is because not only target subsets of d features, but also some of their supersets are evaluated; the problem may then follow from insufficient number of sub-tree cut-offs. Several techniques can reduce this problem.

The Fast Branch & Bound (FBB) [32] algorithm aims to reduce considerably the number of criterion evaluations in internal search tree nodes. For each feature it collects *prediction information* in form of averaged difference between criterion values before and after the feature removal. This information is later used for prediction of criterion values in internal tree nodes instead of their true computation.

If the predicted value remains significantly higher than the current *bound*, it may be expected that even the true value would not be lower, and the corresponding sub-tree can not be cut-off. In this situation the algorithm continues to construct the consecutive tree level. However, if the predicted value is equal or lower than the *bound* (and therefore there is a chance that the true value is lower than the *bound*), the true criterion value must be computed. Only if true criterion values are lower than the *bound*, sub-trees may be cut-off. Note that this scheme does not affect the optimality of obtained results. The idea is illustrated in Figure 1.

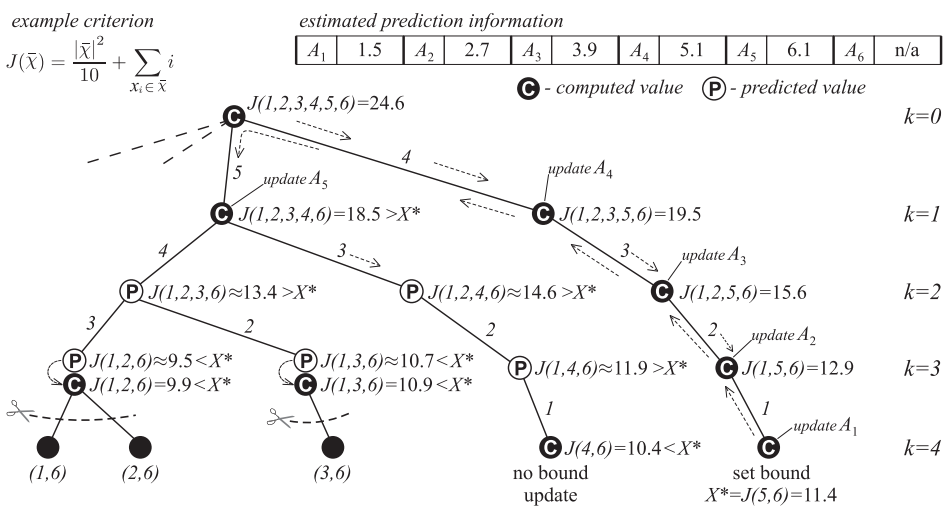

Fig. 1. Example of a "Fast Branch & Bound" problem solution; $d = 2$ features are to be selected from a set of $D = 5$ features. Dashed arrows show the way of tracking the tree.

3.2. Other Branch & Bound acceleration ideas

Many improvements of the B&B idea can be combined. One of the earliest consists in optimizing the order of feature removals in the search tree to enable faster *bound* increase and more effective branch cutting in late search stages. This modification, known as IBB (Improved Branch & Bound) [3, 5] is now part of FBB. Similarly, all B&B algorithms should construct the "minimum solution tree" [38].

Although FBB can be expected to be the fastest among all B&B algorithms, it exhibits some drawbacks: it cannot be used with recursive criterion forms and there is no theoretical guarantee that extensive prediction failures won't hinder the overall speed, despite the fact that such faulty behavior has not been observed with real data. The B&B with Partial Prediction (BBPP) [32] constitutes the less effective but robust alternative, capable of optimizing recursively computed critera.

Among other recent B&B related ideas the "trading space for speed" approach [8] deserves attention as an alternative that may operate exceptionally fast under certain circumstances. The idea of prediction is further investigated in [33, 36].

3.3. Experimental results of optimal search methods

When compared to simpler B&B algorithms the predictive algorithms spend additional time for maintaining the prediction mechanism. However, this additional time shows to be negligible when compared to considerable time savings following from omitted criterion computations. Here we show representative results on 30-dimensional mammogram data from Wisconsin Diagnostic Breast Center (2 classes – 357 benign and 212 malignant samples, see UCI repository [19]). We used Bhattacharyya distance as the criterion function. The results are collected in Figure 2.

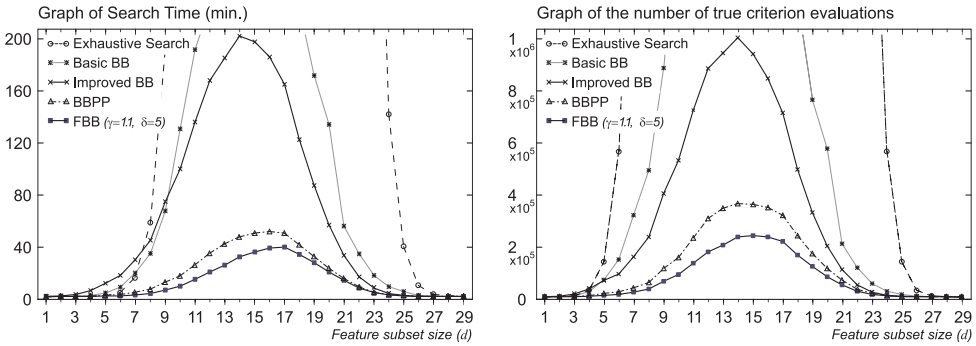

Fig. 2. Optimal subset search methods performance when maximizing Bhattacharyya distance on 30-dimensional data (Wisconsin Diagnostic Breast Center).

3.4. Summary of optimal methods

The only optimal subset search method usable with non-monotonic criteria is the exhaustive (full) search. However, because of exponential nature of the search prob-

lem, alternative methods are often needed. Our several recent improvements of the B&B idea especially in the form of prediction based FBB and BBPP resulted in a speed-up factor of 10 to 100 over the simplest B&B form, depending on particular data and criterion used.

It should be stressed that despite all advances the optimal methods are exponential in nature. If there is no particular need to request the global optimum of results, sub-optimal search methods offer greater flexibility and acceptable speed even for high-dimensional problems, while the solutions found are not necessarily much worse than the optimum. Note, that the optimality is with respect to the chosen criterion. Criterion choice may affect the final system performance more than the possibly slight difference between the optimal global and local type of feature selection.

4. HEURISTIC SEARCH

The simplest yet widely used *sequential forward (or backward) selection* methods [3], SFS (SBS), can be viewed as analogy to Perez's methods realized in TIBIS (see Section 1). They iteratively add (remove) one feature at a time so as to maximize the intermediate criterion value until the required dimensionality is achieved. Among the more interesting recent approaches the following two families of methods can be pointed out for general applicability and performance reasons, namely *sequential floating search methods* [23, 29] and *oscillating search methods* [30], both developed within Institute of Information Theory and Automation.

Earlier sequential methods suffered from the so-called nesting of feature subsets which significantly deteriorated the performance. The first attempt to overcome this problem was to employ either the Plus-l-Take Away-r [denoted (l, r)] or its generalized form [3] which involve successive augmentation and depletion process. Similar idea in an extended and refined form constitutes the basis of floating search.

4.1. Sequential floating search

The sequential forward floating selection procedure consists of applying after each forward step a number of backward steps as long as the resulting subsets are better than previously evaluated ones at that level. Consequently, there are no backward steps at all if intermediate result at actual level (of corresponding dimensionality) cannot be improved. The same applies for the backward version of the procedure. Both algorithms allow a "self-controlled backtracking" so they can eventually find good solutions by adjusting the trade-off between forward and backward steps dynamically. In a certain way, they compute only what they need without any parameter setting.

Before describing the floating algorithm formally, the following definitions have to be introduced.

Let $X_k = \{x_{i_1}, \ldots, x_{i_k}\}$ be the set of k features selected from the original set of D possible features, $X_D = \{x_i \mid i = 1, \ldots, D\}$, where $\{i_1, \ldots, i_k\}$ is a subset of $\{1, \ldots, D\}$. Then $X_D \setminus X_k = \{x_{j_1}, \ldots, x_{j_{D-k}}\}$ is the set of features not yet selected, $\{j_1, \ldots, j_{D-k}\}$ is the complement of $\{i_1, \ldots, i_k\}$ in $\{1, \ldots, D\}$. We say that the

feature x_{i_r} from the set X_k is the *best* (*worst*) feature *in the set* X_k if

$$J(X_k \setminus \{x_{i_r}\}) = \min_{s=1,\ldots,k} J(X_k \setminus \{x_{i_s}\}) \left(J(X_k \setminus \{x_{i_r}\}) = \max_{s=1,\ldots,k} J(X_k \setminus \{x_{i_s}\}) \right).$$

We say that the feature x_{j_p} from the set $X_D \setminus X_k$ is the *best* (*worst*) feature *with respect to the set* X_k if

$$J(X_k \cup \{x_{j_p}\}) = \max_{q=1,\ldots,D-k} J(X_k \cup \{x_{j_q}\}) \left(J(X_k \cup \{x_{j_p}\}) = \min_{q=1,\ldots,D-k} J(X_k \cup \{x_{j_q}\}) \right).$$

4.1.1. Sequential forward floating search – formal algorithm description

SFFS procedure is basically a bottom-up search procedure which adds (includes) new features by means of applying the basic sequential forward selection (SFS) procedure starting from the current feature set, followed by a series of successive conditional removing (exclusion) of the worst feature in the newly updated set provided a further improvement can be made to the previous sets.

SFFS can be described algorithmically as follows: Suppose k features have already been selected from the complete set of features X_D to form set X_k with the corresponding value $J(X_k)$ of the criterion J. In addition, the values of $J_i = J(X_i)$ for all so-far best found sets of size $i = 1, \ldots, k-1$ are known and stored.

Step 1: (Initialization) The algorithm is initialized by setting $k = 0$ and $X_0 = \emptyset$, and the SFS method is used until a feature set of size 2 is obtained.

Step 2: (Inclusion) Using the basic SFS method, select from the set of available features, $X_D \setminus X_k$ the best feature with respect to the set X_k, say x^+, and add it to the current set X_k to form new feature set X_{k+1}; i. e.,

$$x^+ := \arg \max_{x \in X_D \setminus X_k} J(X_k \cup \{x\}), \quad X_{k+1} := X_k \cup \{x^+\}.$$

Step 3: (Conditional exclusion) Find the worst feature x^- in the set X_{k+1}, i. e.,

$$x^- = \arg \max_{x \in X_{k+1}} J(X_{k+1} \setminus \{x\}).$$

If the feature x^- is the same as x^+, set $J_{k+1} = J(X_{k+1})$, $k = k+1$ and go to Step 2; otherwise remove this feature from X_{k+1} to form a new feature set $X_k^{'}$, i. e.,

$$X_k^{'} = X_{k+1} \setminus \{x^-\}.$$

Note that now $J(X_k^{'}) > J(X_k) = J_k$. If $k = 2$, then set $X_k = X_k^{'}$ and $J_k = J(X_k^{'})$ and go to Step 2, otherwise go to Step 4.

Step 4: (Continuation of conditional exclusion) Continue removing the features from the set $X_k^{'}$ to form reduced sets $X_{k-1}^{'}$ while

$$J(X_{k-1}^{'}) > J_{k-1}.$$

Set $k = k-1$ or $k = 2$; then continue with Step 2.

Note that the backward algorithm version (SBFS) can be defined analogously. Floating search algorithms can be considered the universal tools not only outperforming all predecessors, but also keeping advantages not met by more sophisticated algorithms. They find good solutions in all problem dimensions in one run. The overall search speed is high enough for most of ordinary problems.

4.1.2. Adaptive floating search

As the floating search algorithms have been found successful and generally accepted to be an efficient universal tool, their idea was further investigated. The so-called Adaptive Floating Search (AFS) has been proposed in [29]. The AFS algorithms are able to outperform the classical floating algorithms in certain cases, but at a cost of considerable increase of search time and the necessity to deal with unclear parameters. Our experience shows that AFS is usually inferior to newer algorithms, which we focus on in the following.

4.1.3. Hybrid floating search

Recently we have defined a flexible hybrid version of floating search methods [34] for feature selection. The main benefit of the proposed floating search hybridization is the possibility to deal flexibly with the quality-of-result vs. computational time trade-off and to enable wrapper based feature selection in problems of higher dimensionality than before. We have shown that it is possible to trade significant reduction of search time for often negligible decrease of the classification accuracy.

4.2. Oscillating search

The recent Oscillating Search (OS) [30] can be considered a "higher level" procedure, that takes use of other feature selection methods as sub-procedures in its own search. The concept is highly flexible and enables modifications for different purposes. It has shown to be very powerful and capable of over-performing standard sequential procedures, including Floating Search. Unlike other methods, the OS is based on repeated modification of the current subset X_d of d features. In this sense the OS is independent on the predominant search direction. This is achieved by alternating so-called *down-* and *up-swings*. Both *swing*s attempt to improve the current set X_d by replacing some of the features by better ones. The *down-swing* first removes, then adds back, while the *up-swing* first adds, then removes. Two successive opposite swings form an *oscillation cycle*. The OS can thus be looked upon as a controlled sequence of oscillation cycles. The value of o (denoted *oscillation cycle depth*) determines the number of features to be replaced in one swing. o is increased after unsuccessful oscillation cycles and reset to 1 after each X_d improvement.

The algorithm terminates when o exceeds the user-specified *limit* Δ, or it may be left to reach the limits (full or empty set of features).

Every OS algorithm requires some initial set of d features. The initial set may be obtained randomly or in any other way, e. g., using some of the traditional sequential

selection procedures. Furthermore, almost any feature selection procedure can be used in *up-* and *down-swing*s to accomplish the replacements of feature o-tuples. Therefore, for the sake of generality in the following descriptions let us denote the adding / removing of a feature o-tuple by ADD(o) / REMOVE(o).

4.2.1. Oscillating search – formal algorithm description

Step 1: (*Initialization*) By means of any feature selection procedure (or randomly) determine the initial set X_d of d features. Let $c = 0$. Let $o = 1$.

Step 2: (*Down-swing*) By means of REMOVE(o) remove such o-tuple from X_d to get new set X_{d-o} so that $J(X_{d-o})$ is maximal. By means of ADD(o) add such o-tuple from $X_D \setminus X_{d-o}$ to X_{d-o} to get new set $X_d^{'}$ so that $J(X_d^{'})$ is maximal. If $J(X_d^{'}) > J(X_d)$, let $X_d = X_d^{'}$, $c = 0$, $o = 1$ and go to Step 4.

Step 3: (*Last swing has not improved the solution*) Let $c = c + 1$. If $c = 2$, then nor the last *up-* nor *down-swing* led to a better solution. Extend the search by letting $o = o + 1$. If $o > \Delta$, stop the algorithm, otherwise let $c = 0$.

Step 4: (*Up-swing*) By means of ADD(o) add such o-tuple from $X_D \setminus X_d$ to X_d to get new set X_{d+o} so that $J(X_{d+o})$ is maximal. By means of REMOVE(o) remove such o-tuple from X_{d+o} to get new set $X_d^{'}$ so that $J(X_d^{'})$ is maximal. If $J(X_d^{'}) > J(X_d)$, let $X_d = X_d^{'}$, $c = 0$, $o = 1$ and go to Step 2.

Step 5: (*Last swing has not improved the solution*) Let $c = c + 1$. If $c = 2$, then nor the last *up-* nor *down-swing* led to a better solution. Extend the search by letting $o = o + 1$. If $o > \Delta$, stop the algorithm, otherwise let $c = 0$ and go to Step 2.

4.2.2. Oscillating search properties

The generality of the OS search concept allows to adjust the search for better speed or better accuracy (lower Δ and simpler ADD / REMOVE vs. higher Δ and more complex ADD / REMOVE). In this sense let us denote *sequential OS* the simplest possible OS version which uses a sequence of SFS steps in place of ADD( ) and a sequence od SBS steps in place of REMOVE( ). As opposed to all sequential search procedures, OS does not waste time evaluating subsets of cardinalities too different from the target one. The fastest improvement of the target subset may be expected in initial phases of the algorithm, because of the low initial cycle depth. Later, when the current feature subset evolves closer to optimum, low-depth cycles fail to improve and therefore the algorithm broadens the search (by letting $o = o + 1$). Though this improves the chance to get closer to optimum, the trade-off between finding a better solution and computational time becomes more apparent. Consequently, the OS tends to improve the solution most considerably during the fastest initial search stages. This behavior is advantageous, because it gives the option of stopping the search after a while without serious result-degrading consequences. Let us summarize the key OS advantages:

(i) OS may be looked upon as a universal tuning mechanism, being able to improve solutions obtained in another way.

(ii) The randomly initialized OS is very fast, in case of very high-dimensional problems may become the only applicable procedure. E.g., in document analysis for search of the best 1000 words out of a vocabulary of 50 000 even the simple SFS may show to be too slow.

(iii) Because the OS processes subsets of target cardinality from the very beginning, it may find solutions even in cases, where the sequential procedures fail due to numerical problems.

(iv) Because the solution improves gradually after each oscillation cycle, with the most notable improvements at the beginning, it is possible to terminate the algorithm prematurely after a specified amount of time to obtain a usable solution. The OS is thus suitable for use in real-time systems.

(v) In some cases the sequential search methods tend to uniformly get caught in certain local extremes. Running the OS from several different random initial points gives better chances to avoid that local extreme.

4.3. Experimental results of sub-optimal search methods

All described sequential search methods have been tested on a large number of different problems. Here we demonstrate their performance on 2-class, 30-dimensional mammogram data (see Section 3.3). The graphs in Figure 3 show the OS ability to outperform other methods even in the simplest *sequential OS* form (here with $\Delta = d$ in one randomly initialized run). The ASFFS behavior is well illustrated here showing better performance than SFFS at a cost of uncontrollably increased time. SFFS and SFS need one run only to get all solutions. SFFS performance is always better than that of SFS.

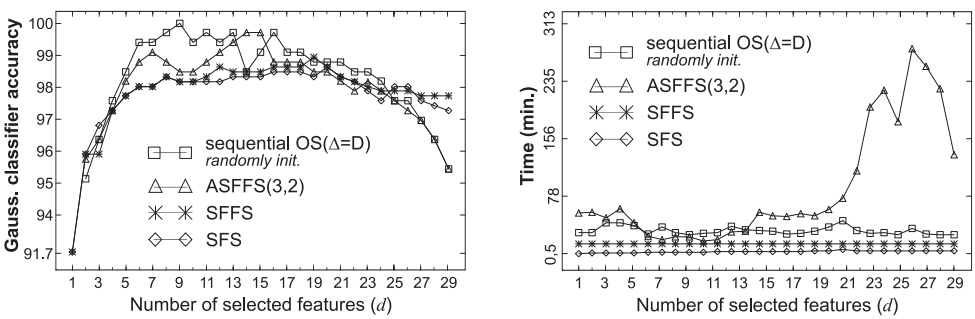

Fig. 3. Comparison of sub-optimal methods on classification problem.

4.4. Recommendations for practitioners

Concerning our current experience, we can give the following recommendations. Floating Search can be considered the tool of the first choice. It is reasonably fast and yields generally very good results in all dimensions at once, often succeeding in finding the global optimum. The Oscillating Search becomes better choice whenever: 1) the highest quality of solution must be achieved but optimal methods are not applicable, or 2) a reasonable solution is to be found as quickly as possible,

or 3) numerical problems hinder the use of sequential methods, or 4) extreme problem dimensionality prevents any use of sequential methods, or 5) the search is to be performed in real-time systems. Especially when applied repeatedly with different random starting points the Oscillating Search shows outstanding potential to overcome local extremes in favor of global optimum.

It should be stressed that, as opposed to B&B, the Floating Search and Oscillating Search methods are tolerant to deviations from monotonic behavior of feature selection criteria. It makes them particularly useful in conjunction with non-monotonic FS criteria like the error rate of a classifier (cf. Wrappers [11]), which according to a number of researchers seem to be the only legitimate criterion for feature subset evaluation.

Note: Floating and Oscillating Search source codes in C can be downloaded from `http://ro.utia.cas.cz/dem.html`.

5. MIXTURE BASED FEATURE SELECTION FOR CLASSIFICATION

All previous approaches depend on criteria that can not be evaluated without certain assumptions about the data (e. g., standard criteria are often available for Gaussian distribution only). If nothing is known about the data, or multi-modal or otherwise complex distribution can be expected, the mixture-based modelling may be a better choice.

Following the statistical approach to pattern recognition, we assume that a pattern represented by a feature vector $\boldsymbol{x} = (x_1, \ldots, x_D)^T$ in D-dimensional feature space is to be classified into one of a finite set of C different classes $\{\omega_1, \ldots, \omega_C\}$. A pattern $\boldsymbol{x}$ belonging to class ω_j is viewed as an observation of a random vector drawn randomly according to the known class conditional probability density function (pdf) $p(\boldsymbol{x}|\omega_j)$ and the respective *a priori* probability $P(\omega_j)$. We can say, that a tested pattern $\boldsymbol{x}$ can be optimally classified using Bayes classification rule based on the knowledge of $P(c_j)p(\boldsymbol{x}|\omega_j)$ for each class ω_j as follows:
assign $\boldsymbol{x}$ to class ω_l if

$$\omega_l = \arg\max_{j=1,\ldots,C} \frac{P(\omega_j)p(\boldsymbol{x}|\omega_j)}{p(\boldsymbol{x})}, \quad l \in \{1, \ldots, C\}, \quad p(\boldsymbol{x}) = \sum_{j=1}^{C} P(\omega_j)p(\mathbf{x}|\omega_j), \qquad (2)$$

where $p(\boldsymbol{x})$ is unconditional pdf of $\boldsymbol{x}$.

In practice, there appear situations, where the class conditional pdf is unknown and only a training set of labeled patterns is available. Finding a proper pdf estimate has a crucial impact to successful classification. Often simple models, such as a single Gaussian distribution, can effectively represent patterns but a more general model, such as a finite mixture model, must be used to approximate more complex pdfs; arbitrarily complex pdfs can be approximated using finite mixture models. For details on mixture models see e. g., [18].

In our papers [21, 24, 25], we proposed a mixture based classification with global feature selection by casting it as an estimation problem, thus avoiding any combinatorial search. Instead of selecting a subset of features, we estimate a set of binary quantities, one for each feature. This estimation is carried out by an Expectation-Maximization (EM) algorithm derived for the task.

5.1. Finite mixture model

It is said that D-dimensional random vector $\mathbf{x}$ follows in class ω_j a finite mixture distribution with M_j components if its pdf can be written as

$$p(\mathbf{x}|\omega_j) = \sum_{m=1}^{M_j} \alpha_m^j p_m(\mathbf{x}|\theta_m^j) \ , \tag{3}$$

where α_m^j is the *mixing probability* for the mth mixture component within the class ω_j, $(\alpha_m^j \geq 0, m = 1, \ldots, M_j, \sum_{m=1}^{M_j} \alpha_m^j = 1)$, θ_m^j is the set of parameters defining the mth component, and $\{\theta_1^j, \ldots, \theta_{M_j}^j, \alpha_1^j, \ldots, \alpha_{M_j}^j\}$ is the complete set of parameters specifying the mixture model (3). This model assumes that each mixture component has a multivariate distribution with its own parameters.

A simplification of the model (3) consists of assuming that given the mixture component label, the features are conditionally statistically independent. In this case the jth class conditional pdf can be rewritten as

$$p(\mathbf{x}|\omega_j) = \sum_{m=1}^{M_j} \alpha_m^j \prod_{i=1}^{D} f(x_i|\theta_{mi}^j) \ , \tag{4}$$

where $f(x_i|\theta_{mi}^j)$ is the pdf of the feature x_i in the mth mixture component within the class ω_j; θ_{mi}^j is the set of parameters of the component m corresponding to feature x_i. Therefore, the class conditional densities are modeled as a mixture of independent probabilistic models. This model has been considered e. g., by McLachlan in [17].

5.2. Mixture model with feature significancy

In this section we introduce the concept of *feature significancy* for the mixture structure of the class conditional probability density function.

Assume now that some features are not significant for the mixture structure of the pdf defined in (4), in the following sense: if feature x_i is *not significant*, then its distribution is common to all mixture components. It means that any specific univariate function $f(x_i|\theta_{mi}^j)$ in the model (4) is substituted by the "background" density, denoted by $f_0(x_i|\theta_{0i})$, whenever the feature x_i is not significant. Let $\Phi = (\phi_1, \ldots, \phi_D) \in \{0, 1\}^D$ be a set of binary parameters, such that $\phi_i = 1$ if the feature x_i is *significant* and $\phi_i = 0$, otherwise. In this way the binary parameters ϕ_i can be looked upon as *control variables* due to that the complexity and the structure of the mixture (4) can be controlled by means of that parameters. Then the class conditional pdf defined in (4) can be rewritten as

$$p(\boldsymbol{x}|\alpha_j, \theta_j, \theta_0, \Phi) = \sum_{m=1}^{M_j} \alpha_m^j \prod_{i=1}^{D} [f_0(x_i|\theta_{0i})^{1-\phi_i} f(x_i|\theta_{mi}^j)^{\phi_i}] \tag{5}$$

$$\alpha_j = (\alpha_1^j, \ldots, \alpha_{M_j}^j), \quad \theta_j = (\theta_1^j, \ldots, \theta_{M_j}^j), \quad \theta_0 = (\theta_{01}, \ldots, \theta_{0D}), \quad \Phi = (\phi_1, \ldots, \phi_D).$$

The novel characteristic of the mixture model defined in (5) are the *control variables* ϕ_i. When $\phi_i = 1$, it means that the feature x_i is significant and should be modeled by the individual mixture components. If $\phi_i = 0$, the feature x_i is useless and should be represented by a common component.

5.3. Feature significancy measures for classification

In order to optimize the accuracy of the Bayes classification rule (2), the EM algorithm is used to find the unknown parameters $\{\alpha_j, \theta_j, \theta_0, \Phi\}$, $j = 1, \ldots, C$ of the model (5) under the following assumptions:

- the training set of labeled samples is available;
- the univariate density functions f and f_0 belong to the family of univariate Gaussian densities;
- density function f_0 is common to all the classes $\omega_j,\ j = 1, \ldots, C$.

The criterion we use for measuring the error resulting from approximating the true probability density function $p(\boldsymbol{x}|\omega_j)$ by $p(\boldsymbol{x}|\alpha_j, \theta_j, \theta_0, \Phi)$ for all $j = 1, \ldots, C$ is a mixture, in the true proportions $P(\omega_1), \ldots, P(\omega_C)$, of the Kullback–Leibler distances between the true and the postulated class conditional pdf of $\boldsymbol{x}$.

Given the approximations of class conditional pdf's it can be easily seen that the "background" pdf f_0 may be reduced in the inequality in the Bayes plug-in classification rule and we may classify the tested pattern $\boldsymbol{x}$ into one of C classes according to only d features $x_{i_1}, \ldots, x_{i_d}$, where d is the required number of the selected features and $\{i_1, \ldots, i_d\}$ is the permutation of $\{1, \ldots, D\}$. We call this classifier in the reduced feature space the pseudo-Bayes classifier.

The proposed model (5) is suitable especially if the underlying distributions of some or all given classes are multi-modal or they conceal the existence of subpopulations. It allows to develop two FS methods: 'approximation' method determines those features which produce a set of approximations to the class probability density functions which is best in the sense of minimizing the mixture of Kullback–Leibler distances; 'divergence' method identifies those features that are most useful, in the sense of maximizing the the mixture of Kullback J-divergence, in describing differences between two possible classes.

Our approach to FS has the following features:

- it yields the feature subset of required size, optimizing the used criterion, without involving any search procedure;
- it provides a pseudo-Bayes plug-in classification rule employing the selected features.

Law et al. [13] (and references therein), Graham and Miller [6] have been inspired by the main idea of our approach to FS. They proposed a solution to the FS problem in unsupervised learning.

6. FEATURE SELECTION TOOLBOX

The Feature Selection Toolbox (FST) software [31] has been serving as a platform for data testing, feature selection, approximation-based modelling of data, classification and mostly testing newly developed methods. It is used basically for pattern recognition purposes. However, we used it for solving decision making problems in economics and in other application fields as well.

A rather simple user interface was constructed upon a strong functional kernel. Most of results are generated in the form of textual protocol into the Console window. Numerical results may be collected in tables and used for generating graphs. Data may be displayed in a 2D projection.

6.1. Mixture-based feature selection for classification task example

We used the following real data-sets:

— 2-class, 15-dimensional *speech* data representing words "yes" and "no" obtained from the British Telecom; classes are separable with great difficulty.

— 2-class, 30-dimensional *mammogram* data (see Section 3.3).

Using the FST we compared the performance of gaussian classifier to the pseudo-Bayes classifier, defined especially for use with multimodal data, and defined in relation to "approximation" and "divergence" methods (c.f. Section 5). Table 1 illustrates the potential of the approximation model based classifiers. However, it also illustrates the necessity of experimenting to find a suitable number of components (the issue is discussed, e. g., in Sardo [26]).

Table 1. Error rates [%] of different classifiers with different parameters. The 'gauss' column holds results of a gaussian classifier. Other columns hold results obtained using the 'approximation' method (in this case the 'divergence' method yielded the same results). (Note: 5c means 5 components of mixture, etc.)

		gauss	*approx.* 1c	*approx.* 5c	*approx.* 10c	*approx.* 20c
speech	(random init.)	8.39	21.61	7.58	9.19	9.03
data	(dogs & rabbits init.)	–	21.61	7.42	6.45	8.39
mammo	(random init.)	5.96	5.26	5.26	5.96	4.56
data	(dogs & rabbits init.)	–	5.26	5.26	5.96	5.96

The results were computed on the full set of features. In case of the "approximation" and "divergence" methods the algorithms were initialized randomly (1st row) by means of the "dogs & rabbits" cluster analysis (McKenzie et al. [16]) pre-processor (2nd row). Classifiers were trained on the first half of the dataset and tested on the second half.

Table 1 demonstrates the potential of mixture approximation methods – with 5 mixture components (see column approx.5c) for the "speech" data and 1, 5 or 20 components for the "mammo" data. The underlying data structure has been modeled precisely enough to achieve better classification rate when compared to the gaussian classifier. Second row for each data contains approximation and divergence method results after preliminary initialization by means of the "dogs and rabbits" clustering method.

7. CONCLUSIONS AND FUTURE WORK

The current state of art in feature selection based dimensionality reduction for modeling and decision problems of classification type have been over-viewed. A number

of recent feature subset search strategies have been reviewed and compared with emphasis put on methods developed within the Institute of Information Theory and Automation. Recent developments of B&B based algorithms for optimal search led to considerable improvements of the speed of search. Nevertheless, the principal exponential nature of optimal search remains and will remain one of key factors motivating the development of sub-optimal strategies. Among the family of sequential search algorithms the Floating and Oscillating search methods deserve particular attention. Two alternative feature selection methods based on mixture modelling have been presented. They are suitable for cases, when no *a priori* information on underlying probability structures is known. Among the most recent developments the hybrid FS methods deserve particular attention. This is reflected in our current field of interest [34]. In the future we intend to "hybridize" other search methods in a similar way and to investigate in detail the hybrid behavior of different combinations of various probabilistic measures and learning methods.

Many of recent feature selection methods have been implemented in Feature Selection Toolbox. The software has been used to demonstrate the differences between different criteria and differently selected feature subsets. The importance of feature selection for classification performance has been clearly shown as well.

ACKNOWLEDGEMENT

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under projects 2C06019 and 1M0572 DAR, by the EC project FP6-507752 MUSCLE, by the Grant Agency of the Academy of Sciences of the Czech Republic under grant No. A2075302, and by the Czech Science Foundation under grant No. 102/07/1594.

(Received June 29, 2006.)

REFERENCES

[1] S. Das: Filters, wrappers and a boosting-based hybrid for feature selection. In: Proc. 18th Internat. Conference Machine Learning, 2001, pp. 74–81.

[2] M. Dash, K. Choi, P. Scheuermann, and H. Liu: Feature selection for clustering – a Filter solution. In: Proc. Second Internat. Conference Data Mining, 2002, pp. 15–122.

[3] P. A. Devijver and J. Kittler: Pattern Recognition: A Statistical Approach. Prentice-Hall, Englewood Cliffs, NJ 1982.

[4] F. J. Ferri, P. Pudil, M. Hatef, and J. Kittler: Comparative study of techniques for large-scale feature selection. In: Pattern Recognition in Practice IV (E. S. Gelsema and L. N. Kanal, eds.), Elsevier Science B.V., 1994, pp. 403–413.

[5] K. Fukunaga: Introduction to Statistical Pattern Recognition. Academic Press, New York 1990.

[6] M. W. Graham and D. J. Miller: Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection. IEEE Trans. Signal Process. *54* (2006), 4, 1289–1303.

[7] R. Hodr, J. Nikl, B. Řeháková, A. Veselý, and J. Zvárová: Possibilities of a prognostic assessment quoad vitam in low birth weight newborns. Acta Facult. Med. Univ. Brunesis *58* (1977), 345–358.

[8] X. Chen: An improved branch and bound algorithm for feature selection. Pattern Recognition Lett. *24* (2003), 12, 1925–1933.

[9] A. K. Jain and D. Zongker: Feature selection: Evaluation, application and small sample performance. IEEE Trans. Pattern Anal. Mach. Intell. *19* (1997), 2, 153–158.
[10] A. K. Jain, R. P. W. Duin, and J. Mao: Statistical pattern eecognition: A review. IEEE Trans. Pattern Anal. Mach. Intell. *22* (2000), 2, 4–37.
[11] R. Kohavi and G. H. John: Wrappers for feature subset selection. Artificial Intelligence *97* (1997), 1–2, 273–324.
[12] M. Kudo and J. Sklansky: Comparison of algorithms that select features for pattern classifiers. Pattern Recognition *33* (2000), 1, 25–41.
[13] M. H. Law, M. A. T. Figueiredo, and A. K. Jain: Simultaneous feature selection and clustering using mixture models. IEEE Trans. Pattern Anal. Mach. Intell. *26* (2004), 1154–1166.
[14] H. Liu and L. Yu: Toward integrating feature selection algorithms for classification and clustering. IEEE Trans. Knowledge Data Engrg. *17* (2005), 491–502.
[15] H. A. Mayer, P. Somol, R. Huber, and P. Pudil: Improving statistical measures of feature subsets by conventional and evolutionary approaches. In: Proc. 3rd IAPR Internat. Workshop on Statistical Techniques in Pattern Recognition, Alicante 2000, pp. 77–81.
[16] P. McKenzie and M. Alder: Initializing the EM Algorithm for Use in Gaussian Mixture Modelling. University of Western Australia, 1994.
[17] G. J. McLachlan: Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York 1992.
[18] G. J. McLachlan and D. Peel: Finite Mixture Models. Wiley, New York 2000.
[19] P. M. Murphy and D. W. Aha: UCI Repository of Machine Learning Databases [ftp.ics.uci.edu]. University of California, Depart ment of Information and Computer Science, Irvine 1994.
[20] P. M. Narendra and K. Fukunaga: A branch and bound algorithm for feature subset selection. IEEE Trans. Computers *26* (1977), 917–922.
[21] J. Novovičová, P. Pudil, and J. Kittler: Divergence based feature selection for multimodal class densities. IEEE Trans. Pattern Anal. Mach. Intell. *18* (1996), 2, 218–223.
[22] J. Novovičová and P. Pudil: Feature selection and classification by modified model with latent structure. In: Dealing With Complexity: Neural Network Approach, Springer–Verlag, Berlin 1997, pp. 126–140.
[23] P. Pudil, J. Novovičová, and J. Kittler: Floating search methods in feature selection. Pattern Recognition Lett. *15* (1994), 11, 1119–1125.
[24] P. Pudil, J. Novovičová, and J. Kittler: Feature selection based on approximation of class densities by finite mixtures of special type. Pattern Recognition *28* (1995), 1389–1398.
[25] P. Pudil, J. Novovičová, and J. Kittler: Simultaneous learning of decision rules and important attributes for classification problems in image analysis. Image Vision Computing *12* (1994), 193–198.
[26] L. Sardo and J. Kittler: Model complexity validation for PDF estimation using Gaussian mixtures. In: Proc. 14th Internat. Conference on Pattern Recognition, Vol. 2, 1998, pp. 195–197.
[27] M. Sebban and R. Nock: A Hybrid filter/wrapper approach of feature selection using information theory. Pattern Recognition *35* (2002), 835–846.
[28] W. Siedlecki and J. Sklansky: On automatic feature selection. Internat. J. Pattern Recognition Artif. Intell. *2* (1988), 2, 197–220.
[29] P. Somol, P. Pudil, J. Novovičová, and P. Paclík: Adaptive floating search methods in feature selection. Pattern Recognition Lett. *20* (1999), 11 – 13, 1157–1163.
[30] P. Somol and P. Pudil: Oscillating search algorithms for feature selection. In: Proc. 15th IAPR Internat. Conference on Pattern Recognition, 2000, pp. 406–409.

[31] P. Somol and P. Pudil: Feature Selection Toolbox. Pattern Recognition *35* (2002), 12, 2749–2759.
[32] P. Somol, P. Pudil. and J. Kittler: Fast branch & bound algorithms for optimal feature selection. IEEE Trans. Pattern Anal. Mach. Intell. *26* (2004), 7, 900–912.
[33] P. Somol, P. Pudil, and J. Grim: On prediction mechanisms in fast branch & bound algorithms. In: Lecture Notes in Computer Science *3138*, Springer–Verlag, Berlin 2004, pp. 716–724.
[34] P. Somol, J. Novovičová, and P. Pudil: Flexible-hybrid sequential floating search in statistical feature selection. In: Lecture Notes in Computer Science 4109, Springer–Verlag, Berlin 2006, pp. 632–639.
[35] S. Theodoridis and K. Koutroumbas: Pattern Recognition. Second edition. Academic Press, New York 2003.
[36] Z. Wang, J. Yang, and G. Li: An improved branch & bound algorithm in feature selection. In: Lecture Notes in Computer Science 2639, Springer, Berlin 2003, pp. 549–556.
[37] A. Webb: Statistical Pattern Recognition. Second edition. Wiley, New York 2002.
[38] B. Yu and B. Yuan: A more efficient branch and bound algorithm for feature selection. Pattern Recognition *26* (1993), 883–889.
[39] L. Yu and H. Liu: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: Proc. 20th Internat. Conf. Machine Learning, 2003, pp. 856–863.
[40] J. Zvárová a J. Benda: Systém programů TIBIS. Ústav hematologie a krevní transfuze, Praha 1975 (in Czech).
[41] J. Zvárová, A. Perez, J. Nikl, and R. Jiroušek: Data reduction in computer-aided medical decision-making. In: MEDINFO 83 (J. H. van Bemmel, M. J. Ball, and O. Wigertz, eds.), North Holland, Amsterdam 1983, pp. 450–453.
[42] J. Zvárová and M. Studený: Information theoretical approach to constitution and reduction of medical data. Internat. J. Medical Informatics *45* (1997), 1 – 2, 65–74.

Petr Somol, Institute of Information Theory and Automation – Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 4, 182 08 Praha 8 and Faculty of Management, University of Economics, Praha. Czech Republic.

Jana Novovičová, Institute of Information Theory and Automation – Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 4, 182 08 Praha 8 and Faculty of Management, University of Economics, Praha. Czech Republic.

Pavel Pudil, Faculty of Management, University of Economics, Praha, 377 01 Jindřichův Hradec, Czech Republic, and Institute of Information Theory and Automation – Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 4, 182 08 Praha 8. Czech Republic.

e-mails: somol, novovic@utia.cas.cz pudil@fm.vse.cz