Proceedings of the Sixth IASTED International Conference
Advances in Computer Science and Engineering (ACSE 2010)
March 15-17, 2010 Sharm El-Sheikh, EGYPT

IMPROVING SEQUENTIAL FEATURE SELECTION METHODS' PERFORMANCE BY MEANS OF HYBRIDIZATION

Petr Somol and Jana Novovičová
Department of Pattern Recognition
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
CZ18208 Prague 8
email: {somol, novovic}@utia.cas.cz

Pavel Pudil
Faculty of Management
Prague University of Economics
Jarošovská 1117/II
CZ37701, Jindřichův Hradec
email: pudil@fm.vse.cz

ABSTRACT
In this paper we propose the general scheme of defining hybrid feature selection algorithms based on standard sequential search with the aim to improve feature selection performance, especially on high-dimensional or large-sample data. We show experimentally that "hybridization" has not only the potential to dramatically reduce FS search time, but in some cases also to actually improve classifier generalization, i.e., its classification performance on previously unknown data.

KEY WORDS
Feature selection, sequential search, hybrid methods, classification performance, subset search, statistical pattern recognition

1 Introduction

A broad class of decision-making problems can be solved by *learning approach*. This can be a feasible alternative when neither an analytical solution exists nor the mathematical model can be constructed. In these cases the required knowledge can be gained from the past data which form the so-called learning or training set. Then the formal apparatus of statistical pattern recognition can be used to learn the decision-making. The first and essential step of statistical pattern recognition is to solve the problem of feature selection (FS) or more generally dimensionality reduction (DR). The problem of effective feature selection – especially in case of high-dimensional and large-sample problems – will be of primary focus in this paper.

1.1 Feature Subset Selection

Given a set Y of $|\mathrm{Y}|$ features, let us denote $\mathcal{X}_d$ the set of all possible subsets of size d, where d represents the desired number of features. Let $J(\mathrm{X})$ be a criterion function that evaluates feature subset $\mathrm{X} \in \mathcal{X}_d$. Without any loss of generality, let us consider a higher value of J to indicate a better feature subset. Then the feature selection problem can be formulated as follows: Find the subset $\tilde{X}_d$ for which

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

Assuming that a suitable criterion function has been chosen to evaluate the effectiveness of feature subsets, feature selection is reduced to a search problem that detects an optimal feature subset based on the selected measure. Note that the choice of d may be a complex issue depending on problem characteristics, unless the d value can be optimized as part of the search process.

An overview of various aspects of feature selection can be found in [1], [2], [3], [4]. Many existing feature selection algorithms designed with different evaluation criteria can be categorized as *filter* [5], [6] *wrapper* [7], *hybrid* [8], [9], [10] or *embedded* [11], [12], [13] or [14], [15]. Filter methods are based on performance evaluation functions calculated directly from the training data such as *distance, information, dependency, and consistency,* [16, 1] and select features subsets without involving any learning algorithm. Wrapper methods require one predetermined learning algorithm and use its estimated performance as the evaluation criterion. They attempt to find features better suited to the learning algorithm aiming to improve performance. 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.

In the following we will investigate the hybrid approach to FS that combines the advantages of 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. In Section 2 an overview of sequential search FS methods is given. We distinguish methods that require the subset size to be specified by user (Sect. 2.1 to 2.6) from methods capable of subset size optimization (Sect. 2.7). The general scheme of hybridization is introduced in Sect. 3. Section 4 describes the performed experiments and Sect. 5 summarizes and concludes the paper.

2 Sub-Optimal Search Methods

Provided a suitable FS criterion function (cf. [16]) is available, the only tool needed is the search algorithm that gen-

erates a sequence of subsets to be tested. Despite the advances in optimal search ([17], [18]), for larger than moderate-sized problems we have to resort to sub-optimal methods. Very large number of various methods exists. The FS framework includes approaches that take use of evolutionary (genetic) algorithms ([19]), tabu search ([20]), or ant colony ([21]). In the following we present a basic overview over several tools that are useful for problems of varying complexity, based mostly on the idea of sequential search (Section 2.2 [16]). Finally, we show a general way of defining *hybrid* versions of sequential FS algorithms.

An integral part of any FS process is the decision about the number of features to be selected. Determining the correct subspace dimensionality is a difficult problem beyond the scope of this chapter. Nevertheless, in the following we will distinguish two types of FS methods: d-parametrized and d-optimizing. Most of the available methods are d-parametrized, i.e., they require the user to decide what cardinality should the resulting feature subset have. In Section 2.7 a d-optimizing procedure will be described, that optimizes both the feature subset size and its contents at once, provided the suitable criterion is available (classifier accuracy in wrappers can be used while monotonic probabilistic measures can not).

2.1 Best Individual Feature

The Best Individual Feature (BIF) approach is the simplest approach to FS. Each feature is first evaluated individually using the chosen criterion. Subsets are then selected simply by choosing the best individual features. This approach is the fastest but weakest option. It is often the only applicable approach to FS in problems of very high dimensionality. BIF is standard in text categorization ([22], [23]), genetics ([24], [25]) etc. BIF may be preferable in other types of problems to overcome FS stability problems. However, more advanced methods that take into account relations among features are likely to produce better results. Several of such methods are discussed in the following.

2.2 Sequential Search Framework

To simplify further discussion let us focus only on the family of sequential search methods. Most of the known sequential FS algorithms share the same "core mechanism" of adding and removing features to/from a current subset. The respective algorithm steps can be described as follows (for the sake of simplicity we consider only non-generalized algorithms that process one feature at a time only):

Definition 1 *For a given current feature set* X_d*, let* f^+ *be the feature such that*

$$f^+ = \arg \max_{f \in \mathrm{Y} \setminus \mathrm{X}_d} J^+(\mathrm{X}_d, f) \ , \tag{2}$$

where $J^+(\mathrm{X}_d, f)$ *denotes the criterion function used to evaluate the subset obtained by adding* f *(*$f \in \mathrm{Y} \setminus \mathrm{X}_d$*) to* X_d*. Then we shall say that* $ADD(\mathrm{X}_d)$ *is an operation of adding feature* f^+ *to the current set* X_d *to obtain set* X_{d+1} *if*

$$ADD(\mathrm{X}_d) \equiv \mathrm{X}_d \cup \{f^+\} = \mathrm{X}_{d+1}, \quad \mathrm{X}_d, \mathrm{X}_{d+1} \subset \mathrm{Y}. \tag{3}$$

Definition 2 *For a given current feature set* X_d*, let* f^- *be the feature such that*

$$f^- = \arg \max_{f \in \mathrm{X}_d} J^-(\mathrm{X}_d, f) \ , \tag{4}$$

where $J^-(\mathrm{X}_d, f)$ *denotes the criterion function used to evaluate the subset obtained by removing* f *(*$f \in \mathrm{X}_d$*) from* X_d*. Then we shall say that* $RMV(\mathrm{X}_d)$ *is an operation of removing feature* f^- *from the current set* X_d *to obtain set* X_{d-1} *if*

$$RMV(\mathrm{X}_d) \equiv \mathrm{X}_d \setminus \{f^-\} = \mathrm{X}_{d-1}, \quad \mathrm{X}_d, \mathrm{X}_{d-1} \subset \mathrm{Y}. \tag{5}$$

In order to simplify the notation for a repeated application of FS operations we introduce the following useful notation

$$\begin{aligned} \mathrm{X}_{d+2} &= ADD(ADD(\mathrm{X}_d)) = ADD^2(\mathrm{X}_d) \ , \\ \mathrm{X}_{d-2} &= RMV(RMV(\mathrm{X}_d)) = RMV^2(\mathrm{X}_d) \ , \end{aligned} \tag{6}$$

and more generally

$$\mathrm{X}_{d+\delta} = ADD^\delta(\mathrm{X}_d), \quad \mathrm{X}_{d-\delta} = RMV^\delta(\mathrm{X}_d) \ . \tag{7}$$

Note that in standard sequential FS methods $J^+(\cdot)$ and $J^-(\cdot)$ stand for

$$\begin{aligned} J^+(\mathrm{X}_d, f) &= J(\mathrm{X}_d \cup \{f\}), \\ J^-(\mathrm{X}_d, f) &= J(\mathrm{X}_d \setminus \{f\}) \ , \end{aligned} \tag{8}$$

where $J(\cdot)$ is either a filter- or wrapper-based criterion function ([26]) to be evaluated on the subspace defined by the tested feature subset.

2.3 Simplest Sequential Selection

The basic feature selection approach is to build up a subset of required number of features incrementally starting with the empty set (*bottom-up* approach) or to start with the complete set of features and remove redundant features until d features retain (*top-down* approach). The simplest (among recommendable choices) yet widely used *sequential forward (or backward) selection* methods, SFS and SBS ([27], [16]), iteratively add (remove) one feature at a time so as to maximize the intermediate criterion value until the required dimensionality is achieved.

SFS (*Sequential Forward Selection*) yielding a subset of d features:

1. $\mathrm{X}_d = ADD^d(\emptyset)$.

As many other of the earlier sequential methods both SFS and SBS suffer from the so-called nesting of feature subsets which significantly deteriorates optimization ability. The first attempt to overcome this problem was to employ either the Plus-l-Take away-r (also known as (l, r)) or generalized (l, r) algorithms ([16]) which involve successive augmentation and depletion process. The same idea in a principally extended and refined form constitutes the basis of Floating Search.

2.4 Sequential Floating Search

The Sequential Forward Floating Selection (SFFS) ([28]) 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.

SFFS (*Sequential Forward Floating Selection*) yielding a subset of d features, with optional search-restricting parameter $\Delta \in [0, D-d]$:

1. Start with $\mathrm{X}_0 = \emptyset$, $k = 0$.
2. $\mathrm{X}_{k+1} = ADD(\mathrm{X}_k)$, $k = k + 1$.
3. Repeat $\mathrm{X}_{k-1} = RMV(\mathrm{X}_k)$, $k = k - 1$ as long as it improves solutions already known for the lower k.
4. If $k < d + \Delta$ go to step 2.

A detailed formal description of this now classical procedure can be found in [28]. The backward counterpart to SFFS is the Sequential Backward Floating Selection (SBFS). Its principle is analogous.

Floating search algorithms can be considered 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 practical problems. The idea of Floating search has been futher developed in [29] and [30].

2.5 Oscillating Search

The more recent Oscillating Search (OS) ([31]) can be considered a "meta" 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 algorithms. 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 of 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 a user-specified *limit* Δ. The course of Oscillating Search is illustrated in comparison to SFS and SFFS in Fig. 1. Every OS algorithm

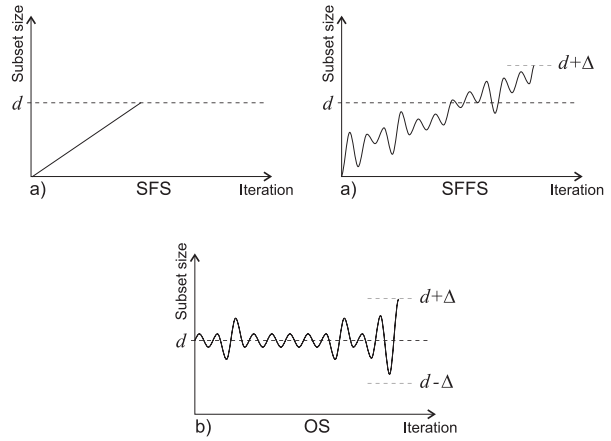

Figure 1. *Graphs demonstrate the course of d-parametrized search algorithms: a) Sequential Forward Selection, b) Sequential Forward Floating Selection, c) Oscillating Search.*

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.

OS (*Oscillating Search*) yielding a subset of d features, with optional search-restricting parameter $\Delta \geq 1$:

1. Start with initial set X_d of d features. Set cycle depth to $o = 1$.
2. Let $\mathrm{X}_d^{\downarrow} = ADD^o(RMV^o(\mathrm{X}_d))$.
3. If $\mathrm{X}_d^{\downarrow}$ better than X_d, let $\mathrm{X}_d = \mathrm{X}_d^{\downarrow}$, let $o = 1$ and go to step 2.
4. Let $\mathrm{X}_d^{\uparrow} = RMV^o(ADD^o(\mathrm{X}_d))$.

5. If $\mathrm{X}_d^{\uparrow}$ better than X_d, let $\mathrm{X}_d = \mathrm{X}_d^{\uparrow}$, let $o = 1$ and go to step 2.

6. If $o < \Delta$ let $o = o + 1$ and go to step 2.

The generality of OS search concept allows to adjust the search for better speed or better accuracy (by adjusting Δ, redefining the initialization procedure or redefining ADD / RMV). As opposed to all sequential search procedures, OS does not waste time evaluating subsets of cardinalities too different from the target one. This "focus" improves the OS ability to find good solutions for subsets of given cardinality. 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 ($o = o + 1$). Though this improves the chance to get closer to the optimum, the trade-off between finding a better solution and computational time becomes more apparent. Consequently, 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:

- It may be looked upon as a universal tuning mechanism, being able to improve solutions obtained in other way.
- The randomly initialized OS is very fast, in case of very high-dimensional problems may become the only applicable alternative to BIF. For example, in document analysis ([32]) for search of the best 1000 words out of a vocabulary of 10000 all other sequential methods prove to be too slow.
- 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.
- 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.
- 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.

2.6 Experimental Comparison of d-Parametrized Methods

The d-parametrized sub-optimal FS methods as discussed in preceding sections 2.1 to 2.5 have been listed in the order of their speed-vs-optimization performance characteristics. The BIF is the fastest but worst performing method, OS offers the strongest optimization ability at the cost of slowest computation (although it can be adjusted differently). To illustrate this behavior we compare the output of BIF, SFS, SFFS and OS on a FS task in wrapper ([7]) setting.

The methods have been used to find best feature subsets for each subset size $d = 1, \ldots, 34$ on the *ionosphere* data (34 dim., 2 classes: 225 and 126 samples) from the UCI Repository ([33]). The dataset had been split to 80% train and 20% test part. FS has been performed on the training part using 10-fold cross-validation, in which 3-Nearest Neighbor classifier was used as FS criterion. BIF, SFS and SFFS require no parameters, OS had been set to repeat each search $15\times$ from different random initial subsets of given size, with $\Delta = 15$. This set-up is highly time consuming but enables avoiding many local extremes that would not be avoided by other algorithms.

Figure 2 shows the maximal criterion value obtained by each method for each subset size. It can be seen that the strongest optimizer in most of cases is OS, although SFFS falls behind just negligibly. SFS optimization ability is shown to be markedly lower, but still higher than that of BIF.

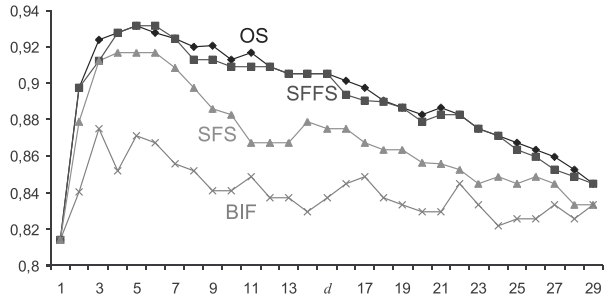

Figure 2. *Sub-optimal FS methods' optimization performance on 3-NN wrapper.*

Figure 3 shows how the optimized feature subsets perform on independent test data. From this perspective the differences between methods largely diminish. The effects of feature over-selection (over-fitting) affect the strongest optimizer – OS – the most. SFFS seems to be the most reliable method in this respect. SFS yields the best performance on independent data (to be denoted independent performance from now on) in this example. Note that although the highest optimized criterion values have been achieved for subsets of roughly 6 features, the best independent performance can be observed for subsets of roughly 7 to 13 features. The example thus illustrates well one of the key problems in FS – the difficulty to find subsets that generalize well, related to the problem of feature over-selection ([34]).

The speed of each tested method decreases with its complexity. BIF runs in linear time. Other methods run in polynomial time. SFFS runs roughly $10\times$ slower than SFS.

OS in the slow test setting runs roughly 10 to $100\times$ slower than SFFS. The speed penalty of more complex methods gets even more notable with increasing dimensionality and sample size.

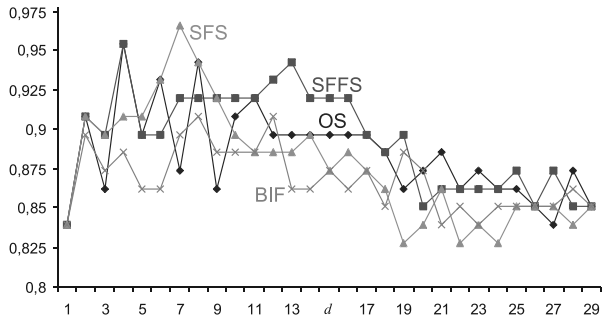

Figure 3. *Sub-optimal FS methods' performance verified using 3-NN on independent data.*

2.7 Dynamic Oscillating Search – The d-optimizing Method

The idea of Oscillating Search (Sect. 2.5) has been further extended in form of the Dynamic Oscillating Search (DOS) ([35]). The DOS algorithm can start from any initial subset of features (including empty set). Similarly to OS it repeatedly attempts to improve the current set by means of repeating oscillation cycles. However, the current subset size is allowed to change, whenever a new globally best solution is found at any stage of the oscillation cycle. Unlike other methods discussed in this chapter the DOS is thus a d-optimizing procedure.

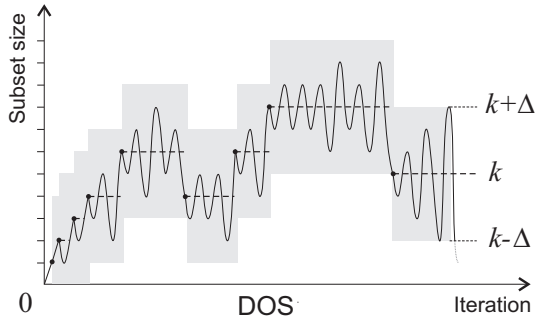

Figure 4. *The DOS course of search.*

The course of Dynamic Oscillating Search is illustrated in Fig. 4. See Fig. 1 for comparison with OS, SFFS and SFS. Similarly to OS the DOS terminates when the current cycle depth exceeds a user-specified *limit* Δ. The DOS also shares with OS the same advantages as listed in Sect. 2.5: the ability to tune results obtained in a different way, gradual result improvement, fastest improvement in initial search stages, etc.

DOS (*Dynamic Oscillating Search*) yielding a subset of optimized size k, with optional search-restricting parameter $\Delta \geq 1$:

1. Start with $\mathrm{X}_k = ADD(ADD(\emptyset))$, $k = 2$. Set cycle depth to $\delta = 1$.
2. Compute $ADD^\delta(RMV^\delta(\mathrm{X}_t))$; if any intermediate subset X_i, $i \in [k-\delta, k]$ is found better than X_k, let it become the new X_k with $k = i$, let $\delta = 1$ and restart step 2.
3. Compute $RMV^\delta(ADD^\delta(\mathrm{X}_t))$; if any intermediate subset X_j, $j \in [k, k+\delta]$ is found better than X_k, let it become the new X_k with $k = j$, let $\delta = 1$ and go to step 2.
4. If $\delta < \Delta$ let $\delta = \delta + 1$ and go to step 2.

In the course of search the DOS generates a sequence of solutions with ascending criterion values and, provided the criterion value does not decrease, decreasing subset size. The search time vs. closeness-to-optimum trade-off can thus be handled by means of pre-mature search interruption. The number of criterion evaluations is in the $\mathrm{O}(n^3)$ order of magnitude. Nevertheless, the total search time depends heavily on the chosen Δ value, on particular data and criterion settings, and on the unpredictable number of oscillation cycle restarts that take place after each solution improvement. Note: with monotonic criteria DOS yields always the full feature set. This behavior makes it unusable with many probabilistic distance measures (Bhattacharyya distance etc.). Nevertheless, DOS performs well with wrapper FS criteria (classifier accuracy).

3 Hybrid Algorithms – Improving Feature Selection Performance

Filter methods [26] for feature selection are general preprocessing algorithms that do not rely on any knowledge of the learning algorithm to be used. They are distinguished by specific evaluation criteria including distance, information, dependency. Since the filter methods apply independent evaluation criteria without involving any learning algorithm they are computationally efficient. Wrapper methods [26] require a predetermined learning algorithm instead of an independent criterion for subset evaluation. They search through the space of feature subsets using a learning algorithm, calculate the estimated accuracy of the learning algorithm for each feature before it can be added to or removed from the feature subset. It means, that learning algorithms are used to control the selection of feature subsets which are consequently better suited to the predetermined learning algorithm. Due to the necessity to train and evaluate the learning algorithm within the feature selection process, the wrapper methods are more computationally expensive than the filter methods.

The main advantage of filter methods is their speed and ability to scale to large data sets. A good argument for wrapper methods is that they tend to give superior performance. Their time complexity, however, may become prohibitive if problem dimensionality exceeds several dozen features. Moreover, wrappers are more prone to feature over-selection [34].

Hybrid FS algorithms can be defined easily to utilize the advantages of both filters and wrappers ([10], [1]). In the course of search, in each algorithm step the filter is used to reduce the number of candidates to be evaluated in wrapper. The scheme can be applied in any sequential FS algorithms (see Section 2) by replacing Definitions 1 and 2 by Definitions 3 and 4 as follows. For sake of simplicity let $J_F(.)$ denote the faster but for the given problem possibly less appropriate *filter* criterion, $J_W(.)$ denote the slower but more appropriate *wrapper* criterion. The *hybridization coefficient*, defining the proportion of feature subset evaluations to be accomplished by wrapper means, is denoted by $\lambda \in [0,1]$. In the following $\lceil\cdot\rceil$ denotes value rounding.

Definition 3 *For a given current feature set* X_d *and given* $\lambda \in [0,1]$*, let* Z^+ *be the set of candidate features*

$$\mathrm{Z}^+ = \{f_i : f_i \in \mathrm{Y} \backslash \mathrm{X}_d; i = 1, \ldots, \max\{1, \lceil \lambda \cdot |\mathrm{Y} \backslash \mathrm{X}_d| \rceil\}\} \tag{9}$$

such that

$$\begin{aligned} \forall f, g \in \mathrm{Y} \setminus \mathrm{X}_d, f \in \mathrm{Z}^+, g \notin \mathrm{Z}^+ \\ J_F^+(\mathrm{X}_d, f) \geq J_F^+(\mathrm{X}_d, g)\ , \end{aligned} \tag{10}$$

where $J_F^+(\mathrm{X}_d, f)$ *denotes the pre-filtering criterion function used to evaluate the subset obtained by adding* f *(*$f \in \mathrm{Y} \setminus \mathrm{X}_d$*) to* X_d*. Let* f^+ *be the feature such that*

$$f^+ = \arg\max_{f \in \mathrm{Z}^+} J_W^+(\mathrm{X}_d, f)\ , \tag{11}$$

where $J_W^+(\mathrm{X}_d, f)$ *denotes the main criterion function used to evaluate the subset obtained by adding* f *(*$f \in \mathrm{Z}^+$*) to* X_d*. Then we shall say that* $ADD_H(\mathrm{X}_d)$ *is an operation of adding feature* f^+ *to the current set* X_d *to obtain set* X_{d+1} *if*

$$ADD_H(\mathrm{X}_d) \equiv \mathrm{X}_d \cup \{f^+\} = \mathrm{X}_{d+1}, \quad \mathrm{X}_d, \mathrm{X}_{d+1} \subset \mathrm{Y}. \tag{12}$$

Definition 4 *For a given current feature set* X_d *and given* $\lambda \in [0,1]$*, let* Z^- *be the set of candidate features*

$$\mathrm{Z}^- = \{f_i : f_i \in \mathrm{X}_d; i = 1, \ldots, \max\{1, \lceil \lambda \cdot |\mathrm{X}_d| \rceil\}\} \tag{13}$$

such that

$$\forall f, g \in \mathrm{X}_d, f \in \mathrm{Z}^-, g \notin \mathrm{Z}^- \qquad J_F^-(\mathrm{X}_d, f) \geq J_F^-(\mathrm{X}_d, g)\ , \tag{14}$$

where $J_F^-(\mathrm{X}_d, f)$ *denotes the pre-filtering criterion function used to evaluate the subset obtained by removing* f *(*$f \in \mathrm{X}_d$*) from* X_d*. Let* f^- *be the feature such that*

$$f^- = \arg\max_{f \in \mathrm{Z}^-} J_W^-(\mathrm{X}_d, f), \tag{15}$$

where $J_W^-(\mathrm{X}_d, f)$ *denotes the main criterion function used to evaluate the subset obtained by removing* f *(*$f \in \mathrm{Z}^-$*) from* X_d*. Then we shall say that* $RMV_H(\mathrm{X}_d)$ *is an operation of removing feature* f^- *from the current set* X_d *to obtain set* X_{d-1} *if*

$$RMV_H(\mathrm{X}_d) \equiv \mathrm{X}_d \setminus \{f^-\} = \mathrm{X}_{d-1}, \quad \mathrm{X}_d, \mathrm{X}_{d-1} \subset \mathrm{Y}. \tag{16}$$

Note that in standard sequential FS methods $J_F^+(\cdot)$, $J_F^-(\cdot)$, $J_W^+(\cdot)$ and $J_W^-(\cdot)$ stand for

$$\begin{aligned} J_F^+(\mathrm{X}_d, f) &= J_F(\mathrm{X}_d \cup \{f\})\ , \\ J_F^-(\mathrm{X}_d, f) &= J_F(\mathrm{X}_d \setminus \{f\})\ , \\ J_W^+(\mathrm{X}_d, f) &= J_W(\mathrm{X}_d \cup \{f\})\ , \\ J_W^-(\mathrm{X}_d, f) &= J_W(\mathrm{X}_d \setminus \{f\})\ . \end{aligned} \tag{17}$$

The idea behind the proposed hybridization scheme is applicable in any of the sequential feature selection methods discussed in Sections 2.3 to 2.7 and can be also expressed in a simplified way as follows:

Operation $ADD_H(\mathrm{X}_d)$ adds a feature to a working subset of d features, X_d, to produce subset X_{d+1}, based on hybridized evaluation of feature subset merit (for simplicity denote $p = |\mathrm{Y} \setminus \mathrm{X}_d|$ and $q = \max\{1, \lceil \lambda \cdot p \rceil\}$):

1. *Pre-filtering*: For each candidate feature $f_i^+ \in \mathrm{Y} \setminus \mathrm{X}_d$, $i = 1, \ldots, p$ and the respective candidate subset $\mathrm{X}_d \cup \{f_i^+\}$ compute the value $\nu_i^+ = J_F(\mathrm{X}_d \cup \{f_i^+\})$.
2. *Main evaluation*: Keep only those q feature candidates $f_{i_j}^+$, $j = 1, \ldots, q$, that yielded q highest $\nu_{i_j}^+$ values. Evaluate the respective candidate subsets $\mathrm{X}_d \cup \{f_{i_j}^+\}$ by computing $\mu_j^+ = J_W(\mathrm{X}_d \cup \{f_{i_j}^+\})$, $j = 1, \ldots, q$.
3. Return $\mathrm{X}_{d+1} = \mathrm{X}_d \cup \{f_{i_{j_{max}}}^+\}$ where $\mu_{j_{max}}^+$ is the highest among all μ_j^+, $j = 1, \ldots, q$ values.

Operation $RMV_H(\mathrm{X}_d)$ removes a feature from a working subset of d features, X_d, to produce subset X_{d-1}, based on hybridized evaluation of feature subset merit (for simplicity denote $r = |\mathrm{X}_d|$ and $s = \max\{1, \lceil \lambda \cdot r \rceil\}$):

1. *Pre-filtering*: For each candidate feature $f_i^- \in \mathrm{X}_d$, $i = 1, \ldots, r$ and the respective candidate subset $\mathrm{X}_d \setminus \{f_i^-\}$ compute the value $\nu_i^- = J_F(\mathrm{X}_d \setminus \{f_i^-\})$.
2. *Main evaluation*: Keep only those s feature candidates $f_{i_j}^-$, $j = 1, \ldots, s$, that yielded s highest $\nu_{i_j}^-$ values. Evaluate the respective candidate subsets $\mathrm{X}_d \setminus \{f_{i_j}^-\}$ by computing $\mu_j^- = J_W(\mathrm{X}_d \setminus \{f_{i_j}^-\})$, $j = 1, \ldots, s$.
3. Return $\mathrm{X}_{d-1} = \mathrm{X}_d \setminus \{f_{i_{j_{max}}}^-\}$ where $\mu_{j_{max}}^-$ is the highest among all μ_j^-, $j = 1, \ldots, s$ values.

Table 1. Performance of hybridized FS methods with Bhattacharyya distance used as pre-filtering criterion and 5-NN performance as main criterion. *Madelon* data, 500-dim., 2 classes of 1000 and 1000 samples. 50% of data used for training by means of 10-fold cross-validation, 50% for independent testing using 5-NN.

Hybridization coeff. λ	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
d-optimizing FS Method	Dynamic Oscillating Search ($\Delta = 15$)										
FS result – criterion value	0.795	0.889	0.903	0.873	0.897	0.891	0.892	0.894	0.884	0.884	0.886
Independent accuracy	0.811	0.865	0.868	0.825	0.854	0.877	0.871	0.849	0.873	0.873	0.875
Determined subset size	8	27	19	19	19	18	23	13	13	13	16
Time (h:m)	0:01	6:20	14:18	8:33	17:50	18:34	13:47	4:49	3:14	3:23	9:13
d-parametrized FS Method	Oscillating Search (BIF initialized, $\Delta = 10$), subset size set in all cases to $d = 20$										
FS result – criterion value	0.812	0.874	0.887	0.891	0.879	0.902	0.891	0.899	0.889	0.891	0.884
Independent accuracy	0.806	0.859	0.869	0.853	0.855	0.864	0.856	0.853	0.857	0.86	0.858
Time (h:m)	0:09	6:25	1:10	2:18	4:05	6:44	9:06	14:16	10:32	10:17	12:57

When applied in sequential FS methods the described hybridization mechanism has several implications: 1) it makes possible to use wrapper based FS in considerably higher dimensional problems as well as with larger sample sizes due to reduced number of wrapper computations and consequent computational time savings, 2) it improves resistance to over-fitting when the used wrapper criterion tends to over-fit and the filter does not, and 3) for $\lambda = 0$ it reduces the number of wrapper criterion evaluations to the absolute minimum of one evaluation in each algorithm step. In this way it is possible to enable monotonic filter criteria to be used in d-optimizing setting, what would otherwise be impossible.

4 Experiments

We have conducted a series of experiments on data of various characteristics. We include low-dimensional low sample size *speech* data from British Telecom, 15-dim., 2 classes of 212 and 55 samples, and *wdbc* data from UCI Repository [33], 30-dim., 2 classes of 357 and 212 samples, moderate-dimensional high sample size *waveform* data [33], 40-dim., first 2 classes of 1692 and 1653 samples, as well as high-dimensional, high sample size data: *madelon* 500-dim., 2 classes of 1000 samples each form UCI Repository [33] and *musk* data [33], 166-dim., 2 classes of 1017 and 5581 samples.

For each data set we compare feature selection results of the d-parametrized Oscillating Search (OS) and the d-optimizing Dynamic Oscillating Search (DOS), the two methods representing some of the most effective subset search tools available. For OS the target subset size d is set manually to a constant value to be comparable to the d as yielded by DOS. In both cases the experiment has been performed for various values of the hybridization coefficient λ ranging from 0 to 1. In each hybrid algorithm the following feature selection criteria have been combined: (normal) Bhattacharyya distance for pre-filtering (filter criterion) and 5-Nearest Neighbor (5-NN) 10-fold cross-validated classification rate on validation data for final feature selection (wrapper criterion). Each resulting feature subset has been eventually tested using 5-NN on independent test data (50% of each dataset).

The results are collected in Tables 1 to 5. Note the following phenomena observable across all tables: 1) hybridization coefficient λ closer to 0 lead generally to lower computational time while λ closer to 1 leads to higher computational time, although there is no guarantee that lowering λ reduces search time (for counter-example see, e.g., Table 1 for $\lambda = 0.7$ or Table 2 for $\lambda = 0.4$), 2) low λ values often lead to results performing equally or better than pure wrapper results ($\lambda = 1$) on independent test data (see esp. Table 2), 3) d-optimizing DOS tends to yield higher criterion values than d-parametrized OS; in terms of the resulting performance on independent data the difference between DOS and OS shows much less notable and consistent, although DOS still shows to be better performing (compare the best achieved accuracy on independent data over all λ values in each Table), 4) it is impossible to predict the λ value for which the resulting classifier performance on independent data will be maximum (note in Table 1 $\lambda = 0.5$ for DOS and 0.2 for OS, etc.). The same holds for the maximum found criterion value (note in Table 1 $\lambda = 0.2$ for DOS and 0.5 for OS).

5 Conclusion

Based on an overview of the framework of sequential search methods we introduced the general scheme of defining hybridized versions of sequential feature selection algorithms. The main reason for defining hybrid feature selection algorithms is the possibility to take advantage of two different FS schemes, each of which being advantageous in different situations. We show experimentally that in the particular case of combining faster but weaker *filter* FS criteria with slow but possibly more appropriate *wrapper* FS criteria it is possible to achieve results comparable to that of wrapper-based FS but in filter-like time. Moreover, in some cases hybrid FS methods exhibit better ability to generalize than pure wrappers, i.e., they occassionally find feature

Table 2. Performance of hybridized FS methods with Bhattacharyya distance used as pre-filtering criterion and 5-NN performance as main criterion. *Musk* data, 166-dim., 2 classes of 1017 and 5581 samples. 50% of data used for training by means of 10-fold cross-validation, 50% for independent testing using 5-NN.

Hybridization coeff. λ	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
d-optimizing FS Method	Dynamic Oscillating Search ($\Delta = 15$)										
FS result – criterion value	0.968	0.984	0.985	0.985	0.985	0.985	0.986	0.985	0.986	0.985	0.985
Independent accuracy	0.858	0.869	0.862	0.872	0.863	0.866	0.809	0.870	0.861	0.853	0.816
Determined subset size	7	7	9	14	16	17	18	7	16	12	12
Time (h:m)	0:05	2:00	6:24	16:03	22:26	25:30	37:59	11:56	48:11	29:09	40:58
d-parametrized FS Method	Oscillating Search (BIF initialized, $\Delta = 10$), subset size set in all cases to $d = 20$										
FS result – criterion value	0.958	0.978	0.984	0.983	0.985	0.985	0.984	0.985	0.986	0.986	0.986
Independent accuracy	0.872	0.873	0.864	0.855	0.858	0.875	0.868	0.864	0.853	0.846	0.841
Time (h:m)	1:07	4:27	33:03	10:52	62:13	32:09	47:20	70:11	62:45	65:30	31:11

subsets that yield better classifier accuracy on independent test data.

The key advantage of evaluated hybrid methods – considerably reduced search time when compared to wrappers – effectively open new application fields for non-trivial feature selection. Previously it was often perceived impossible to apply sequential search with wrapper criteria to problems of higher dimensionality (roughly of hundreds of features). In our experiments we show that hybridization enables reasonable feature selection outcome for a 500-dimensional problem; higher-dimensional problems can be tackled as well, as the proportion between the number of performed slow and fast (stronger and weaker) FS criterion evaluation steps can be user-adjusted (by hybridization coefficient λ). It has been shown that the behavior of hybrid algorithms is very often advantageous in the sense that a considerable reduction of search time is often achieved at the cost of only negligible (or zero) decrease of resulting criterion value. The only problem stemming from hybridization is the necessity to choose a suitable value of the hybridization coefficient λ, while there is no analytical way of doing this optimally. Nevertheless, the meaning of λ on the scale from 0 to 1 is well understandable; lower values can be expected to yield results more filter-like while higher values yield results more wrapper-like. Values closer to 0 enable hybridized feature selection in (considerably) higher-dimensional problems than values closer to 1.

Remark: Some related source codes can be found at `http://ro.utia.cas.cz/dem.html`.

Acknowledgements

The work has been supported by grants of the Czech Ministry of Education 2C06019 ZIMOLEZ and 1M0572 DAR, and the GACR Nos. 102/08/0593 and 102/07/1594.

References

[1] H. Liu and L. Yu, Toward integrating feature selection algorithms for classification and clustering, *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, 491–502, 2005.

[2] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. (Norwell, MA, USA: Kluwer Academic Publishers, 1998).

[3] Special issue on variable and feature selection, *Journal of Machine Learning Research. http://www.jmlr.org/papers/special/feature.html*, 2003.

[4] A. Salappa, M. Doumpos, and C. Zopounidis, Feature selection algorithms in classification problems: An experimental evaluation, *Optimization Methods and Software*, vol. 22, no. 1, 199–212, 2007.

[5] L. Yu and H. Liu, Feature selection for high-dimensional data: A fast correlation-based filter solution, in *Proceedings of the 20th International Conference on Machine Learning*, 2003, 56–63.

[6] M. Dash, K. Choi, S. P., and H. Liu, Feature selection for clustering - a filter solution, in *Proceedings of the Second International Conference on Data Mining*, 2002, 115–122.

[7] R. Kohavi and G. John, Wrappers for feature subset selection, *Artificial Intelligence*, vol. 97, 273–324, 1997.

[8] S. Das, Filters, wrappers and a boosting-based hybrid for feature selection, in *Proc. of the 18th International Conference on Machine Learning*, 2001, 74–81.

[9] M. Sebban and R. Nock, A hybrid filter/wrapper approach of feature selection using information theory, *Pattern Recognition*, vol. 35, 835–846, 2002.

[10] P. Somol, J. Novovičová, and P. Pudil, Flexible-hybrid sequential floating search in statistical feature selection, in *Structural, Syntactic, and Statistical Pattern Recognition*, vol. LNCS 4109. Berlin / Heidelberg, Germany:

Table 3. Performance of hybridized FS methods with Bhattacharyya distance used as pre-filtering criterion and 5-NN performance as main criterion. *Speech* data, 15-dim., 2 classes of 212 and 55 samples. 50% of data used for training by means of 10-fold cross-validation, 50% for independent testing using 5-NN.

Hybridization coeff. λ	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
d-optimizing FS Method	Dynamic Oscillating Search ($\Delta = 15$)										
FS result – criterion value	0.935	0.935	0.941	0.945	0.945	0.948	0.946	0.948	0.948	0.948	0.948
Independent accuracy	0.918	0.918	0.932	0.928	0.928	0.921	0.922	0.924	0.924	0.924	0.929
Determined subset size	6	6	7	10	10	10	8	8	8	8	7
Time	4s	4s	8s	14s	18s	26s	27s	40s	45s	51s	42s
d-parametrized FS Method	Oscillating Search (BIF initialized, $\Delta = 10$), subset size set in all cases to $d = 6$										
FS result – criterion value	0.935	0.935	0.935	0.935	0.942	0.945	0.943	0.945	0.945	0.945	0.945
Independent accuracy	0.918	0.918	0.918	0.918	0.932	0.920	0.924	0.920	0.920	0.920	0.920
Time	3s	3s	3s	6s	9s	24s	15s	30s	18s	19s	34s

Table 4. Performance of hybridized FS methods with Bhattacharyya distance used as pre-filtering criterion and 5-NN performance as main criterion. *Waveform* data, 40-dim., first 2 classes of 1692 and 1653 samples. 50% of data used for training by means of 10-fold cross-validation, 50% for independent testing using 5-NN.

Hybridization coeff. λ	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
d-optimizing FS Method	Dynamic Oscillating Search ($\Delta = 15$)										
FS result – criterion value	0.912	0.913	0.921	0.922	0.927	0.927	0.928	0.927	0.926	0.934	0.934
Independent accuracy	0.919	0.912	0.906	0.904	0.909	0.909	0.909	0.901	0.903	0.909	0.909
Determined subset size	12	10	11	21	12	12	18	16	19	14	14
Time	1m	1m	3m	7m	8m	9m	17m	16m	17m	16m	18m
d-parametrized FS Method	Oscillating Search (BIF initialized, $\Delta = 10$), subset size set in all cases to $d = 15$										
FS result – criterion value	0.914	0.914	0.918	0.927	0.925	0.925	0.925	0.925	0.926	0.928	0.928
Independent accuracy	0.898	0.898	0.911	0.909	0.897	0.907	0.910	0.911	0.910	0.909	0.909
Time	19s	31s	2m	3m	7m	4m	5m	5m	6m	10m	11m

Table 5. Performance of hybridized FS methods with Bhattacharyya distance used as pre-filtering criterion and 5-NN performance as main criterion. *Wdbc* data, 30-dim., 2 classes of 357 and 212 samples. 50% of data used for training by means of 10-fold cross-validation, 50% for independent testing using 5-NN.

Hybridization coeff. λ	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
d-optimizing FS Method	Dynamic Oscillating Search ($\Delta = 15$)										
FS result – criterion value	0.919	0.919	0.926	0.926	0.961	0.961	0.961	0.961	0.961	0.961	0.961
Independent accuracy	0.930	0.930	0.933	0.933	0.944	0.944	0.944	0.944	0.944	0.944	0.944
Determined subset size	3	2	3	3	5	5	3	3	3	3	3
Time	1s	1s	1s	2s	7s	10s	11s	19s	26s	28s	26s
d-parametrized FS Method	Oscillating Search (BIF initialized, $\Delta = 10$), subset size set in all cases to $d = 8$										
FS result – criterion value	0.919	0.919	0.919	0.919	0.919	0.919	0.919	0.919	0.919	0.919	0.919
Independent accuracy	0.933	0.933	0.933	0.933	0.933	0.933	0.933	0.933	0.933	0.933	0.933
Time	1s	2s	2s	3s	4s	5s	6s	6s	7s	8s	8s

Springer-Verlag, 2006, 632–639.

[11] I. Guyon and A. Elisseeff, An introduction to variable and feature selection, *J. Mach. Learn. Res.*, vol. 3, 1157–1182, 2003.

[12] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, Eds., *Feature Extraction – Foundations and Applications*, ser. Studies in Fuzziness and Soft Computing vol. 207. (Physica-Verlag, Springer, 2006).

[13] I. Kononenko, Estimating attributes: Analysis and extensions of relief, in *ECML-94: Proc. European Conf. on Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1994, 171–182.

[14] P. Pudil, J. Novovičová, N. Choakjarernwanit, and J. Kittler, Feature selection based on approximation of class densities by finite mixtures of special type, *Pattern Recognition*, vol. 28, 1389–1398, 1995.

[15] J. Novovičová, P. Pudil, and J. Kittler, Divergence based feature selection for multimodal class densities, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 2, 218–223, 1996.

[16] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. (Englewood Cliffs, London, UK: Prentice Hall, 1982).

[17] P. Somol, P. Pudil, and J. Kittler, Fast branch & bound algorithms for optimal feature selection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, 900–912, 2004.

[18] S. Nakariyakul and D. P. Casasent, Adaptive branch and bound algorithm for selecting optimal features, *Pattern Recogn. Lett.*, vol. 28, no. 12, 1415–1427, 2007.

[19] F. Hussein, R. Ward, and N. Kharma, Genetic algorithms for feature selection and weighting, a review and study, *Proc. ICDAR, vol. 00*, IEEE Comp. Soc., 1240–1244, 2001.

[20] H. Zhang and G. Sun, Feature selection using tabu search method, *Pattern Recognition*, vol. 35, 701–711, 2002.

[21] R. Jensen, *Performing Feature Selection with ACO*, ser. Studies in Computational Intelligence. Springer Berlin / Heidelberg, 2006, vol. 34, 45–73.

[22] Y. Yang and J. O. Pedersen, A comparative study on feature selection in text categorization, in *ICML '97: Proc. 14th Int. Conf. on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, 412–420.

[23] F. Sebastiani, Machine learning in automated text categorization, *ACM Computing Surveys*, vol. 34, no. 1, 1–47, March 2002.

[24] E. P. Xing, *Feature Selection in Microarray Analysis*. Springer, 2003, 110–129.

[25] Y. Saeys, I. naki Inza, and P. L. naga, A review of feature selection techniques in bioinformatics, *Bioinformatics*, vol. 23, no. 19, 2507–2517, 2007.

[26] R. Kohavi and G. H. John, Wrappers for feature subset selection, *Artif. Intell.*, vol. 97, no. 1-2, 273–324, 1997.

[27] A. W. Whitney, A direct method of nonparametric measurement selection, *IEEE Trans. Comput.*, vol. 20, no. 9, 1100–1103, 1971.

[28] P. Pudil, J. Novovičová, and J. Kittler, Floating search methods in feature selection, *Pattern Recogn. Lett.*, vol. 15, no. 11, 1119–1125, 1994.

[29] P. Somol, P. Pudil, J. Novovičová, and P. Paclík, Adaptive floating search methods in feature selection, *Pattern Recogn. Lett.*, vol. 20, no. 11-13, 1157–1163, 1999.

[30] S. Nakariyakul and D. P. Casasent, An improvement on floating search algorithms for feature subset selection, *Pattern Recognition*, vol. 42, no. 9, 1932–1940, 2009.

[31] P. Somol and P. Pudil, Oscillating search algorithms for feature selection, in *ICPR 2000*, vol. 02. Los Alamitos, CA, USA: IEEE Computer Society, 2000, 406–409.

[32] J. Novovičová, P. Somol, and P. Pudil, Oscillating feature subset search algorithm for text categorization, in *Structural, Syntactic, and Statistical Pattern Recognition*, vol. LNCS 4109. Berlin / Heidelberg, Germany: Springer-Verlag, 2006, 578–587.

[33] A. Asuncion and D. Newman, UCI machine learning repository, http://www.ics.uci.edu/ ∼mlearn/ mlrepository.html, 2007.

[34] Š. J. Raudys, Feature over-selection, in *Structural, Syntactic, and Statistical Pattern Recognition*, vol. LNCS 4109. Berlin / Heidelberg, Germany: Springer-Verlag, 2006, 622–631.

[35] P. Somol, J. Novovičová, J. Grim, and P. Pudil, Dynamic oscillating search algorithms for feature selection, in *ICPR 2008*. Los Alamitos, CA, USA: IEEE Computer Society, 2008.