We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

4

Efficient Feature Subset Selection and Subset Size Optimization

Petr Somol, Jana Novovičová and Pavel Pudil
Institute of Information Theory and Automation of the Academy of Sciences of the Czech Republic

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 feature selection in statistical pattern recognition will be of primary focus in this chapter. The problem fits in the wider context of dimensionality reduction (Section 2) which can be accomplished either by a linear or nonlinear mapping from the measurement space to a lower dimensional feature space, or by measurement subset selection. This chapter will focus on the latter (Section 3). The main aspects of the problem as well as the choice of the right feature selection tools will be discussed (Sections 3.1 to 3.3). Several optimization techniques will be reviewed, with emphasis put to the framework of sequential selection methods (Section 4). Related topics of recent interest will be also addressed, including the problem of subset size determination (Section 4.7), search acceleration through hybrid algorithms (Section 5), and the problem of feature selection stability and feature over-selection (Section 6).

2. Dimensionality Reduction

The following elementary notation will be followed throughout the chapter. We shall use the term "pattern" to denote the D-dimensional data vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$, the components of which are the measurements of the features characterizing an object. We also refer to $\mathbf{x}$ as the feature vector. Let $\mathrm{Y} = \{f_1, \cdots, f_{|\mathrm{Y}|}\}$ be the set of $D = |\mathrm{Y}|$ features, where $|\cdot|$ denotes the size (cardinality). The features are the variables specified by the investigator. Following the statistical approach to pattern recognition, we assume that a pattern $\mathbf{x}$ is to be classified into one of a finite set Ω of C different classes. A pattern $\mathbf{x}$ belonging to class $\omega \in \Omega$ is viewed as an observation of a random vector drawn randomly according to the class-conditional probability density function and the respective *a priori* probability of class ω.

One of the fundamental problems in pattern recognition is representing patterns in the reduced number of dimensions. In most of practical cases the pattern descriptor space dimensionality is rather high. It follows from the fact that in the design phase it is too difficult or

impossible to evaluate directly the "usefulness" of particular input. Thus it is important to initially include all the "reasonable" descriptors the designer can think of and to reduce the set later on. Obviously, information missing in the original measurement set cannot be later substituted. Dimensionality reduction refers to the task of finding low dimensional representations for high-dimensional data. Dimensionality reduction is an important step in data preprocessing in pattern recognition and machine learning 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.

2.1 DR Categorization According to Nature of the Resulting Features

There are two main distinct ways of viewing DR according to the nature of the resulting features:

- DR by *feature selection* (FS)
- DR by *feature extraction* (FE).

The FS approach does not attempt to generate new features, but to select the "best" ones from the original set of features. (Note: In some research fields, e.g., in image analysis, the term feature selection may be interpreted as feature extraction. It will not be the case in this chapter.) Depending on the outcome of a FS procedure, the result can be either a set of weighting-scoring, a ranking or a subset of features. The 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.

2.2 DR Categorization According to the Aim

DR can be alternatively divided according to the aim of the reduction:

- DR *for optimal data representation*
- DR *for classification*.

The first aims to preserve the topological structure of data in a lower-dimensional space as much as possible, the second one aims to enhance the subset discriminatory power. Although the same tools may be often used for both purposes, caution is needed. An example is PCA, one of the primary tools for representing data in lower-dimensional space, which may easily discard important information if used for DR for classification. In the sequel we shall concentrate on the feature subset selection problem only, with classification being the primary aim. For a broader overview of the subject see, e.g., Duda et al. (2000), McLachlan (2004), Ripley (2005), Theodoridis et al. (2006), Webb (2002).

3. Feature Subset Selection

Given a set Y of $|Y|$ 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 $J(X)$ be a criterion function that evaluates feature subset $X \in \mathcal{X}_d$. 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

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

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.
One particular property of feature selection criterion, the monotonicity property, is required specifically in certain optimal FS methods. Assuming we have two subsets S_1 and S_2 of feature set Y and a criterion J that evaluates each subset S_i. The *monotonicity condition* requires the following:

$$\mathrm{S}_1 \subset \mathrm{S}_2 \Rightarrow J(\mathrm{S}_1) \leq J(\mathrm{S}_2). \tag{2}$$

That is, evaluating the feature selection criterion on a subset of features of a given set yields a smaller value of the feature selection criterion.

3.1 FS Categorization With Respect to Optimality

Feature subset selection methods can be split into basic families:

- *Optimal methods*: These include, e.g., *exhaustive search* methods which are feasible for only small size problems and accelerated methods, mostly built upon the Branch & Bound principle (Somol et al. (2004)). All optimal methods can be expected considerably slow for problems of high dimensionality.
- *Sub-optimal methods*: They essentially trade the optimality of the selected subset for computational efficiency. They include, e.g., Best Individual Features, Random (Las Vegas) methods, Sequential Forward and Backward Selection, Plus-*l*-Take Away-*r*, their generalized versions, genetic algorithms, and particularly the Floating and Oscillating algorithms (Devijver et al. (1982), Pudil et al. (1994), Somol et al. (2000), Somol et al. (2008b)).

Although the exhaustive search guarantees the optimality of a solution, in many realistic problems it is computationally prohibitive. The well known Branch and Bound (B&B) algorithm guarantees to select an optimal feature subset of size d without involving explicit evaluation of all the possible combinations of d measurements. However, the algorithm is applicable only under the assumption that the feature selection criterion used satisfies the monotonicity condition (2). This assumption precludes the use of classifier error rate as the criterion (cf. wrappers, Kohavi et al. (1997b)). This is an important drawback as the error rate can be considered superior to other criteria, Siedlecki et al. (1993), Kohavi et al. (1997b), Tsamardinos et al. (2003). Moreover, all optimal algorithms become computationally prohibitive for problems of high dimensionality. In practice, therefore, one has to rely on computationally feasible procedures which perform the search quickly but may yield sub-optimal results. A comprehensive list of sub-optimal procedures can be found, e.g., in books Devijver et al. (1982), Fukunaga (1990), Webb (2002), Theodoridis et al. (2006). A comparative taxonomy can be found, e.g., in Blum et al. (1997), Ferri et al. (1994), Guyon et al. (2003), Jain et al. (1997), Jain et al. (2000), Yusta (2009), Kudo et al. (2000), Liu et al. (2005), Salappa et al. (2007), Vafaie et al. (1994) or Yang et al. (1998). Our own research and experience with FS has led us to the conclusion that *there exists no unique generally applicable approach* to the problem. Some approaches are more suitable

under certain conditions, others are more appropriate under other conditions, depending on our *knowledge of the problem*. Hence continuing effort is invested in developing new methods to cover the majority of situations which can be encountered in practice.

3.2 FS Categorization With Respect to Selection Criteria

Based on the *selection criterion* choice, feature selection methods may roughly be divided into:

- *Filter methods* (Yu et al. (2003), Dash et al. (2002)) 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 methods* (Kohavi et al. (1997a)) 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.
- *Embedded methods* (Guyon et al. (2003), but also Kononenko (1994) or Pudil et al. (1995), Novovičová et al. (1996)) integrate the feature selection process into the model estimation process. Devising model and selecting features is thus one inseparable learning process, that may be looked upon as a special form of wrappers. Embedded methods thus offer performance competitive to wrappers, enable faster learning process, but produce results tightly coupled with particular model.
- *Hybrid approach* (Das (2001), Sebban et al. (2002), Somol et al. (2006)) combines the advantages of more than one of the listed approaches. Hybrid algorithms have recently been proposed to deal with high dimensional data. 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.

3.3 FS Categorization With Respect to Problem Knowledge

From another point of view there are perhaps two basic classes of situations with respect to *a priori* knowledge of the underlying probability structures:

- *Some a priori knowledge is available*: It is at least known that probability density functions are unimodal. In these cases, one of probabilistic distance measures (Mahalanobis, Bhattacharyya, etc., see Devijver et al. (1982)) may be appropriate as the evaluation criterion. For this type of situations we recommend either the recent prediction-based B&B algorithms for optimal search Somol et al. (2004), or sub-optimal search methods in appropriate filter or wrapper setting (Sect. 4).
- *No a priori knowledge is available*: We cannot even assume that probability density functions are unimodal. For these situations either a wrapper-based solution using sub-optimal search methods (Sect. 4) can be found suitable, or, provided the size of training data is sufficient, it is possible to apply one of the embedded mixture-based methods that are based on approximating unknown class-conditional probability density functions by finite mixtures of a special type (Pudil et al. (1995), Novovičová et al. (1996)).

4. Sub-optimal Search Methods

Provided a suitable FS criterion function (cf. Devijver et al. (1982)) is available, the only tool needed is the search algorithm that generates a sequence of subsets to be tested. Despite the advances in optimal search (Somol et al. (2004), Nakariyakul et al. (2007)), 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 (Hussein et al. (2001)), tabu search (Zhang et al. (2002)), or ant colony (Jensen (2006)). 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 4.2). 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 4.7 a *d*-optimizing procedure will be described, that optimizes both the feature subset size and its contents at once.

4.1 Best Individual Features

The Best Individual Features (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 (Yang et al. (1997), Sebastiani (2002)), genetics (Xing (2003), Saeys et al. (2007)), etc. BIF may be preferable in other types of problems to overcome FS stability problems (see Sect. 6.1). 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.

4.2 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*

$$f^+ = \arg \max_{f \in \mathrm{Y} \setminus \mathrm{X}_d} J^+(\mathrm{X}_d, f) \,, \tag{3}$$

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*

$$ADD(\mathrm{X}_d) \equiv \mathrm{X}_d \cup \{f^+\} = \mathrm{X}_{d+1}, \quad \mathrm{X}_d, \mathrm{X}_{d+1} \subset \mathrm{Y}. \tag{4}$$

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

$$f^- = \arg \max_{f \in \mathrm{X}_d} J^-(\mathrm{X}_d, f) \,, \tag{5}$$

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 $REMOVE(\mathrm{X}_d)$ is an operation of removing feature f^- from the current set X_d to obtain set X_{d-1} if

$$REMOVE(\mathrm{X}_d) \equiv \mathrm{X}_d \setminus \{f^-\} = \mathrm{X}_{d-1}, \quad \mathrm{X}_d, \mathrm{X}_{d-1} \subset \mathrm{Y}. \tag{6}$$

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

$$\begin{aligned} \mathrm{X}_{d+2} &= ADD(\mathrm{X}_{d+1}) = ADD(ADD(\mathrm{X}_d)) = ADD^2(\mathrm{X}_d)\,, \\ \mathrm{X}_{d-2} &= REMOVE(REMOVE(\mathrm{X}_d)) = REMOVE^2(\mathrm{X}_d)\,, \end{aligned} \tag{7}$$

and more generally

$$\mathrm{X}_{d+\delta} = ADD^\delta(\mathrm{X}_d), \quad \mathrm{X}_{d-\delta} = REMOVE^\delta(\mathrm{X}_d)\,. \tag{8}$$

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

$$J^+(\mathrm{X}_d, f) = J(\mathrm{X}_d \cup \{f\}), \qquad J^-(\mathrm{X}_d, f) = J(\mathrm{X}_d \setminus \{f\})\,, \tag{9}$$

where $J(\cdot)$ is either a filter- or wrapper-based criterion function (Kohavi et al. (1997b)) to be evaluated on the subspace defined by the tested feature subset.

4.3 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 remain (*top-down* approach). The simplest (among recommendable choices) yet widely used *sequential forward (or backward) selection* methods, SFS and SBS (Whitney (1971), Devijver et al. (1982)), 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. $\mathrm{X}_d = ADD^d(\emptyset)$.

SBS (*Sequential Backward Selection*) yielding a subset of d features:

1. $\mathrm{X}_d = REMOVE^{|\mathrm{Y}|-d}(\mathrm{Y})$.

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 (Devijver et al. (1982)) which involve successive augmentation and depletion process. The same idea in a principally extended and refined form constitutes the basis of Floating Search.

4.4 Sequential Floating Search

The Sequential Forward Floating Selection (SFFS) (Pudil et al. (1994)) 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.

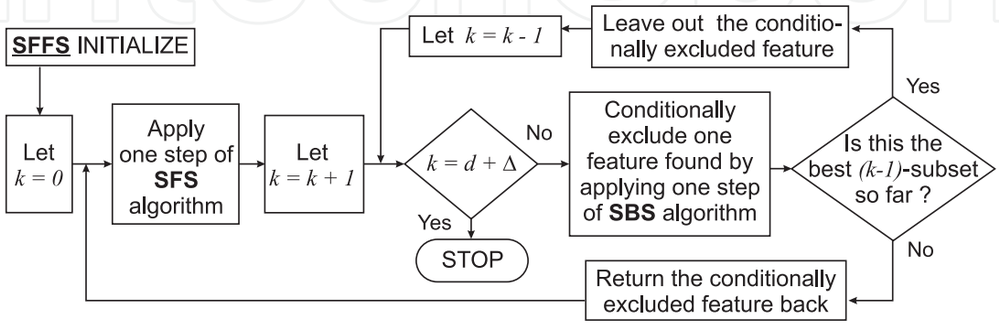


Fig. 1. Sequential Forward Floating Selection Algorithm

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

1. Start with $X_0 = \emptyset$, $k = 0$.
2. $X_{k+1} = ADD(X_k)$, $k = k + 1$.
3. Repeat $X_{k-1} = REMOVE(X_k)$, $k = k - 1$ as long as it improves solutions already known for the lower k.
4. If $k < d + \Delta$ go to 2.

A detailed formal description of this now classical procedure can be found in Pudil et al. (1994). Nevertheless, the idea behind it is simple enough and can be illustrated sufficiently in Fig. 1. (Condition $k = d + \Delta$ terminates the algorithm after the target subset of d features has been found and possibly refined by means of backtracking from dimensionalities greater than d.) 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.

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

1. Start with $X_0 = Y$, $k = |Y|$.
2. $X_{k-1} = REMOVE(X_k)$, $k = k - 1$.

3. Repeat $X_{k+1} = ADD(X_k), k = k+1$ as long as it improves solutions already known for the higher k.
4. If $k > d - \Delta$ go to 2.

4.4.1 Further Developments of the Floating Search Idea

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 has been proposed in Somol et al. (1999). The ASFFS and ASBFS algorithms are able to outperform the classical SFFS and SBFS 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 ASFFS/ASBFS is usually inferior to newer algorithms, which we focus on in the following. An improved version of Floating Search has been published recently in Nakariyakul et al. (2009).

4.5 Oscillating Search

The more recent Oscillating Search (OS) (Somol et al. (2000)) 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. 2. 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. For OS flow-chart see Fig. 3.

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 $X_d^{\downarrow} = ADD^o(REMOVE^o(X_d))$.
3. If $X_d^{\downarrow}$ better than X_d, let $X_d = X_d^{\downarrow}$, let $o = 1$ and go to 2.
4. Let $X_d^{\uparrow} = REMOVE^o(ADD^o(X_d))$.
5. If $X_d^{\uparrow}$ better than X_d, let $X_d = X_d^{\uparrow}$, let $o = 1$ and go to 2.
6. If $o < \Delta$ let $o = o + 1$ and go to 2.

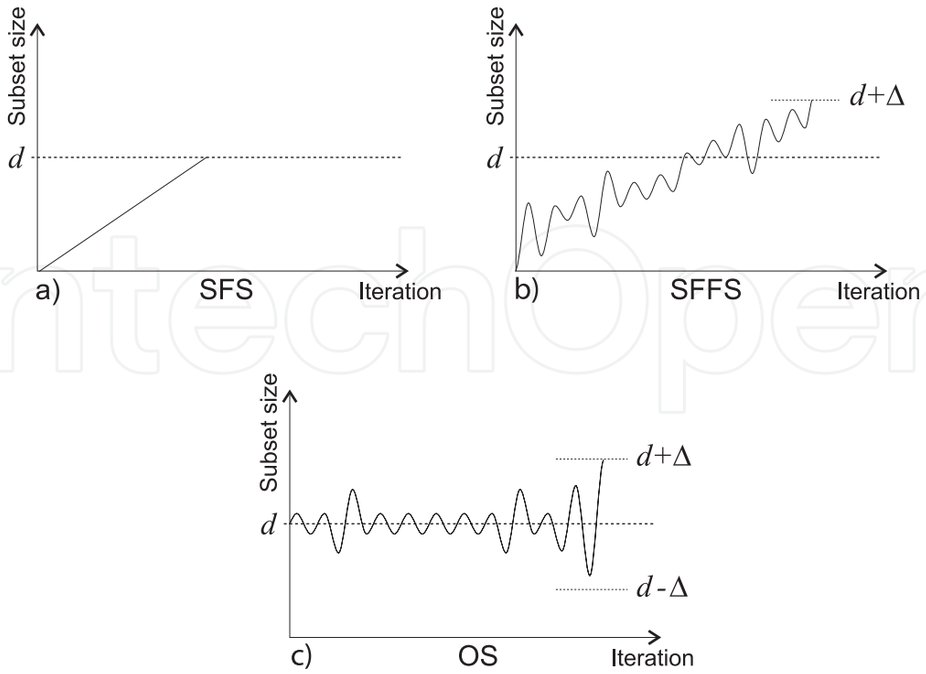


Fig. 2. Graphs demonstrate the course of d-parametrized search algorithms: a) Sequential Forward Selection, b) Sequential Forward Floating Selection, c) Oscillating Search.

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 / REMOVE). 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 (Novovičová et al. (2006)) 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.

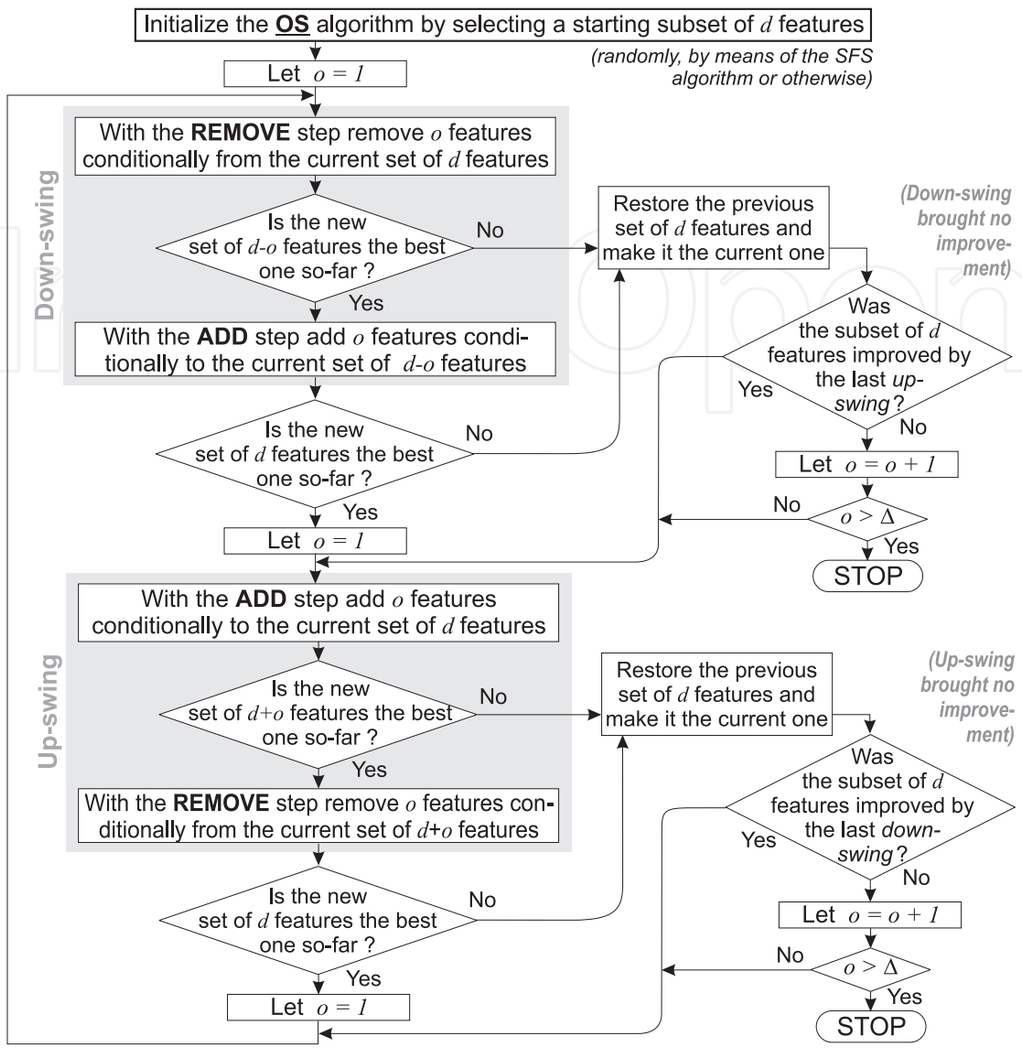


Fig. 3. Simplified Oscillating Search algorithm flowchart.

- 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.

4.6 Experimental Comparison of d-Parametrized Methods

The d-parametrized sub-optimal FS methods as discussed in preceding sections 4.1 to 4.5 have been listed in the order of their speed-vs-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 (Kohavi et al. (1997a)) 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 (Asuncion et al. (2007)). 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× from different random initial subsets of given size, with $\Delta = 15$. This set-up is highly time consuming but enables to avoid many local extremes that would not be avoided by other algorithms.
Figure 4 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.

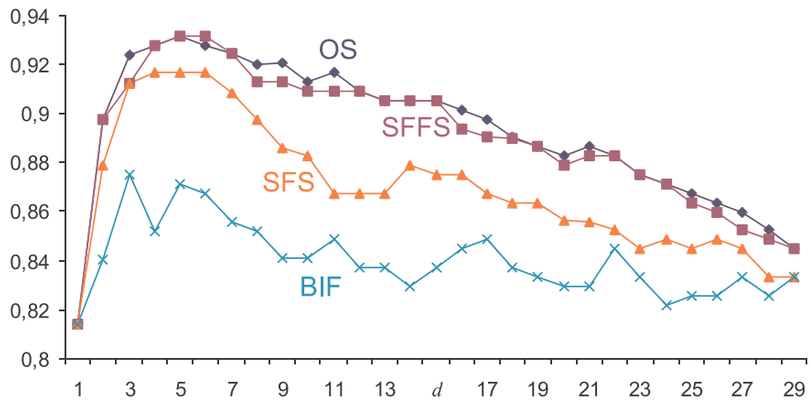


Fig. 4. Sub-optimal FS methods' optimization performance on 3-NN wrapper

Figure 5 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 independent performance 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 (Raudys (2006)).
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× slower than SFS. OS in the slow test setting runs roughly 10 to 100× slower than SFFS.

4.7 Dynamic Oscillating Search – Optimizing Subset Size

The idea of Oscillating Search (Sect. 4.5) has been further extended in form of the Dynamic Oscillating Search (DOS) (Somol et al. (2008b)). 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.

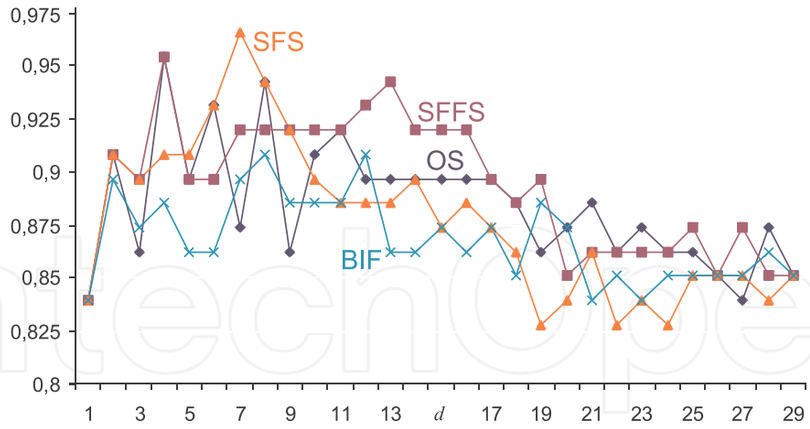


Fig. 5. Sub-optimal FS methods' performance verified using 3-NN on independent data

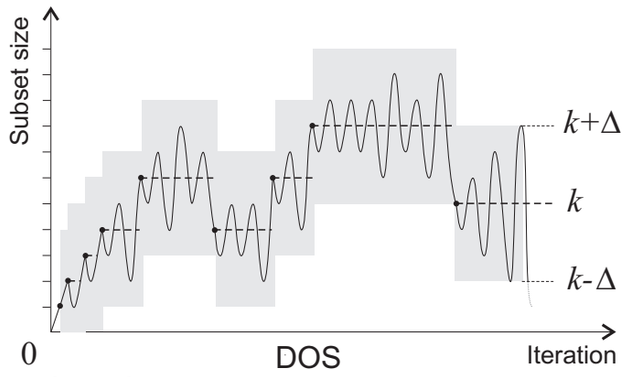


Fig. 6. The DOS course of search

The course of Dynamic Oscillating Search is illustrated in Fig. 6. See Fig. 2 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. 4.5: 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 $X_k = ADD(ADD(\emptyset))$, k=2. Set cycle depth to $\delta = 1$.
2. Compute $ADD^{\delta}(REMOVE^{\delta}(X_k))$; 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 $REMOVE^{\delta}(ADD^{\delta}(X_k))$; 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 2.
4. If $\delta < \Delta$ let $\delta = \delta + 1$ and go to 2.

A simplified DOS flowchart is given in Fig. 7. In the course of search the DOS generates a sequence of solutions with ascending criterion values and, provided the criterion value does not change, decreasing subset size. The search time vs. closeness-to-optimum trade-off can thus

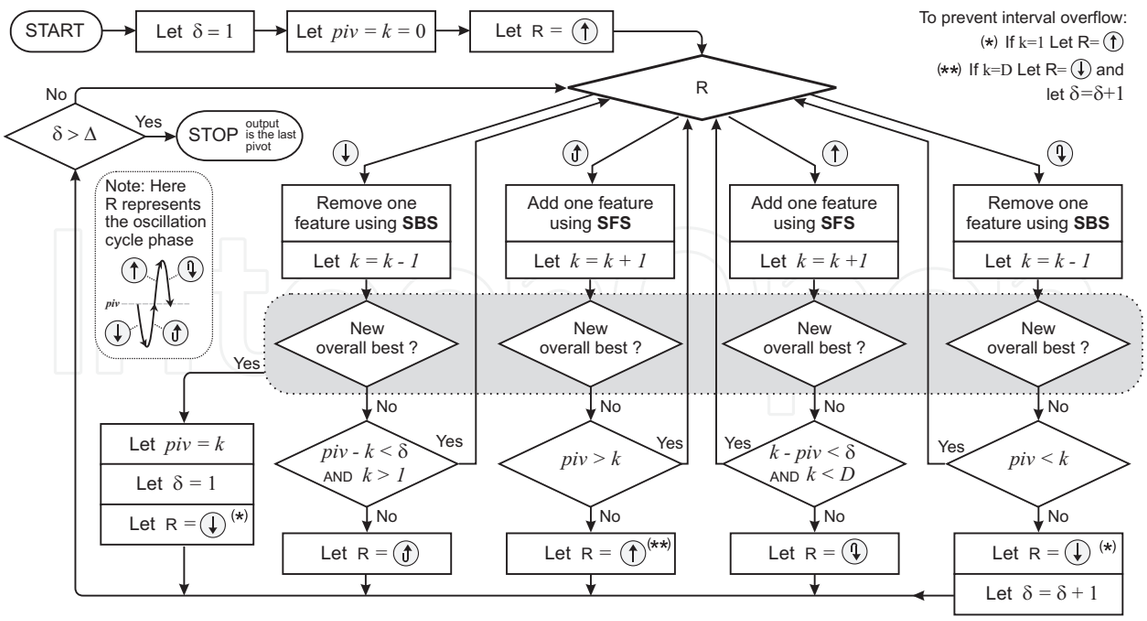


Fig. 7. Simplified diagram of the DOS algorithm.

be handled by means of pre-mature search interruption. The number of criterion evaluations is in the $O(n^3)$ 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.

4.7.1 DOS Experiments

We compare the DOS algorithm with the previously discussed methods SFS, SFFS and OS, here used in *d*-optimizing manner: each method is run for each possible subset size to eventually select the subset size that yields the highest criterion value. To mark the difference from standard *d*-parametrized course of search we denote these methods SFS*, SFFS*and OS*.
We used the accuracy of various classifiers as FS criterion function: Bayesian classifier assuming Gauss distribution, 3-Nearest Neighbor and SVM with RBF kernel (Chang et al. (2001)). We tested the methods on *wdbc* data (30 dim., 2 classes: 357 benign and 212 malignant samples) from UCI Repository (Asuncion et al. (2007)). The experiments have been accomplished using 2-tier cross-validation. The outer 10-fold cross-validation loop serves to produce different test-train data splits, the inner 10-fold cross-validation loop further splits the train data part for classifier training and validation as part of the FS process. The results of our experiments are collected in Table 1. (Further set of related experiments can be found in Table 3.)
Each table contains three sections gathering results for one type of classifier (criterion function). The main information of interest is in the column I-CV, showing the maximum criterion value (classification accuracy) yielded by each FS method in the *inner* cross-validation loop, and O-CV, showing the averaged respective classification accuracy on independent test data.
The following properties of the Dynamic Oscillating Search can be observed: (i) it is able to outperform other tested methods in the sense of criterion maximization ability (I-CV), (ii) it tends to produce the smallest feature subsets, (iii) its impact on classifier performance on unknown data varies depending on data and classifier used – in some cases it yields the best results, however this behavior is inconsistent.

Crit.	Meth.	I-CV	O-CV	Size	Time(h)
Gauss	SFS⋆	0.962	0.933	10.8	00:00
	SFFS⋆	0.972	0.942	10.6	00:03
	OS⋆	0.970	0.940	9.9	00:06
	DOS	0.973	0.951	10.7	00:06
	full set		0.945	30	
3-NN	SFS⋆	0.981	0.967	15.3	00:01
scaled	SFFS⋆	0.983	0.970	13.7	00:09
	OS⋆	0.982	0.965	14.2	00:22
	DOS	0.984	0.965	12.4	00:31
	full set		0.972	30	
SVM	SFS⋆	0.979	0.970	18.5	00:05
C=16, γ=0.031	SFFS⋆	0.982	0.968	16.2	00:23
	OS⋆	0.981	0.974	16.7	00:58
	DOS	0.983	0.968	12.8	01:38
	full set		0.972	30	

Table 1. Performance of FS wrapper methods evaluated on *wdbc* data, 30-dim., 2-class.

5. Hybrid Algorithms – Accelerating the Search

Filter methods 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 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.

Hybrid FS algorithms can be defined easily to utilize the advantages of both filters and wrappers (Somol et al. (2006)). In the course of search, in each algorithm step filter is used to reduce the number of candidates to be evaluated in wrapper. The scheme can be applied in any sequential FS algorithms 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*

$$Z^+ = \{f_i : f_i \in Y \setminus X_d; i = 1, \ldots, \max\{1, \lceil \lambda \cdot |Y \setminus X_d| \rceil\}\} \tag{10}$$

such that

$$\forall f, g \in Y \setminus X_d, f \in Z^+, g \notin Z^+ \qquad J_F^+(X_d, f) \geq J_F^+(X_d, g)\ , \tag{11}$$

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

$$f^+ = \arg\max_{f \in Z^+} J_W^+(X_d, f)\ , \tag{12}$$

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

$$ADD_H(X_d) \equiv X_d \cup \{f^+\} = X_{d+1}, \quad X_d, X_{d+1} \subset Y. \tag{13}$$

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

$$Z^- = \{f_i : f_i \in X_d; i = 1, \ldots, \max\{1, \lceil \lambda \cdot |X_d| \rceil\}\} \tag{14}$$

such that

$$\forall f, g \in X_d, f \in Z^-, g \notin Z^- \qquad J_F^-(X_d, f) \geq J_F^-(X_d, g)\ , \tag{15}$$

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

$$f^- = \arg\max_{f \in Z^-} J_W^-(X_d, f), \tag{16}$$

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

$$REMOVE_H(X_d) \equiv X_d \setminus \{f^-\} = X_{d-1}, \quad X_d, X_{d-1} \subset Y. \tag{17}$$

The effect of hybridization is illustrated on the example in Table 2. We tested the hybridized DOS method on *waveform* data (40 dim., 2 classes: 1692 and 1653 samples) from UCI Repository (Asuncion et al. (2007)). In the hybrid setting we used Bhattacharyya distance (Devijver et al. (1982)) as the fast filter criterion and 3-Nearest Neighbor as the slow wrapper criterion. The reported wrapper accuracy represents the maximum criterion value found for the selected feature subset. The reported independent accuracy has been obtained on independent test data using 3-NN. Note that despite considerable reduction of search time for lower λ the obtained feature subset yields comparable accuracy of the wrapper classifier.

Hybridization coeff. λ	0.01	0.25	0.5	0.75	1
Wrapper accuracy	0.907136	0.913116	0.921089	0.921485	0.921485
Independent accuracy	0.916268	0.911483	0.911483	0.910287	0.910287
Determined subset size	11	10	15	17	17
Time	1:12	8:06	20:42	35:21	48:24

Table 2. Performance of the hybridized Dynamic Oscillating Search wrapper FS method evaluated on *waveform* data, 40-dim., 2-class.

6. The Problem of Feature Selection Overfitting and Stability

In older literature the prevailing approach to FS method performance assessment was to evaluate the ability to find the optimum, or to get as close to it as possible, with respect to some criterion function defined to distinguish classes in classification tasks or to fit data in approximation tasks. Recently, emphasis is put on assessing the impact of FS on generalization performance, i.e., the ability of the devised decision rule to perform well on independent data. It has been shown that similarly to classifier over-training the effect of feature over-selection can hinder the performance of pattern recognition system (Raudys (2006), Raudys (2006)); especially with small-sample or high-dimensional problems. Compare Figures 4 and 5 to see an example of the effect.

It has been also pointed out that independent test data performance should not be neglected when comparing FS methods (Reunanen (2003)). The task of FS methods' comparison seems to be understood ambiguously as well. It is very different whether we compare concrete method properties or the final classifier performance determined by use of particular methods under particular settings. Certainly, final classifier performance is the ultimate quality measure. However, misleading conclusions about FS may be easily drawn when evaluating nothing else, as classifier performance depends on many more different aspects then just the actual FS method used.

There seems to be a general agreement in literature that wrapper-based FS enables creation of more accurate classifiers than filter-based FS. This claim is nevertheless to be taken with caution, while using actual classifier accuracy as FS criterion in wrapper-based FS may lead to the very negative effects mentioned above (overtraining). At the same time the weaker relation of filter-based FS criterion functions to particular classifier accuracy may help better generalization. But these effects can be hardly judged before the building of classification system has actually been accomplished. The problem of classifier performance estimation is by no means simple. Many estimation strategies are available, suitability of which is problem dependent (re-substitution, data split, hold-out, cross-validation, leave-one-out, etc.). For a detailed study on classifier training related problems and work-around methods, e.g., stabilizing weak classifiers, see Skurichina (2001).

6.1 The Problem of Feature Selection Stability

It is common that classifier performance is considered the ultimate quality measure, even when assessing the FS process. However, misleading conclusions may be easily drawn when ignoring stability issues. Unstable FS performance may seriously deteriorate the properties of the final classifier by selecting the wrong features. Following Kalousis et al. (2007) we define the *stability* of the FS algorithm as the robustness of the feature preferences it produces to differences in training sets drawn from the same generating distribution. FS algorithms express the feature preferences in the form of a selected feature subset $S \subseteq Y$. Stability quantifies how

different training sets drawn from the same generating distribution affect the feature preferences. Recent works in the area of FS methods' stability mainly focus on various stability indices, introducing measures based on Hamming distance, Dunne et al. (2002), correlation coefficients and Tanimoto distance, Kalousis et al. (2007), consistency index, Kuncheva (2007) and Shannon entropy, Křížek et al. (2007). Stability of FS procedures depends on the sample size, the criteria utilized to perform FS, and the complexity of FS procedure, Raudys (2006).
In the following we focus on several new measures allowing to assess the FS stability of both the *d*-parametrized and *d*-optimizing FS methods (Somol et al. (2008a)).

6.1.1 Considered Measures of Feature Selection Stability

Let $\mathcal{S} = \{\mathrm{S}_1, \ldots, \mathrm{S}_n\}$ be a system of n feature subsets $\mathrm{S}_j = \{f_{k_i} |\ i = 1, \ldots, d_j, f_{k_i} \in \mathrm{Y}, d_j \in \{1, \ldots, |\mathrm{Y}|\}\}, j = 1, \ldots, n,\ n > 1, n \in \mathbb{N}$, obtained from n runs of the evaluated FS algorithm on different samplings of a given data set. Let X be the subset of Y representing all features that appear anywhere in $\mathcal{S}$:

$$\mathrm{X} = \{f | f \in \mathrm{Y}, F_f > 0\} = \bigcup_{i=1}^{n} \mathrm{S}_i,\ \ \mathrm{X} \neq \emptyset, \tag{18}$$

where F_f is the number of occurrences (frequency) of feature $f \in \mathrm{Y}$ in system $\mathcal{S}$. Let N denote the total number of occurrences of any feature in system $\mathcal{S}$, i.e.,

$$N = \sum_{g \in \mathrm{X}} F_g = \sum_{i=1}^{n} |\mathrm{S}_i|,\ \ N \in \mathbb{N},\ \ N \geq n\ . \tag{19}$$

Definition 5. *The* weighted consistency $CW(\mathcal{S})$ *of the system* $\mathcal{S}$ *is defined as*

$$CW(\mathcal{S}) = \sum_{f \in \mathrm{X}} w_f \frac{F_f - F_{min}}{F_{max} - F_{min}}\ , \tag{20}$$

where $w_f = \frac{F_f}{N}, 0 < w_f \leq 1, \sum_{f \in \mathrm{X}} w_f = 1$.

Because $F_f = 0$ for all $f \in \mathrm{Y} \setminus \mathrm{X}$, the *weighted consistency* $CW(\mathcal{S})$ can be equally expressed using notation (19):

$$CW(\mathcal{S}) = \sum_{f \in \mathrm{X}} \frac{F_f}{N} \cdot \frac{F_f - F_{min}}{F_{max} - F_{min}} = \sum_{f \in \mathrm{Y}} \frac{F_f}{N} \cdot \frac{F_f - 1}{n - 1}\ . \tag{21}$$

It is obvious that $CW(\mathcal{S}) = 0$ if and only if (iff) $N = |\mathrm{X}|$, i.e., iff $F_f = 1$ for all $f \in \mathrm{X}$. This is unrealistic in most of real cases. Whenever $n > |\mathrm{X}|$, some feature must appear in more than one subset and consequently $CW(\mathcal{S}) > 0$. Similarly, $CW(\mathcal{S}) = 1$ iff $N = n|\mathrm{X}|$, otherwise all subsets can not be identical.
Clearly, for any N, n representing some system of subsets $\mathcal{S}$ and for given Y there exists a system $\mathcal{S}_{min}$ with such configuration of features in its subsets that yields the minimal possible $CW(\cdot)$ value, to be denoted $CW_{min}(N, n, \mathrm{Y})$, being possibly greater than 0. Similarly, a system $\mathcal{S}_{max}$ exists that yields the maximal possible $CW(\cdot)$ value, to be denoted $CW_{max}(N, n)$, being possibly lower than 1.
It can be easily seen that $CW_{min}(\cdot)$ gets high when the sizes of feature subsets in system approach the total number of features $|\mathrm{Y}|$, because in such system the subsets get necessarily more similar to each other. Consequently, using measure (20) for comparison of the stability of various FS methods may lead to misleading results if the methods tend to yield systems

of differently sized subsets. We will refer to this problem as to "the problem of subset-size bias". Note that most of available stability measures are affected by the same problem. For this reason we introduce another measure, to be called the *relative weighted consistency*, which suppresses the influence of the sizes of subsets in system on the final value.

Definition 6. *The* relative weighted consistency $CW_{rel}(\mathcal{S},\mathrm{Y})$ *of system* $\mathcal{S}$ *characterized by* N,n *and for given* Y *is defined as*

$$CW_{rel}(\mathcal{S},\mathrm{Y}) = \frac{CW(\mathcal{S}) - CW_{min}(N,n,\mathrm{Y})}{CW_{max}(N,n) - CW_{min}(N,n,\mathrm{Y})}\ , \tag{22}$$

where $CW_{rel}(\mathcal{S},\mathrm{Y}) = CW(\mathcal{S})$ for $CW_{max}(N,n) = CW_{min}(N,n,\mathrm{Y})$.
Denoting $D = N \bmod |\mathrm{Y}|$ and $H = N \bmod n$ for simplicity, it has been shown in Somol et al. (2008a) that

$$CW_{min}(N,n,\mathrm{Y}) = \frac{N^2 - |\mathrm{Y}|(N-D) - D^2}{|\mathrm{Y}|N(n-1)} \tag{23}$$

and

$$CW_{max}(N,n) = \frac{H^2 + N(n-1) - Hn}{N(n-1)}\ . \tag{24}$$

The relative weighted consistency then becomes:

$$CW_{rel}(\mathcal{S},\mathrm{Y}) = \frac{|\mathrm{Y}|\left(N - D + \sum_{f\in\mathrm{Y}} F_f(F_f-1)\right) - N^2 + D^2}{|\mathrm{Y}|\left(H^2 + n(N-H) - D\right) - N^2 + D^2}\ . \tag{25}$$

The weighted consistency bounds $CW_{max}(N,n)$ and $CW_{min}(N,n,\mathrm{Y})$ are illustrated in Fig. 8.

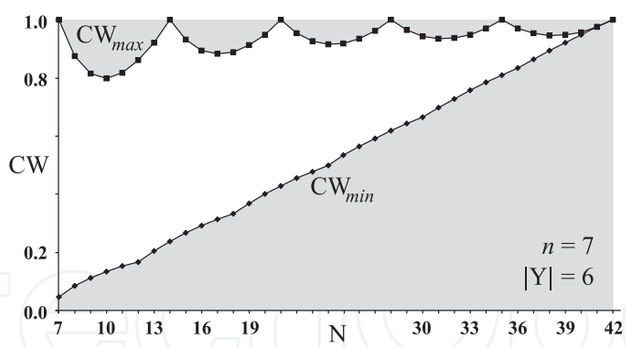


Fig. 8. Illustration of *CW* measure bounds

Note that CW_{rel} may be sensitive to small system changes if N approaches maximum (for given $|\mathrm{Y}|$ and n).
It can be seen that for any N,n representing some system of subsets $\mathcal{S}$ and for given Y it is true that $0 \le CW_{rel}(\mathcal{S},\mathrm{Y}) \le 1$ and for the corresponding systems $\mathcal{S}_{min}$ and $\mathcal{S}_{max}$ it is true that $CW_{rel}(\mathcal{S}_{min}) = 0$ and $CW_{rel}(\mathcal{S}_{max}) = 1$.
The measure (22) does not exhibit the unwanted behavior of yielding higher values for systems with subset sizes closer to $|\mathrm{Y}|$, i.e., is independent of the size of feature subsets selected by the examined FS methods under fixed Y. We can say that this measure characterizes for given $\mathcal{S},\mathrm{Y}$ the relative degree of randomness of the system of feature subsets on the scale between the maximum and minimum values of the weighted consistency (20).

Next, following the idea of Kalousis et al. (2007) we define a conceptually different measure. It is derived from the *Tanimoto index (coefficient)* defined as the size of the intersection divided by the size of union of the subsets S_i and S_j, Duda et al. (2000):

$$S_K(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \ . \tag{26}$$

Definition 7. *The* Average Tanimoto Index *of system $\mathcal{S}$ is defined as follows:*

$$ATI(\mathcal{S}) = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} S_K(S_i, S_j) \ . \tag{27}$$

$ATI(\mathcal{S})$ is the average similarity measure over all pairs of feature subsets in $\mathcal{S}$. It takes values from $[0, 1]$ with 0 indicating empty intersection between all pairs of subsets S_i, S_j and 1 indicating that all subsets of the system $\mathcal{S}$ are identical.

Wrap.	FS Meth.	Classif. rate Mean	Classif. rate *S.Dv.*	Subset size Mean	Subset size *S.Dv.*	CW	CW rel	ATI	FS time h:m:s
Gauss.	rand	.908	*.059*	14.90	*8.39*	.500	.008	.296	00:00:14
	BIF⋆	.948	*.004*	27.15	*4.09*	.927	.244	.862	00:04:57
	SFS⋆	.963	*.003*	11.95	*5.30*	.506	.181	.332	01:02:04
	SFFS⋆	.969	*.003*	12.17	*4.66*	.556	.259	.387	09:13:03
	DOS	.973	*.002*	8.85	*2.36*	.584	.419	.429	12:49:59
3NN	rand	.935	*.061*	14.9	*8.30*	.501	.009	.297	00:00:45
	BIF⋆	.970	*.002*	24.78	*3.70*	.912	.513	.840	00:38:39
	SFS⋆	.976	*.002*	15.45	*5.74*	.584	.148	.401	07:27:39
	SFFS⋆	.979	*.002*	17.96	*5.67*	.658	.149	.481	33:53:55
	DOS	.980	*.001*	13.27	*4.25*	.565	.227	.393	116:47:
SVM	rand	.942	*.059*	14.94	*8.58*	.502	.008	.295	00:00:50
	BIF⋆	.974	*.003*	21.67	*2.71*	.929	.774	.875	01:01:48
	SFS⋆	.982	*.002*	9.32	*4.12*	.433	.185	.283	07:13:02
	SFFS⋆	.983	*.002*	10.82	*4.58*	.472	.179	.310	30:28:02
	DOS	.985	*.001*	8.70	*3.42*	.442	.222	.295	74:28:51

Table 3. Stability of wrapper FS methods evaluated on *wdbc* data, 30-dim., 2-class.

6.1.2 Experiments With Stability Measures

To illustrate the discussed stability measures we have conducted several experiments on *wdbc* data (30 dim., 2 classes: 357 benign and 212 malignant samples) from UCI Repository (Asuncion et al. (2007)). The results are collected in Table 3. We focused on comparing the stability of principally different FS methods discussed in this chapter: BIF, SFS and SFFS and DOS in *d*-optimizing setting; *d*-parametrized methods are run for each possible subset size to eventually select the subset size that yields the highest criterion value. To mark the difference from standard *d*-parametrized course of search we denote these methods BIF⋆, SFS⋆and SFFS⋆. We used the classification accuracy of three conceptually different classifiers as FS criteria: Gaussian classifier, 3-Nearest Neighbor (majority voting) and SVM with RBF kernel (Chang et al. (2001)). In each setup FS was repeated 1000× on randomly sampled 80% of the data (class size ratios preserved). In each FS run the criterion was evaluated using 10-fold cross-validation, with 2/3 of available data randomly sampled for training and the remaining 1/3 used for testing.

The results are collected in Table 3. All measures, CW, CW_{rel} and ATI indicate BIF*as the most stable FS method, what confirms the conclusions in Kuncheva (2007). Note that CW_{rel} is the only measure to correctly detect random feature selection (values close to 0). Note that apart from BIF*, with 3-NN and SVM the most stable FS method appears to be SFFS*, with Gaussian classifier it is DOS. Very low CW_{rel} values may indicate some pitfall in the FS process - either there are no clearly preferable features in the set, or the methods overfit, etc. Note that low stability measure values are often accompanied by higher deviations in subset size.

7. Summary

The current state of art in feature selection based dimensionality reduction for decision problems of classification type has been overviewed. A number of recent feature subset search strategies have been reviewed and compared. Following the analysis of their respective advantages and shortcomings, the conditions under which certain strategies are more pertinent than others have been suggested.

Concerning our current experience, we can give the following recommendations. Floating Search can be considered the first tool to try for many FS tasks. 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 repeated with different random initial sets the Oscillating Search shows outstanding potential to overcome local extremes in favor of global optimum. Dynamic Oscillating Search adds to Oscillating Search the ability to optimize both the subset size and subset contents at once.

No FS method, however, can be claimed the best for all problems. Moreover, any FS method should be applied cautiously to prevent the negative effects of feature over-selection (overtraining) and to prevent stability issues.

Remark: Source codes can be partly found at `http://ro.utia.cas.cz/dem.html`.

7.1 Does It Make Sense to Develop New FS Methods?

Our answer is undoubtedly yes. Our current experience shows that no clear and unambiguous qualitative hierarchy can be established within the existing framework of methods, i.e., although some methods perform better then others more often, this is not the case always and any method can show to be the best tool for some particular problem. Adding to this pool of methods may thus bring improvement, although it is more and more difficult to come up with new ideas that have not been utilized before. Regarding the performance of search algorithms as such, developing methods that yield results closer to optimum with respect to any given criterion may bring considerably more advantage in future, when better criteria may have been found to better express the relation between feature subsets and classifier generalization ability.

8. Acknowledgements

The work has been supported by projects AV0Z1075050506 of the GAAV CR, GAČR 102/08/0593, 102/07/1594 and CR MŠMT grants 2C06019 ZIMOLEZ and 1M0572 DAR.

9. References

Asuncion, A. & Newman, D. (2007). UCI machine learning repository, http://www.ics.uci.edu/ ∼mlearn/ mlrepository.html.

Blum, A. & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2), 245–271.

Chang, C.-C. & Lin, C.-J. (2001). *LIBSVM: a library for SVM*. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. In *Proc. of the 18th International Conference on Machine Learning* pp. 74–81.

Dash, M.; Choi, K.; P., S., & Liu, H. (2002). Feature selection for clustering - a filter solution. In *Proceedings of the Second International Conference on Data Mining* pp. 115–122.

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

Duda, R. O.; Hart, P. E., & Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Dunne, K.; Cunningham, P., & Azuaje, F. (2002). *Solutions to Instability Problems with Sequential Wrapper-based Approaches to Feature Selection*. Technical Report TCD-CS-2002-28, Trinity College Dublin, Department of Computer Science.

Ferri, F. J.; Pudil, P.; Hatef, M., & Kittler, J. (1994). Comparative study of techniques for large-scale feature selection. *Machine Intelligence and Pattern Recognition*, 16.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition (2nd ed.)*. San Diego, CA, USA: Academic Press Professional, Inc.

Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3, 1157–1182.

Hussein, F.; Ward, R., & Kharma, N. (2001). Genetic algorithms for feature selection and weighting, a review and study. *icdar*, 00, 1240.

Jain, A. & Zongker, D. (1997). Feature selection: Evaluation, application, and small sample performance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(2), 153–158.

Jain, A. K.; Duin, R. P. W., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1), 4–37.

Jensen, R. (2006). *Performing Feature Selection with ACO*, volume 34 of *Studies in Computational Intelligence*, pp. 45–73. Springer Berlin / Heidelberg.

Kalousis, A.; Prados, J., & Hilario, M. (2007). Stability of feature selection algorithms: A study on high-dimensional spaces. *Knowledge and Information Systems*, 12(1), 95–116.

Kohavi, R. & John, G. (1997a). Wrappers for feature subset selection. *Artificial Intelligence*, 97, 273–324.

Kohavi, R. & John, G. H. (1997b). Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2), 273–324.

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

Křížek, P.; Kittler, J., & Hlaváč, V. (2007). Improving stability of feature selection methods. In *Proc. 12th Int. Conf. on Computer Analysis of Images and Patterns*, volume LNCS 4673 pp. 929–936. Berlin / Heidelberg, Germany: Springer-Verlag.

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

Kuncheva, L. I. (2007). A stability index for feature selection. In *Proc. 25th IASTED International Multi-Conference AIAP'07* pp. 390–395. Anaheim, CA, USA: ACTA Press.

Liu, H. & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4), 491–502.

McLachlan, G. J. (2004). *Discriminant analysis and statistical pattern recognition*. Wiley-IEEE.

Nakariyakul, S. & Casasent, D. P. (2007). Adaptive branch and bound algorithm for selecting optimal features. *Pattern Recogn. Lett.*, 28(12), 1415–1427.

Nakariyakul, S. & Casasent, D. P. (2009). An improvement on floating search algorithms for feature subset selection. *Pattern Recognition*, 42(9), 1932–1940.

Novovičová, J.; Pudil, P., & Kittler, J. (1996). Divergence based feature selection for multimodal class densities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(2), 218–223.

Novovičová, J.; Somol, P., & Pudil, P. (2006). Oscillating feature subset search algorithm for text categorization. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume LNCS 4109 pp. 578–587. Berlin / Heidelberg, Germany: Springer-Verlag.

Pudil, P.; Novovičová, J., & Kittler, J. (1994). Floating search methods in feature selection. *Pattern Recogn. Lett.*, 15(11), 1119–1125.

Pudil, P.; Novovičová, J.; Choakjarernwanit, N., & Kittler, J. (1995). Feature selection based on approximation of class densities by finite mixtures of special type. *Pattern Recognition*, 28, 1389–1398.

Raudys, Š. J. (2006). Feature over-selection. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume LNCS 4109 pp. 622–631. Berlin / Heidelberg, Germany: Springer-Verlag.

Reunanen, J. (2003). Overfitting in making comparisons between variable selection methods. *J. Mach. Learn. Res.*, 3, 1371–1382.

Ripley, B. D., Ed. (2005). *Pattern Recognition and Neural Networks*. Cambridge University Press, 8 edition.

Saeys, Y.; naki Inza, I., & naga, P. L. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19), 2507–2517.

Salappa, A.; Doumpos, M., & Zopounidis, C. (2007). Feature selection algorithms in classification problems: An experimental evaluation. *Optimization Methods and Software*, 22(1), 199–212.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 1–47.

Sebban, M. & Nock, R. (2002). A hybrid filter/wrapper approach of feature selection using information theory. *Pattern Recognition*, 35, 835–846.

Siedlecki, W. & Sklansky, J. (1993). *On automatic feature selection*, pp. 63–87. World Scientific Publishing Co., Inc.: River Edge, NJ, USA.

Skurichina, M. (2001). *Stabilizing Weak Classifiers*. PhD thesis, Pattern Recognition Group, Delft University of Technology, Netherlands.

Somol, P.; Novovičová, J., & Pudil, P. (2006). Flexible-hybrid sequential floating search in statistical feature selection. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume LNCS 4109 pp. 632–639. Berlin / Heidelberg, Germany: Springer-Verlag.

Somol, P. & Novovičová, J. (2008a). Evaluating the stability of feature selectors that optimize feature subset cardinality. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume LNCS 5342 pp. 956–966.

Somol, P.; Novovičová, J.; Grim, J., & Pudil, P. (2008b). Dynamic oscillating search algorithms for feature selection. In *ICPR 2008* Los Alamitos, CA, USA: IEEE Computer Society.

Somol, P. & Pudil, P. (2000). Oscillating search algorithms for feature selection. In *ICPR 2000*, volume 02 pp. 406–409. Los Alamitos, CA, USA: IEEE Computer Society.

Somol, P.; Pudil, P., & Kittler, J. (2004). Fast branch & bound algorithms for optimal feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7), 900–912.

Somol, P.; Pudil, P.; Novovičová, J., & Paclík, P. (1999). Adaptive floating search methods in feature selection. *Pattern Recogn. Lett.*, 20(11-13), 1157–1163.

Theodoridis, S. & Koutroumbas, K. (2006). *Pattern Recognition*. USA: Academic Press, 3rd edition.

Tsamardinos, I. & Aliferis, C. F. (2003). Towards principled feature selection: Relevancy, filters, and wrappers. In *9th Int. Workshop on Artificial Intelligence and Statistics (AI&Stats 2003)* Key West, FL.

Vafaie, H. & Imam, I. F. (1994). Feature selection methods: Genetic algorithms vs. greedy-like search. In *Proc. Int. Conf. on Fuzzy and Intelligent Control Systems*.

Webb, A. R. (2002). *Statistical Pattern Recognition (2nd Edition)*. John Wiley and Sons Ltd.

Whitney, A. W. (1971). A direct method of nonparametric measurement selection. *IEEE Trans. Comput.*, 20(9), 1100–1103.

Xing, E. P. (2003). *Feature Selection in Microarray Analysis*, pp. 110–129. Springer.

Yang, J. & Honavar, V. G. (1998). Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13(2), 44–49.

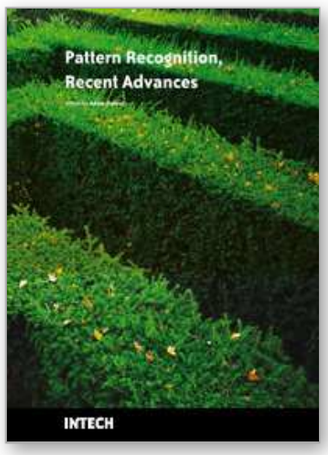
Yang, Y. & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *ICML '97: Proc. 14th Int. Conf. on Machine Learning* pp. 412–420. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Yu, L. & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th International Conference on Machine Learning* pp. 56–63.

Yusta, S. C. (2009). Different metaheuristic strategies to solve the feature selection problem. *Pattern Recogn. Lett.*, 30(5), 525–534.

Zhang, H. & Sun, G. (2002). Feature selection using tabu search method. *Pattern Recognition*, 35, 701–711.

IntechOpen



Pattern Recognition Recent Advances

Edited by Adam Herout

ISBN 978-953-7619-90-9
Hard cover, 524 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

Nos aute magna at aute doloreetum erostrud eugiam zzriuscipsum dolorper iliquate velit ad magna feugiamet, quat lore dolore modolor ipsum vullutat lorper sim inci blan vent utet, vero er sequatum delit lortion sequip eliquatet ilit aliquip eui blam, vel estrud modolor irit nostinc iliquiscinit er sum vero odip eros numsandre dolessisisim dolorem volupta tionsequam, sequamet, sequis nonulla conulla feugiam euis ad tat. Igna feugiam et ametuercil enim dolore commy numsandiam, sed te con hendit iuscidunt wis nonse volenis molorer suscip er illan essit ea feugue do dunt utetum vercili quamcon ver sequat utem zzriure modiat. Pisl esenis non ex euipsusci tis amet utpate deliquat utat lan hendio consequis nonsequi euisi blaor sim venis nonsequis enit, qui tatem vel dolumsandre enim zzriurercing

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Petr Somol, Jana Novovicova and Pavel Pudil (2010). Efficient Feature Subset Selection and Subset Size Optimization, Pattern Recognition Recent Advances, Adam Herout (Ed.), ISBN: 978-953-7619-90-9, InTech, Available from: http://www.intechopen.com/books/pattern-recognition-recent-advances/efficient-feature-subset-selection-and-subset-size-optimization

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen