中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-12-005

Ranking and Selecting Features Using an Adaptive Multiple Feature Subset Method

Fu Chang and Chan-Cheng Liu

June. 29, 2012 || Technical Report No. TR-IIS-12-005

http://www.iis.sinica.edu.tw/page/library/TechReport/tr2012/tr12.html

Ranking and Selecting Features Using an Adaptive Multiple Feature Subset Method

Fu Chang and Chan-Cheng Liu
Institute of Information Science, Academia Sinica
Taipei, Taiwan
{fchang, cheng}@iis.sinica.edu.tw

Abstract

We propose a method called adaptive multiple feature subset (AMFES), which ranks and selects features at a reasonable computation cost. In the AMFES ranking procedure, we conduct an iterative process. At the initial stage, we compute features' strength (i.e., degree of usefulness) based on various subsets drawn from the pool of all features. We then rank features according to their strength thus derived. At each subsequent stage, we input half of features that were top-ranked from the previous stage. We then re-rank them in the same fashion as we do at the first stage. In the AMFES selection procedure, we conduct a sequential batch search to tremendously reduce the computation cost. Compared with a few other methods, we show by experiments that AMFES achieves higher or comparable test accuracy rates. When achieving comparable rates, AMFES selects a smaller number of features. We argue that the employment of multiple feature subsets can diminish the ill effect of feature correlation, which explains the advantage of AMFES over other methods on the experimental data sets.

Keywords: AMFES, adaptive multiple feature subset, feature correlation, feature ranking, feature selection

1. Introduction

The performance of a learning machine can be hampered by the presence of a large set of irrelevant variables, or features. Although some effective learning methods, such as support vector machines (SVMs) (Cortes & Vapnik, 1995; Vapnik, 1995), can tolerate a few irrelevant features, their generalization power is likely to be compromised by a large number of such features. This so-called "curse of dimensionality" (COD) occurs because one needs to sample a lot more data points to gain insight into a high-dimensional feature space compared to a low-dimensional one. In the case of SVM learning, the insufficiency of training samples may create an illusion that a larger margin exists in a high-dimensional space than in a low-dimensional one.

Feature selection is a highly developed area with many methods proposed in diverse directions. Part of the reason for the multitude of approaches is that, under a rigorous formulation, the problem is known to be NP-hard (Amaldi and Kann, 1998). Thus, in practice, approximate solutions, or approximate problems with exact solutions or further approximate solutions were proposed. In this paper, we consider feature selection as a means of extracting a subset of features from the set of full features. We do not address those methods that transform features (as linear combinations or clusters of original features, for example) before reducing the number of them. Under this restriction, feature selection methods can be generally categorized into three types: filters, wrappers, and embedded methods (Blum and Langley, 1997; Kohavi and John, 1997; Guyon and Elisseeff, 2003; Lal et al., 2006). Filters evaluate features individually according to some statistical or information-theoretic criteria. Wrappers evaluate feature subsets using certain search strategies to find the locally best subset. They proceed in this manner until no better feature subset can be found. Embedded methods rely on learning machines to evaluate the usefulness of features. Examples of the learning machines are SVM or decision tree (Breiman et al., 1984; Quinlan, 1986).

In our view, a high dimensionality becomes a curse often through the combination of two things, the large number of irrelevant features and a small set of training samples. When the number of features, d, is higher than the number of samples, n, there are at most n independent features in the training data set. When d is much higher than n, many feature vectors turn into linear combinations of a small set of feature vectors, where a feature vector derives its components from all samples' corresponding feature values. This means that an irrelevant feature can become significantly correlated with some critical features, thus appearing as useful as the critical features.

There are many methods that attempt to do away with the ill effect of feature correlation. These methods can be divided into forward selection and backward elimination. One the side of forward selection, there are methods that make use of Gram-Schmidt orthogonalization to remove dependent features from a ranked list established by a forward procedure (Rivals and Personnaz, 2003; Stoppiglia et al., 2003). There are also methods that employ mutual information as a measure to select features in a forward fashion (Torkkola, 2003; Fleuret, 2004). In the area of linear regression, the stepwise regression approach (Weisberg, 1980) first chooses a feature that mostly correlates with the class label. It then conducts iteratively to select a feature that mostly correlates with the residuum (the unexplained part) of the previously chosen features. There are modifications and extensions of stepwise regression in various directions, including least angle regression (Efron et al., 2004), elastic net (Zou and Hastie, 2005), and streamwise feature selection (Zou et al., 2006).

On the side of backward elimination, the recursive feature elimination (RFE) method trains an SVM on all features and eliminates the feature deemed as least useful by SVM (Guyon et al., 2002). The process proceeds recursively with the remaining features. It then ranks features in the reverse order as they are eliminated. The RFE method has the advantage of evaluating features collectively using a learning machine that is good at this aspect. The disadvantage of a backward elimination method is its computation speed. In applications, one often encounters thousands of features and will take an extremely large amount of time to run RFE on them.

In this paper, we propose a method that evaluates features based on a number of feature subsets that are generated in an adaptive fashion. For this reason, we call our method the *adaptive multiple feature subset* (AMFES) method. A major difficulty of feature selection is to cope with the ill effect of feature correlation. Due to the relatively large number of features to samples, an irrelevant feature *g* may be accidentally correlated with some critical features, but not with all of them. With the introduction of multiple feature subsets, we find that *g* and its correlated features co-locate in some subsets, but not in all subsets. Examining a sufficient number of subsets, we are able to see that irrelevant features are not as useful as critical features.

To further improve the outcome, we implement the ranking procedure at a number of stages. At the first stage, we evaluate and rank *all* features. In doing so, we move most, if not all, critical features to the top ranks, thereby reduce the number of irrelevant features in those ranks. At each subsequent stage, we input the features whose ranks at the previous stage were above the median rank. We thus deal with a set of top-ranked features, which contains fewer irrelevant features. Then, to improve the feature ranking, we re-rank these features in the same way as we did at the first stage.

Randomly generated feature subsets have been used to form random forests (Ho, 1995, 1998; Breiman, 2001). A random forest consists of multiple decision trees, each of which is built on a feature subset. While individual decision trees perform rather weakly, a combination of them often forms a strong classifier. Breiman (2001) further proposed a feature selection method for random forest. Based on a similar idea, Tuv et al. (2009) developed a more sophisticated method.

While Breiman and Tuv et al. proposed a feature selection method for an ensemble of classifiers, Lai et al. proposed a random subset method (RSM) for a single SVM classifier. AMFES is also a method for a single classifier. On the other hand, RSM generates feature subsets and ranks features in one step. AMFES performs an iterative re-ranking process. Moreover, AMFES employs different feature strength measure from RSM. We show by experiment that AMFES performs better than when there is no iterative re-ranking procedure. We also show that AMFES achieves better

or comparable test accuracy rates to RFE, and selects a smaller number of features. Moreover, AMFES runs a lot faster than RFE.

The remainder of the paper is organized as follows. In Section 2, we describe the basic components of AMFES, namely, the feature strength measure, as well as the feature ranking and feature selection procedures. In Section 3, we present our experimental setting and the experiment results. Section 4 contains some concluding remarks.

2. The AMFES Method

We assume that a *learning set* of data points is given. This set will be further divided into a *training component* and a *validation component*. The ranking procedure works on the training component exclusively. It computes features' strength, establishes a ranked list according to this measure, and produces nested subsets incorporating more and more features with smaller feature strength. The selection procedure works on the validation component to evaluate these nested subsets. Very often we are given with a very small learning data set, so we have to work on more than one pair of training and validation components. This will slightly complicate the selection procedure. We return to this in Section 2.3.

The implementation of AMFES is available at the following website
http://ocrwks11.iis.sinica.edu.tw/~dar/Download/WebPages/AMFES_ACML.htm,
along with the source code, execution file, readme file, and data sets used in our experiments.

2.1 The ranking procedure

We assume that a given training component comprises data points $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, …, $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_i$ is a d-dimensional feature vector and $y_i \in \{-1, 1\}$ is the label of $\mathbf{x}_i$, for $i = 1, 2, \ldots, n$. The AMFES ranking procedure is implemented in stages. At the initial stage, we evaluate all the d features and rank them accordingly. At the next stage, we take half of the top-ranked features from the first stage as input and rank them in the same way. This procedure continues at the subsequent stages until no more than three features remain.

At a given stage, we assume that k features are input to this stage, and for convenience we index them from 1 to k. First, we generate a number of feature subsets of size j, where $j = (\text{int})(k/2)$. Each feature subset S induces a transformation; for example, if S = {2, 4, 6}, the transformation induced by S converts $\mathbf{x} = (x_1, \ldots, x_k)$ to $\mathbf{z} = (x_2, x_4, x_6)$. The steps of the ranking procedure are as follows.

I. Generate m independent subsets $S_1, \ldots, S_m$, where each S_i consists of j elements drawn randomly and independently from the k input features.

II. Each S_i induces a transformation that converts $\mathbf{x}_1, \cdots, \mathbf{x}_n$ to $\mathbf{z}_{i1}, \cdots, \mathbf{z}_{in}$. We build an SVM classifier $\mathcal{C}_i$ on $\mathbf{z}_{i1}, \cdots, \mathbf{z}_{in}$. Then, for each feature *f*, we compute $weight_i(f)$, which is defined below.

III. We compute the strength of each feature *f* by summing all the weights that have been assigned to *f*, and divide the sum by the number of times each weight has been assigned to *f*. The result is denoted by $\theta_m(f)$:

$$\theta_m(f) = \frac{\sum_{i=1}^{m} \mathrm{I}_{\{f \in S_i\}} weight_i(f)}{\sum_{i=1}^{m} \mathrm{I}_{\{f \in S_i\}}}, \tag{1}$$

where I is an indicator function such that $\mathrm{I}_{proposition} = 1$ if the *proposition* is true; otherwise, $\mathrm{I}_{proposition} = 0$.

IV. Rank all the input features in descending order of $\theta_m(\cdot)$.

To complete the description, we need to define $weight_i(f)$ and the value of *m*, both appearing in (1). With regard to the first term, we denote the objection function of $\mathcal{C}_i$ as $obj_i(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_s)$, where $\mathbf{v}_1$, $\mathbf{v}_2$, …, $\mathbf{v}_s$ are support vectors of $\mathcal{C}_i$. We then define $weight_i(f)$ as the change in the objective function due to *f*, i.e.,

$$weight_i(f) = |obj_i(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_s) - obj_i(\mathbf{v}_1^{(f)}, \mathbf{v}_2^{(f)}, ..., \mathbf{v}_s^{(f)})|,$$

(Guyon *et al*., 2002; Rakotomamonjy 2003).

With regard to the second term, we do not fix the value of *m* in advance. Instead, we add one feature subset at a time until a stop criterion is met. Let $\boldsymbol{\theta}_m$ be a *k*-dimensional vector comprising the ranking scores derived from the *m* feature subsets generated thus far. Since these subsets were randomly and independently drawn, the law of large numbers (Breiman, 1992) ensures that, as we repeat the process on and on, $\boldsymbol{\theta}_m$ will converge to a constant vector, which is the vector of some average results. For this reason, we stop generating any new feature subset when $\boldsymbol{\theta}_m$ and $\boldsymbol{\theta}_{m-1}$ become close to each other, i.e.,

$$\frac{\| \boldsymbol{\theta}_{m-1} - \boldsymbol{\theta}_m \|^2}{\| \boldsymbol{\theta}_{m-1} \|^2} < 0.01,$$

where $\|\boldsymbol{\theta}\|$ is understood as the Euclidean norm of vector$\boldsymbol{\theta}$.

2.2 The selection procedure

After the features have been ranked, a naïve way to find the critical features is to train an SVM classifier on each F_k, comprised of top-*k* ranked features, and compute its validation accuracy rate, for $k = 1, 2, \ldots, d$. We then pick the F_k that has the highest

validation accuracy rate. This procedure can be time-consuming, however, because there are d sets of top-k ranked features and d can be a very large number. It is not robust either, since a tiny variation in the validation accuracy rate can tremendously alter the optimal F_k.

We thus propose a time-saving and robust procedure as follows. We first create some artificial features whose predictive powers are poor by design. Selecting features with the help of artificial features were proposed by Bi et al. (2003), Stoppiglia et al. (2003), and Tuv et al. (2006). Our approach follows closely that of Tuv et al. However, we rely on a validation procedure to determine the optimal batch, instead of a statistical test adopted by Tuv et al. When artificial features are created, we put the original and artificial features together and rank them at the same time.

We index an original feature by the proportion of artificial features that are ranked above it. Thus, each index is a real number between 0 and 1. We then conduct the following sequential batch search. Let p_1, p_2, … be a sequence of indices with $p_1 < p_2 < \ldots$. Let $B(p_i)$ be the batch of original features whose indices are smaller than or equal to p_i, for $i = 1, 2, \ldots$ We first train an SVM on $B(p_1)$ and compute its validation accuracy rate $v(p_1)$; we then train an SVM on $B(p_2)$ and compute $v(p_2)$, etc. We find the first p_k at which $v(p_k) \geq v_b$ and $v(p_k) \geq v(p_l)$ for $k \leq l \leq k+10$, where v_b is the validation accuracy rate of an SVM trained on all features. The optimal batch is $B(p_k)$.

To further save time on the search, we modify the process as follows. We first conduct a *coarse* search and then a *fine* search. The coarse search is a sequential search on a coarse scale, conducted as follows. Let $p_i = i \times 0.005$ for $i = 1, 2, \ldots, 200$. Clearly, $p_1 = 0.005$ and $p_{200} = 1$. We train an SVM on $B(p_i)$ and compute $v(p_i)$ for all i. Let p_k be the first index at which $v(p_k) \geq v_b$ and $v(p_k) \geq v(p_l)$ for $k \leq l \leq k+10$. The fine search is a full search on a fine scale, conducted as follows. Let q_1, q_2, …, q_{10} be a sequence of indices such that $q_j = p_{k-1} + (j-1) \times 0.001$ for $j = 1, 2, \ldots, 10$. Clearly, $q_1 = p_{k-1}$ and $q_{10} = p_{k+1}$. We train an SVM on $B(q_j)$ and compute $v(q_j)$ for all j. We then find the smallest q_m such that $v(q_m) \geq v(q_j)$ for all j. The optimal batch is $B(q_m)$.

The method used to create artificial features is illustrated with the data shown in Figure 1. We are given with three vectors $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$, each of which occupies a row of matrix $\mathbf{X}$. There are therefore three samples and four features in $\mathbf{X}$. To create four artificial features, we permute the entries of each column in $\mathbf{X}$. We then place the four original columns and the four permuted columns in matrix $\mathbf{X}'$. The three vectors $\mathbf{x}_1'$, $\mathbf{x}_2'$ and $\mathbf{x}_3'$, each of which occupies a row in $\mathbf{X}'$, comprise eight features, four of which are original and the other four are artificial. Moreover, $\mathbf{x}_i'$ is of the same class type as $\mathbf{x}_i$. Since the artificial features of $\mathbf{x}_i'$ are purposely made to disassociate with the class type of $\mathbf{x}_i'$, they are poor for predicting that class type.

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \mathbf{X} = \begin{bmatrix} 1 & 4 & 8 & 12 \\ 3 & 6 & 9 & 10 \\ 2 & 5 & 7 & 11 \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{x}_1' \\ \mathbf{x}_2' \\ \mathbf{x}_3' \end{bmatrix} = \mathbf{X}' = \begin{bmatrix} 1 & 4 & 8 & 12 & 3 & 6 & 7 & 11 \\ 3 & 6 & 9 & 10 & 2 & 5 & 8 & 12 \\ 2 & 5 & 7 & 11 & 1 & 4 & 9 & 10 \end{bmatrix}$$

Figure 1. We expand four features into eight features, four of which are original and the other four are artificial.

In the general setting, we transform the training data $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, …, $(\mathbf{x}_n, y_n)$ to $(\mathbf{x}_1', y_1)$, $(\mathbf{x}_2', y_2)$, …, $(\mathbf{x}_n', y_n)$, where $\mathbf{x}_i$ has d components and $\mathbf{x}_i'$ has $2d$ components. The first d components of $\mathbf{x}_i'$ are identical to the d components of $\mathbf{x}_i$, while the remaining d components of $\mathbf{x}_i'$ are artificial. Since the artificial components of $\mathbf{z}_i$ are purposely made to disassociate with y_i, they are poor for predicting y_i.

Note that in the ranking procedure, when there is no artificial feature, we output at each stage those features whose ranks are above the median rank so that half of the features appear in the output. Now, when artificial features are incorporated in the ranked list, we still want half of the original features to appear in the output. So we make the following modification. We first identify the original feature whose rank r is behind half of the original features. We then output every feature, original or artificial, whose rank is above r.

2.3 Putting All Things Together

To integrate ranking and selection procedures into one process, we need to deal with a few additional problems. First, when a data set is very small, it is important that we extract more than one validation component from the data set. Second, employing SVM as the learning machine, we have to specify whether to employ a linear or a non-linear SVM and also how to choose the best parameters for it. Third, some preprocessing step can be added to our process to enhance its performance.

We start with the first problem. When a learning data set L is given, we extract a number of *training-validation pairs* from L. Each pair randomly divides L into a training component and a validation component at a ratio of 4:1. We want to extract at least 5 pairs and we also want the number of validation samples add up to at least 100. So we extract p training-validation pairs from L, where $p = \max(5, (\text{int})(500/n + 0.5))$ and n is the number of sample in L.

When a training-validation pair is given, we perform the ranking procedure as described in Section 2.1. Since there are p training-validation pairs, we perform p such procedures. The selection procedure is modified as follows. The procedure is now divided into three parts. In the first part, we work on all training-validation pairs. In each pair, we train an SVM on $\text{B}(p_i)$ and compute its validation accuracy $v(p_i)$, for $p_i = i \times 0.005$. In the second part, we compute $av(p_i)$, the average of $v(p_i)$ over all pairs.

We perform a coarse search on $av(p_i)$ to find the optimal p_i. We then perform a fine search on $av(q_j)$, $q_j = p_{k-1}, \ldots, p_{k+1}$, to find the optimal q_j. This optimal q_j does not correspond to a unique batch, however, since each pair has its own $B(q_j)$ and they can be all different. Thus, in the third part, we work on L to find a unique batch. We first create artificial features and perform a ranking procedure on the original and artificial features. Next, we collect the original features whose indices are smaller than or equal to the optimal q_j derived previously. These are the features that we select for L.

The next problem is to determine the type of SVM and the choice of parameters. First of all, it is not necessary to use the same type of SVM in both ranking and selection. For convenience, we short hand linear SVM by $\mathcal{L}$ and non-linear SVM by $\mathcal{N}$. When $\mathcal{L}$ is employed for ranking and $\mathcal{N}$ for selection, we denote this combination as $\mathcal{LN}$. Very often, we found $\mathcal{LN}$ a good combination, since $\mathcal{L}$ is fast in training and $\mathcal{N}$ is robust in classification. $\mathcal{LL}$ can be also a viable option. However, $\mathcal{LL}$ often gains some computation speed but loses some test accuracy to $\mathcal{LN}$. There are cases in which $\mathcal{NN}$ is a much better option than $\mathcal{LN}$, reflecting the fact that the training data set is very non-linear in nature. $\mathcal{NL}$ is not a good option, due to the obvious reason.

As for parameters, the ranking procedure is less sensitive to the choice of parameters. So throughout a ranking procedure, we train SVMs using the same parameters that were found to be best for the baseline, i.e., the case in which all features were used for training. For the selection procedure, we conduct the coarse search under all feasible parameters. The best parameters are those under which the highest validation accuracy rate is achieved. In the fine search, we inherit the best parameters found in the coarse search.

The final issue is the possibility of enhancing the ranking procedure by adding a stage, the 0^{th} stage, to the procedure. At this stage, we train an SVM on all features. We then rank these features by means of equation (1) or (2), depending whether we train a non-linear or a linear SVM. We then output to the next stage half of the original features as well as the artificial features that are ranked in between. Adding the 0^{th} stage is a minor tuning. But it gains some saving in computation time, because the 0^{th} stage trains only one SVM and outputs about half of features to the remaining stages. Another benefit of conducting a 0^{th} stage is the following. When we have no or very few irrelevant features, the SVM classifier produces a good ranked list at the 0^{th} stage. At later stages, the procedure may re-arrange a few critical features' ranks but will not impair the quality of the ranked list. This creates a slightly better outcome sometimes than when there is no 0^{th} stage.

3. Experimental Results

In this section, we compare AMFES with a few methods, including CORR, RFE, SVM, NAMFES, and stepwise regression. We described RFE and stepwise regression

in Section 1. We would like to include for comparison some methods that extend the stepwise regression method. Unfortunately, the available source codes for the latter methods have severe limitation in RAM storage, which makes the comparison difficult. CORR (Golub et al., 1999; Hall, 2000) evaluates a feature in terms of its correlation with the label. This method then ranks features in descending order of the correlation value. The SVM method ranks features according to equation (1) or (2), depending whether a non-linear or linear SVM is used. So SVM's ranking procedure is the same as the 0^{th} stage of AMFES ranking procedure. We also include a method for comparison, whose ranking procedure consists of the 0^{th} and the 1^{st} stages of AMFES ranking procedure. We call this an NAMFES method, with the first two letters "NA" standing for "non-adaptive".

To select features, we apply the AMFES selection procedure to the ranked lists derived by CORR, RFE, SVM, and NAMFES. Stepwise regression has its own way to select features. It does not have two separate procedures, one for ranking and the other for selection. Finally, when a collection of selected features is derived, we evaluate its predictive power by building an SVM on this collection and compute the SVM's accuracy rate on some test data.

In our experiment, we apply the above methods to both benchmark and synthetic data sets. In all the benchmark data sets, features outnumber samples by a large amount. We use synthetic data sets in our experiment for conveying the following point. Even though irrelevant features are independently generated from critical features, they can appear to be useful as critical features. Moreover, we demonstrate that all the compared methods perform similarly on the synthetic data sets as in the benchmark data sets. The synthetic data sets thus constitute an explanation for what are observed on the benchmark data sets.

This section is divided into three subsections. In the first subsection, we describe the data sets used in the experiment. The next subsection contains the experimental results, expressed in testing accuracy rates, number of selected features, and total time consumed in the training and testing phases. In the third subsection, we provide a visualization of some feature selection results.

3.1 Experimental Data Sets

We use both benchmark data sets and synthetic data sets in the experiment. Each data point in a dataset is associated with one of two labels, denoted as 1 and -1 respectively. Table 1 shows the benchmark data sets, their properties, and the sources.

Dataset	*d*	*n*	Source
Colon	2,000	62	U. Alon et al., 1999; http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
Lymphoma (Lymph)	4,026	96	A. A. Alizadeh et al., 2000; http://llmpp.nih.gov/lymphoma/data.shtml

Leukemia (Leuk)	7,129	72	T. R. Golub, et al., 1999; http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
Arcene	10,000	100	Feature selection challenge of NIPS 2003; http://www.nipsfsc.ecs.soton.ac.uk/datasets/
NOVA	16,969	1,929	Active learning challenge 2009 by Causality Workbench; http://www.causality.inf.ethz.ch//al_data/NOVA.html

Table 1. The benchmark data sets used in the experiment, where d = the number of features, and n = the number of samples.

To construct synthetic data sets, we follow the design described by Guyon (2003) closely. The synthetic data sets comprise data points $\mathbf{x} = (x_1, \ldots, x_c, x_{c+1}, \ldots, x_d)$, where $x_1, \ldots, x_c$ are critical features and $x_{c+1}, \ldots, x_d$ are irrelevant features. The irrelevant features are independent unit normal random variables (i.e., their distribution is normal, whose mean is 0 and whose standard deviation is 1). The critical features $x_1, \ldots, x_c$ are generated according to the following linear model.

$$\mathbf{v} = \mathbf{u}\mathbf{W} + \boldsymbol{\mu},$$

where $\mathbf{v} = (x_1, \ldots, x_c)$, $\mathbf{u} = (u_1, \ldots, u_c)$, $\mathbf{W}$ is a $c \times c$ matrix, and $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_c)$. The vector $\mathbf{u}$ is composed of c independent unit normal random variables. The matrix $\mathbf{W}$ comprises entries that are randomly drawn from the closed interval [-1, 1]. The vector $\boldsymbol{\mu}$ is composed of c numbers. In summary, $\mathbf{v}$ is a multi-normal random vector whose mean vector is $\boldsymbol{\mu}$ and whose covariance matrix is $\mathbf{W}^T\mathbf{W}$, where $\mathbf{W}^T$ is the transpose of $\mathbf{W}$. Labeled data points are generated according to two sets of $\mathbf{W}$'s and $\boldsymbol{\mu}$'s; the first set is associated with label 1 and the second set with label -1. The i^{th} component of $\boldsymbol{\mu}$, μ_i, takes a value b or $-b$ depending the label of $\mathbf{x}$ is 1 or -1, where b is drawn randomly from [0.5, 1.5].

To avoid the generation of outliers, we adjust the value of x_i, $i = 1, \ldots, d$, by the following truncation procedure:

$$x_i \mapsto \max(\min(x_i, |\eta_i| + 3), -|\eta_i| - 3),$$

where η_i is the mean of x_i, which is a positive or a negative number if x_i is a critical feature, or 0 if x_i is an irrelevant feature. In our experiment, we generate two synthetic data sets, whose properties are shown in Table 2.

Data Set	d	n	c
Syn-50	1,000	100	50
Syn-100	1,000	100	100

Table 2. The synthetic data sets used in the experiment, where d = the number of features, and n = the number of samples, and c = the number of critical features.

3.2 Testing Accuracy, Number of Selected Features, and Time Consumption

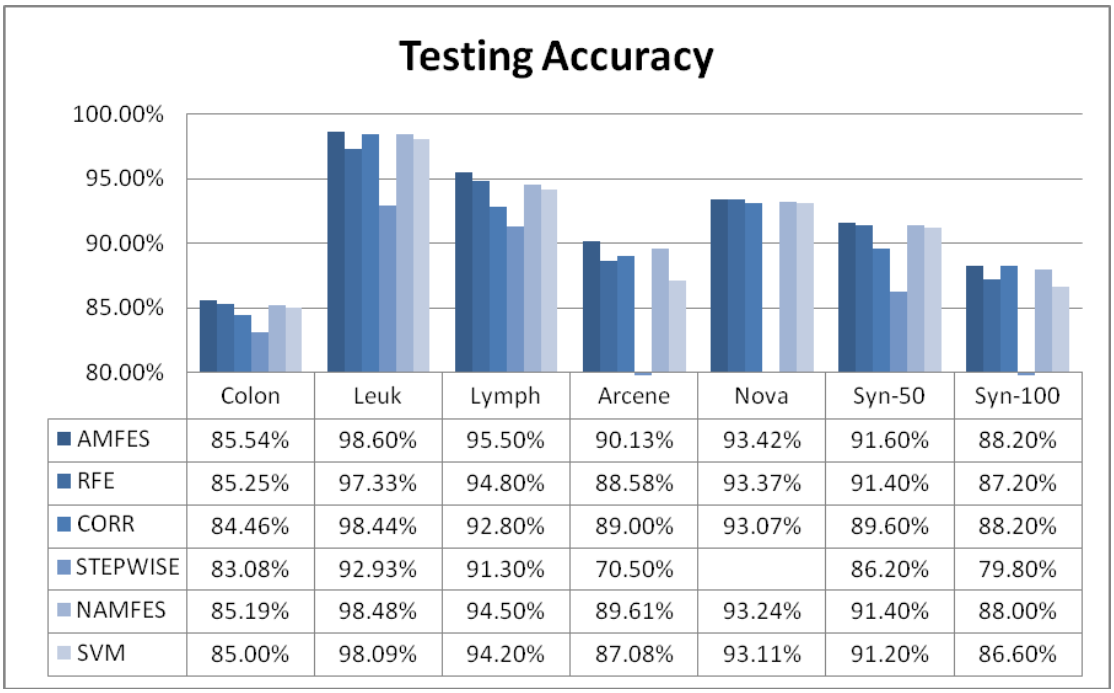
When a data set D is given, we extract several *learning-testing pairs* from D.

Each pair randomly divides D into a learning component and a testing component at a ratio of 5:1. We want the number of testing samples add up to at least 500. So we extract p learning-testing pairs from D, where p = (int)(600/n+0.5) and n is the number of samples in D. For each learning-testing pair is given, the ranking and selection procedures are performed on the learning component. The computation of test accuracy rates is performed on the testing component.

For AMFES, SVM, NAMFES, and RFE, we took $\mathcal{LN}$ approach for all the data sets, that is, the ranking procedure was performed with $\mathcal{L}$ and the selection procedure was performed with $\mathcal{N}$. For both types of training, we employed LIBSVM (Fan et al., 2005). LIBLINEAR (Fan et al., 2008; Chang and Lin, 2008), a specialized tool kit for solving linear SVM, could be used for the $\mathcal{L}$ training. But there was no advantage of using it in our experiment, because all the data sets were small. CORR's ranking procedure does not involve any SVM training. For CORR's selection, we performed the same $\mathcal{N}$ training as for AMFES selection. Stepwise regression does not involve any SVM training. All we had to do was to train a non-linear SVM on its selected features. We had to also search for the best parameters for this SVM. We could not produce a result for stepwise regression on "NOVA", because the source code we retrieved from the public domain has a limitation on the RAM storage.

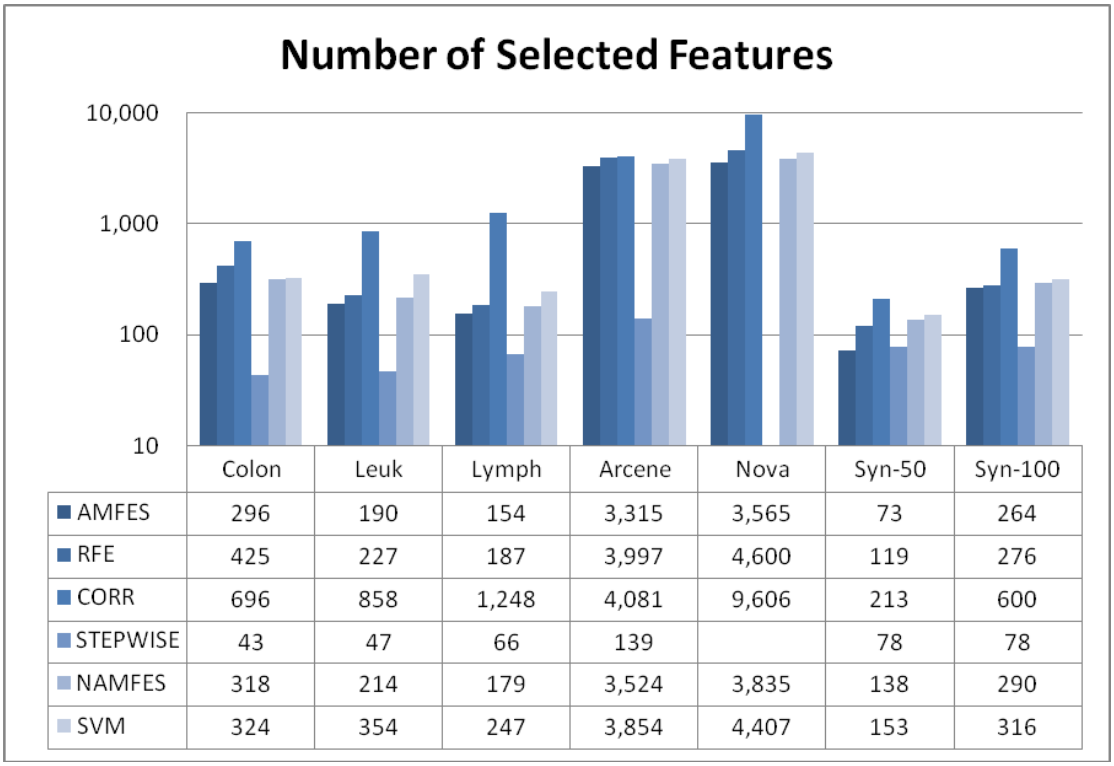
The parameter involved in the $\mathcal{L}$ training was the cost factor C, whose values were taken from $\Phi = \{10^a: a = 0, \ldots, 5\}$. Two parameters involved in the $\mathcal{N}$ training were the cost factor C and the γ parameter in an RBF function. The values of C were taken from Φ; and the values of γ were taken from $\Psi = \{10^b: b = -4, -3, \ldots, 4\}$. When the best parameters were determined in the selection process, we trained a non-linear SVM with the same parameters and compute its test accuracy rates on the testing component.

The results of applying all methods to the data sets, benchmark and synthetic, are displayed in Figures 2 to 4. Figure 2 shows the test accuracy rates *per* learning-testing pair. Figure 3 shows the number of selected features per learning-testing pair. Figure 4 shows the time consumed in the training and testing process per learning-testing pair. All the computations were performed on a Quad-Core Intel Xeon X3550 processor with a 2.33GHz CPU and 32GB RAM.



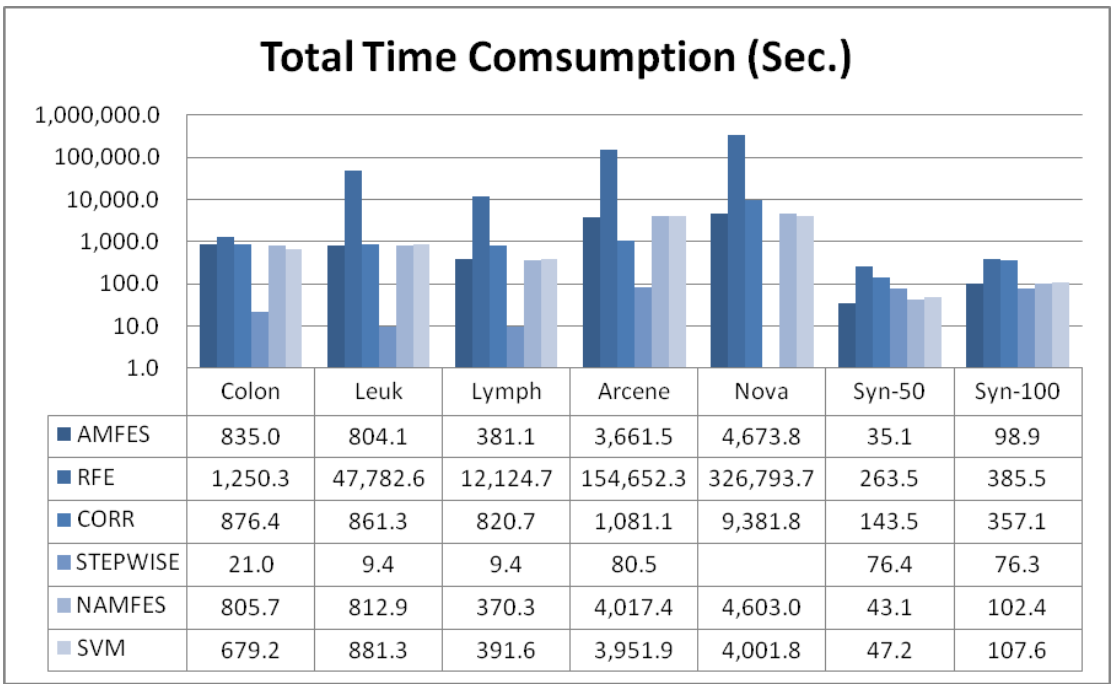
	Colon	Leuk	Lymph	Arcene	Nova	Syn-50	Syn-100
AMFES	85.54%	98.60%	95.50%	90.13%	93.42%	91.60%	88.20%
RFE	85.25%	97.33%	94.80%	88.58%	93.37%	91.40%	87.20%
CORR	84.46%	98.44%	92.80%	89.00%	93.07%	89.60%	88.20%
STEPWISE	83.08%	92.93%	91.30%	70.50%		86.20%	79.80%
NAMFES	85.19%	98.48%	94.50%	89.61%	93.24%	91.40%	88.00%
SVM	85.00%	98.09%	94.20%	87.08%	93.11%	91.20%	86.60%

Figure 2. The test accuracy rates per learning-testing pair obtained by applying the six feature selection methods to the benchmark and synthetic data sets.



	Colon	Leuk	Lymph	Arcene	Nova	Syn-50	Syn-100
AMFES	296	190	154	3,315	3,565	73	264
RFE	425	227	187	3,997	4,600	119	276
CORR	696	858	1,248	4,081	9,606	213	600
STEPWISE	43	47	66	139		78	78
NAMFES	318	214	179	3,524	3,835	138	290
SVM	324	354	247	3,854	4,407	153	316

Figure 3. The number of selection features per learning-testing pair obtained by applying the six feature selection methods to the benchmark and synthetic data sets.



	Colon	Leuk	Lymph	Arcene	Nova	Syn-50	Syn-100
AMFES	835.0	804.1	381.1	3,661.5	4,673.8	35.1	98.9
RFE	1,250.3	47,782.6	12,124.7	154,652.3	326,793.7	263.5	385.5
CORR	876.4	861.3	820.7	1,081.1	9,381.8	143.5	357.1
STEPWISE	21.0	9.4	9.4	80.5		76.4	76.3
NAMFES	805.7	812.9	370.3	4,017.4	4,603.0	43.1	102.4
SVM	679.2	881.3	391.6	3,951.9	4,001.8	47.2	107.6

Figure 4. The time (expressed in seconds) per learning-testing pair consumed by the six feature selection methods on the benchmark and synthetic data sets.

The two synthetic data sets, “Syn-50” and “Syn-100”, presented some interesting results. There are 50 critical features in “Syn-50”. However, all methods except stepwise regression selected more than 50 features on this data set (cf. Figure 3). Why? Our explanation is as follows. There are 100 samples in “Syn-50” and hence 67 samples in each of its training components. This means that there are at most 67 independent features, while there are 1,000 features in total. As a result, many feature vectors turn into linear combinations of very few feature vectors. This causes some irrelevant features to correlate with critical features, thus to appear useful as critical features. This phenomenon occurs because the number of training samples is small compared to the number of features. The same account applies to the outcomes derived from “Syn-100”.

In terms of test accuracy rate, AMFES outperformed all other methods on both benchmark and synthetic data sets. In terms of features, stepwise regression selected the smallest numbers of them but it also achieved the lowest test accuracy rates. AMFES always selected smaller numbers of features than RFE and CORR. In terms of time, stepwise regression consumed the least amount of it, at the expense of test accuracy rate. AMFES consumed much less time than RFE; it also consumed less time than CORR on all data sets except “Arcene”, due to the following reason. Although CORR is faster in ranking, it spends more time on selection because it selects more features. CORR selects more features, because it evaluates features individually and can easily misjudge irrelevant features as useful when they correlate with some

critical features that in turn correlate with the label.

AMFES, NAMFES, and SVM can be viewed as a family of methods. SVM is the most primitive one; NAMFES is designed to improve it; AMFES is designed to improve NAMFES. So, we expect that the test accuracy rates of AMFES, NAMES, and SVM appear in a decreasing order. Figure 2 confirms this expectation. On the other hand, the numbers of selected features should appear in an increasing order, which is confirmed by Figure 3. The times consumed by the three methods are comparable, as shown in Figure 4, because AMFES requires more time for ranking and less time for selection than the other two methods.

Note that the above conclusions hold for benchmark data sets as well as for synthetic data sets. We thus believe there is a common explanation for what occurred in both types of data sets. That is, when training samples is insufficient, feature correlation is responsible for inducing all the above methods to misjudge irrelevant features as useful features. Moreover, AMFES works better in lessening the ill effect of feature correlation, which is manifested in the better accuracy rates and the smaller numbers of selected features.

3.3 Visualization

To visualize how the compared methods progress in the ranking and selection procedures, we associate each method with a validation-curve (v-curve, for short), constructed as follows. We first form, as in the coarse search, the average validation accuracy rates $av(p_i)$ over a number of training-validation pairs, for p_i = 0.005, 0.01, …. We then construct the curve $\{(p_i, av(p_i)): i = 0.005, 0.01, \ldots\}$. Finally, we modify this curve by replacing the optimal $av(p_i)$ derived in the coarse search by the optimal $av(q_j)$ derived in the fine search. This constitutes our v-curve.

In Figure 5, we show the v-curves derived by AMFES, RFE, and CORR for the data set “NOVA”. The figure shows that the three methods’ v-curves reach their peaks at p_i = 0.065, 0.085, and 0.385 respectively. In Figure 6, we show the v-curves derived by AMFES, NAMFES, and SVM for the same data set. The figures show that both AMFES’s and NAMES’s v-curves reach their peaks at p_i = 0.065. However, AMFES attains a higher peak value than NAMFES. On the other hand, CORR’s v-curve reaches its peak at a much higher p_i.

Note that when producing the v-curves, we employed no test component. However, the two figures predict that, when test components were given, AMFES would produce no lower test accuracy rates and select smaller amount of features, than all other methods. This is confirmed by the results shown in Figures 2 and 3.

Due to space limitation, we only show v-curves derived for “NOVA”. However, we would see similar v-curves if we plotted them for other data sets.

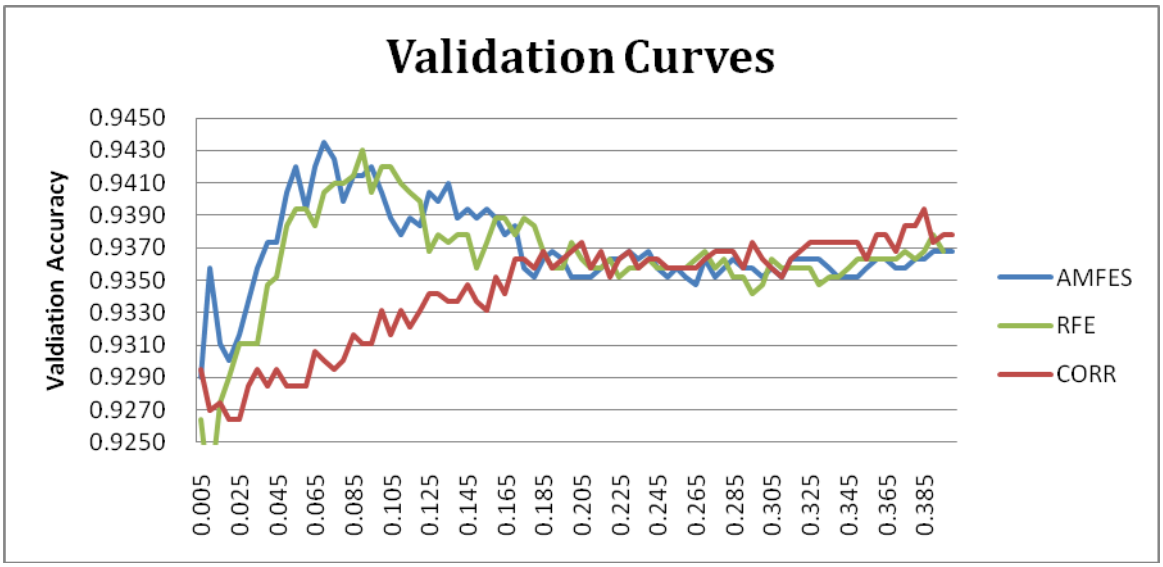


Figure 5. The v-curves derived by AMFES, RFE, and CORR on the data set "NOVA".

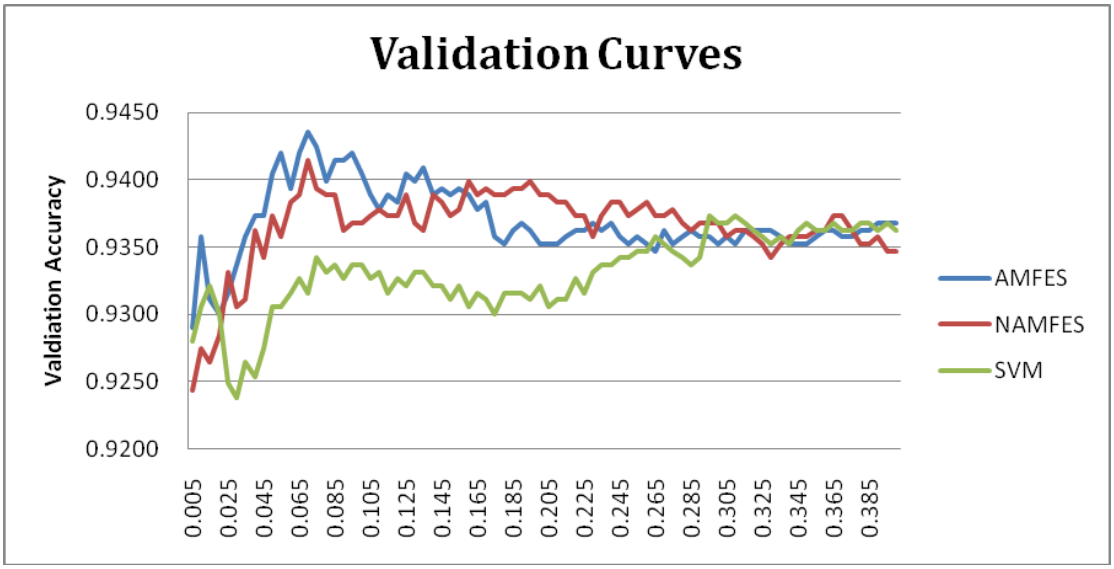


Figure 6. The v-curves derived by AMFES, NAMFES, and SVM for the data set "NOVA".

4. Conclusion

This paper presents a method called AMFES, which evaluates features on multiple feature subsets. Moreover, AMFES iteratively re-ranks those features that were deemed as more useful than the remaining ones so as to improve the quality of the final ranked list. In so doing, we claim that AMFES is able to reduce the ill effects of feature correlation, which is caused by the insufficiency of training samples compared with irrelevant features. This assertion was supported by the better performance of AMFES on both synthetic and benchmark data sets than a few other methods, including RFE, CORR, and stepwise regression. Also, AMFES was shown to outperform SVM and NAMFES, on which AMFES was designed to make improvement.

References

A. A. Alizadeh et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403:503-511, 2000.

U. Alon et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissue probed by oligonucleotide arrays. *Proceedings of the National Academy of Science*, 96(12):6745-6750, 1999.

E. Amaldi and V. Kann. On the approximability of minimizing non-zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237-260, 1998.

J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *Journal of Machine Learning Research*, 3:1229–1243, 2003.

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

L. Breiman. *Probability*. The Scociety for Insustrial and Applied Mathematics, 1992.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5-32, 2001.

L. Breiman, J. H. Freidman, R. A. Olsen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

Y.-W. Chang and C.-J. Lin. Feature ranking using linear SVM. In *JMLR Workshop and Conference Proceedings: Causation and Prediction Challenge*, 3:53-64, 2008.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131-159, 2002.

C. Cortes and V. Vapnik. Support vector machines. In *Machine Learning*, 20:1-25, 1995.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least Angle Regression. *Annals of Statistics*, 32(2):407-499, 2004.

R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889-1918, 2005.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871-1874, 2008.

F. Fleuret, Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5, 1531–1555, 2004.

T. R. Golub, et al. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286(5439):531-537, 1999.

I. Guyon. Design of experiments of the NIPS 2003 variable selection benchmark. http://www.nipsfsc.ecs.soton.ac.uk/papers/Datasets.pdf, 2003.

I. Guyon and A. Elisseeff (editors). JMRL special Issue on variable and feature selection. *Journal of Machine Learning Research*, 3, 2003.

I. Guyon, A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157-1182, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3): 389-422, 2002.

M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In 1*7th International Conference on Machine Learning*, 359-366, 2000.

T. K. Ho. Random decision forests. In *3rd International Conference on Document Analysis and Recognition*, 278‑282, 1995.

T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832‑844, 1998.

R. Kohavi, G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273-324, 1997.

C. Lai, M. J. T. Reinders, and L. Wessels. Random subspace method for multivariate feature selection. *Pattern Recognition Letters*, 27(10):1067-1076, 2006.

T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff. Embedded methods. In I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh (editors) *Feature Extraction: Foundations and Applications*, Springer, Berlin, 137-165, 2006.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81-106, 1986.

A. Rakotomamonjy. Variable selection using SVM-based criteria. *Journal of Machine Learning Research*, 3:1357-1370, 2003.

I. Rivals and L. Personnaz. MLPs (mono-layer polynomials and multi-layer perceptrons) for nonlinear modeling. *Journal of Machine Learning Research*, 3:1383-1398, 2003.

H. Stoppiglia, G. Dreyfus, R. Dubois, Y. Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399-1414, 2003.

K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415-1438.

E. Tuv, A. Borisov, G. Runger, K. Torkkola. Feature selection with ensembles, artificial variables, and redundancy elimination. *Journal of Machine Learning Research*, 10:1341-1366, 2009.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.

S. Weisberg. *Applied Linear Regression*. Wiley, New York, 1980.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, Series B, 67:301-320, 2005.

J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar. Streamwise feature selection. *Journal of Machine Learning Research*, 7:1861-1885, 2006.