Technical Report

Feature Selection for High-Dimensional Data with RapidMiner

Sangkyun Lee, Benjamin Schowe, and Viswanath Sivakumar

01/2011

technische universität dortmund

SFB 876 Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung

Part of the work on this technical report has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project C1.

Speaker: Prof. Dr. Katharina Morik
Address: TU Dortmund University
Joseph-von-Fraunhofer-Str. 23
D-44227 Dortmund
Web: http://sfb876.tu-dortmund.de

Abstract

Feature selection is an important task in machine learning, reducing dimensionality of learning problems by selecting few relevant features without losing too much information. Focusing on smaller sets of features, we can learn simpler models from data that are easier to understand and to apply. In fact, simpler models are more robust to input noise and outliers, often leading to better prediction performance than the models trained in higher dimensions with all features. We implement several feature selection algorithms in an extension of RapidMiner, that scale well with the number of features compared to the existing feature selection operators in RapidMiner.

Contents

1 Introduction

Feature selection is a task of identifying relevant subsets of features for making accurate prediction. The number of features translates to the dimensionality of data, and high dimensionality makes data mining challenging in several aspects:

- The higher the dimension is, the more complicated models are typically required to fit the data, that are harder to comprehend for human eyes.
- A larger number of samples is required to produce statistically stable learning models in higher dimension. For instance, a binary classification task in p dimensional space would require $O(2^p)$ samples to learn a PAC hypothesis without any inductive bias (Mitchell, 1997).
- High dimensionality often entails high variance, leading to unstable learning outcomes. (Saeys et al., 2008; Kalousis et al., 2007).
- More computation is required to deal with larger dimensions.

Feature selection provides effective ways to discover relevant features for many learning tasks. Using only the relevant features, we can perform data mining in reduced spaces, thereby producing more stable learning models (which often leads to more accurate prediction) in shorter time. Such models are also easier to understand and to apply.

Types of Feature Selection Methods There are many feature selection algorithms with numerous ways to measure relevance and redundancy of features and with different computational requirements. We broadly categorize them into three types:

- **Filter Methods**: Filters select features by ranking them according to certain scoring schemes. They are also known as the *variable ranking* methods, which are simple and scale well with dimensions and the number of samples. There exist two types of filters, *univariate* and *multivariate*. Univariate filters treat each feature individually, whereas multivariate filters take care of interactions among features. *Examples*: t-test and correlation-based filters (univariate); MRMR (multivariate) (Ding and Peng, 2003).
- **Wrapper Methods**: A wrapper assesses subsets of features against a certain usefulness criterion using a given predictor. Subset selection is performed separately from training, and thus any off-the-shelf machine learning algorithm can be used as a predictor. Exact subset search is known to be NP-hard, but a wide range of subset search strategies can be adopted, including best-first, branch-and-bound, simulated annealing, genetic algorithms (for a review, see Kohavi and John (1997)). Among these, greedy search strategies are computationally advantageous and robust against overfitting, which comes in two flavors: *forward selection* and *backward elimination*. Forward selection methods incorporate features progressively into growing subsets, whereas backward elimination methods start from all features and progressively eliminate the least promising ones (Guyon and Elisseeff, 2003). *Examples*: genetic algorithms (Mierswa and Wurst, 2006) and SVM-RFE (Guyon et al., 2002).

- **Embedded Methods**: Embedded methods are similar to the wrapper approaches, finding subsets of features by optimizing certain goodness criteria. However, embedded methods perform feature selection as a part of training, not separately to it as in wrappers. Therefore we can make better use of training data, since no separate validation set is needed to evaluate subsets; also, training can be done much faster since we can avoid training a predictor again from scratch for every subset. *Examples*: LASSO (Tibshirani, 1996) and ℓ_1-regularized logistic regression (Ng, 2004).

We note that there are two different goals in feature selection: to achieve concise representation of the data (*unsupervised* feature selection), or to make efficient predictions (*supervised* feature selection). Clustering and matrix factorization algorithms can be used for unsupervised feature selection, which tends to be more robust to overfitting than the supervised counterpart. In this paper we focus on supervised feature selection. Please refer to Guyon and Elisseeff (2003) for more discussion on unsupervised cases.

Redundancy of Features: Somewhat contrary to our intuition, it is not trivial to determine "redundancy" of features. We present some examples that are illustrated in Guyon and Elisseeff (2003):

- Independent and identically distributed features are not always redundant.
- Perfect correlation between features means that they are truly redundant, since no additional information is gained by adding the other. However, when the correlation is not perfect, even very high correlation (or anti-correlation) does not always mean that the features are redundant.

Also, when we have many features, it is very tempting to reduce their number by first applying a filter method before considering more complicated approaches. However, one could potentially lose some important features in that way, since:

- A feature that is completely useless by itself can provide significant performance gain when taken with others.
- Two features that are useless by themselves can be useful together.

In this paper we describe the feature selection methods that we implement in an extension of RapidMiner, called the Feature Selection Extension. We implement feature selection algorithms that are preferable for the cases where the number of features is large. For this reason we exclude the wrapper approaches from our consideration, except for the SVM-REF (Guyon et al., 2002) because of its popularity in bioinformatics.

This paper is built up as follows. In Section 2, 3, 4, we discuss the filter, the wrapper, and the embedded feature selection approaches in turn. The idea of obtaining stable feature sets is presented in Section 5, which is especially important when we have only small samples; we provide an ensemble method to select stable feature subsets. Some utility functions are introduced in Section 6, which can help speed up performing feature selection. Finally we present some numerical experiments illustrating the benefits of our software in Section 7.

Terminology and Notations The terms features, attributes, and variables are regarded to have the same meaning. We use the symbols $\mathbf{x}_k \in \mathbb{R}^p$ and $y_k \in \mathbb{R}$ to denote an input feature vector and its label respectively, for the k-th instance in a data set, $k = 1, 2, \ldots, n$. We use $\mathbf{x}_k(i) \in \mathbb{R}$ to denote the i-th feature of the input vector $\mathbf{x}_k$ for $i = 1, 2, \ldots, p$, and $\mathbf{x}(i) \in \mathbb{R}^n$ to represent the i-th feature vector of the n examples. For a finite set $\mathcal{C}$, we denote by $|\mathcal{C}|$ the cardinality of the set C. When the samples are from two categories, we denote the sets of sample indices belong to each category by $\mathcal{P}$ and $\mathcal{N}$. Finally we show the names of software objects/operators in SMALL CAPITALS.

2 Filter Feature Selection Methods

We present the filter feature selection methods that we implement in our software.

2.1 Univariate Filters

Univariate filters rank features according to certain scoring schemes treating each feature individually, and thus simple and fast, although selected features may not produce the most accurate prediction. Scoring functions have to be chosen depending on data types, i.e., whether the features and the label have numerical (continuous) or nominal (discrete) values. Also, scoring functions may produce statistics in different scales, therefore it is desirable to normalize features when a data set contains both nominal and numerical features (Ding and Peng, 2003; Hall, 2000).

In the following, we describe the scoring functions in such ways to measure the *relevance* of a feature with respect to the class label. Note that the same definitions can be used to measure *redundancy* between two features, by replacing the label with another feature.

2.1.1 Pearson's Correlation

When both the feature and the label are numerical, we measure the linear dependency of them by Pearson's correlation coefficient, which can be estimated by

$$R(i) = \frac{\sum_{k=1}^{n} [\mathbf{x}_k(i) - \bar{\mathbf{x}}(i)][y_k - \bar{y}]}{\sqrt{\sum_{k=1}^{n} [\mathbf{x}_k(i) - \bar{\mathbf{x}}(i)]^2} \sqrt{\sum_{k=1}^{n} [y_k - \bar{y}]^2}}. \quad (1)$$

Here $\bar{\mathbf{x}}(i) := \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k(i)$ and $\bar{y} := \frac{1}{n}\sum_{k=1}^{n} y_k$. The value of $R(i)^2$ represents the fraction of the total variation around the mean $\bar{y}$ that is explained by the linear relation between the i-th feature and the labels (Guyon and Elisseeff, 2003). The features with high $R(i)^2$ values are chosen as relevant features.

2.1.2 F-Statistic

When the feature is numerical but the label has one of C different nominal values, we can compute the F-statistic of them as follows:

$$F(i) := \frac{\sum_{c=1}^{C} |\mathcal{G}_c|(\bar{\mathbf{x}}_c(i) - \bar{\mathbf{x}}(i))^2/(C-1)}{\sum_{c=1}^{C} \sum_{k \in \mathcal{G}_c} (\mathbf{x}_k(i) - \bar{\mathbf{x}}_c(i))^2/(n-C)}, \quad (2)$$

where $\mathcal{G}_c$ is the partition of sample indices $\{1, 2, \ldots, n\}$ that belongs to the group indexed by c, and $\bar{\mathbf{x}}_c(i) := \frac{1}{|\mathcal{G}_c|}\sum_{k \in \mathcal{G}_c} \mathbf{x}_k(i)$. This statistic represents the ratio of the variance

between groups and the average variance within the groups. Higher values imply larger relevance.

2.1.3 Mutual Information

If both the feature and the label are nominal-valued, we use the mutual information to measure the shared information between two random variables (a feature and the label):

$$\mathrm{MI}(i) = \sum_{\mathbf{x}(i)} \sum_{y} \mathbb{P}(X = \mathbf{x}(i), Y = y) \log \frac{\mathbb{P}(X = \mathbf{x}(i), Y = y)}{\mathbb{P}(X = \mathbf{x}(i))\mathbb{P}(y)},$$

where $\mathbf{x}(i)$ and y represent the realizations of the i-th feature and the label in data, respectively. Higher values imply larger relevance.

Operator: Pearson's correlation, F-statistic, and mutual information scoring functions are implemented in the operator Weight by Maximum Relevance. It creates the scores of given features, choosing a suitable scoring function for the type of the data.

2.1.4 Welch's t-Test

Welch's t-test is a generalization of Student's t-test for the cases when the variance of two sample populations are not equal (Sawilowsky, 2002). The t-statistic of the i-th feature for testing the difference of two sample means (corresponding to the two classes denoted by $\mathcal{P}$ and $\mathcal{N}$) in Welch's test is defined by

$$t(i) := \frac{\bar{\mathbf{x}}_{\mathcal{P}}(i) - \bar{\mathbf{x}}_{\mathcal{N}}(i)}{\sqrt{\frac{1}{|\mathcal{P}|} \sum_{k \in \mathcal{P}} [\mathbf{x}_k(i) - \bar{\mathbf{x}}_{\mathcal{P}}(i)]^2 + \frac{1}{|\mathcal{N}|} \sum_{k \in \mathcal{N}} [\mathbf{x}_k(i) - \bar{\mathbf{x}}_{\mathcal{N}}(i)]^2}}, \tag{3}$$

where $\bar{\mathbf{x}}_{\mathcal{P}}(i) := \frac{1}{|\mathcal{P}|} \sum_{k \in \mathcal{P}} \mathbf{x}_k(i)$ and $\bar{\mathbf{x}}_{\mathcal{N}}(i) := \frac{1}{|\mathcal{N}|} \sum_{k \in \mathcal{N}} \mathbf{x}_k(i)$.

Operator: The Weight by Welch-test operator computes the p-value of each feature using two-sided, two-sample Welch's t-test. (This operator is implemented by Miriam Bützken.) Features with smaller p-values are preferred for selection. The degree of freedom values are estimated from data.

2.1.5 Significance Analysis for Microarrays (SAM)

For high dimensional microarray data in bioinformatics, Tusher et al. (2001) suggested the Significance Analysis for Microarrays (SAM) to identify genes with significant changes in their expression, assimilating a set of gene-specific t-tests. To measure gene-specific fluctuations, SAM defines *relative difference* measure $d(i)$ for the i-th gene as follows:

$$d(i) := \frac{\bar{\mathbf{x}}_{\mathcal{P}}(i) - \bar{\mathbf{x}}_{\mathcal{N}}(i)}{s_i + s_0},$$

where $\bar{\mathbf{x}}_{\mathcal{P}}(i)$ and $\bar{\mathbf{x}}_{\mathcal{N}}(i)$ are the average levels of expression of gene i corresponding to the groups $\mathcal{P}$ and $\mathcal{N}$, respectively. The s_i in the denominator represents the *gene-specific scatter* which is defined by

$$s_i := \sqrt{\frac{|\mathcal{P}| + |\mathcal{N}|}{|\mathcal{P}||\mathcal{N}|(|\mathcal{P}| + |\mathcal{N}| - 2)} \left(\sum_{k \in \mathcal{P}} [\mathbf{x}_k(i) - \bar{\mathbf{x}}_{\mathcal{P}}(i)]^2 + \sum_{k \in \mathcal{N}} [\mathbf{x}_k(i) - \bar{\mathbf{x}}_{\mathcal{N}}(i)]^2 \right)}.$$

The parameter s_0 is chosen to make the variance of $d(i)$ independent of gene expression.

Operator: The SAM is implemented in the WEIGHT BY SAM operator, which returns the magnitude of the relative difference values for genes.

2.2 Multivariate Filters

Multivariate filters take the interaction among features into account, in order to overcome the restriction of univariate filters that only consider individual effects of features. We present several methods in chronological order.

2.2.1 Prediction Analysis for Microarrays (PAM)

The Prediction Analysis for Microarrays (PAM) (Tibshirani et al., 2002) is a method of shrunken centroids, performing feature selection and classification using the nearest centroids.

Suppose that $\mathcal{G}_c \subset \{1, 2, \ldots, n\}$ denotes the sample indices belong to the class c, for $c = 1, 2, \ldots, C$. Then the centroid for class c is defined by $\bar{\mathbf{x}}_c := \frac{1}{|\mathcal{G}_c|} \sum_{k \in \mathcal{G}_c} \mathbf{x}_k$, which is the mean expression vector of class c. The overall centroid is defined by $\bar{\mathbf{x}} := \frac{1}{n} \sum_{k=1}^{n} \mathbf{x}_k$. PAM shrinks the class centroids toward the overall centroid, after standardizing each gene by within-class standard deviation, to give higher weight to genes with stable expression within the samples of the same class. This is done by computing the standardized distance between the centroid of class c to the overall centroid for gene i:

$$d_c(i) := \frac{\bar{\mathbf{x}}_c(i) - \bar{\mathbf{x}}(i)}{m_c(s_i + s_0)}$$

where s_i is the pooled within-class standard deviation for gene i:

$$s_i^2 := \frac{1}{n - |\mathcal{G}_c|} \sum_{c=1}^{C} \sum_{k \in \mathcal{G}_c} (\mathbf{x}_k(i) - \bar{\mathbf{x}}_c(i))^2, \quad m_c := \sqrt{1/|\mathcal{G}_c| + 1/n}.$$

The value of s_0 is set to the median value of the s_i values over genes, which is introduced to avoid large $d_c(i)$ values arising by chance from genes with low expression levels. The expression above can be rewritten as

$$\bar{\mathbf{x}}_c(i) = \bar{\mathbf{x}}(i) + m_c(s_i + s_0)d_c(i).$$

PAM shrinks each $d_c(i)$ toward zero via soft thresholding, i.e., $d_c(i)' := \text{sign}(d_c(i)) \max\{|d_c(i)| - \Delta, 0\}$, producing shrunken centroids:

$$\bar{\mathbf{x}}_c(i)' = \bar{\mathbf{x}}(i) + m_c(s_i + s_0)d_c(i)'.$$

The shrinkage parameter $\Delta > 0$ is provided by users. For gene i, if $d_c(i)$ is shrunken to zero for all classes $c = 1, 2, \ldots, C$, then the gene is considered to be removed, since the centroid for gene i becomes the overall centroid $\bar{\mathbf{x}}(i)$, the same for all classes, and no longer contributes to the nearest-centroid computation.

Test examples are classified using the nearest shrunken centroid. For an example $\mathbf{x}$, the decision function is defined for class c,

$$\delta_c(\mathbf{x}) := \sum_{i=1}^{p} \frac{\mathbf{x}(i) - \bar{\mathbf{x}}_c(i)'}{(s_i + s_0)^2} - 2 \log \pi_c,$$

where the second term is a correction based on the class prior probability π_c, which can be estimated by $\hat{\pi}_c = |\mathcal{G}_c|/n$. The class c that gives the smallest $\delta_c(\mathbf{x})$ becomes the prediction outcome.

Operator: PAM is implemented in the SHRUNKEN CENTROIDS / PAM - PREDICTION ANALYSIS FOR MICROARRAYS operator. The operator requires numerical inputs, and outputs the original EXAMPLESET, an ATTRIBUTEWEIGHTS object, a PREDICTION-MODEL and the class weights. The weight of an attribute contains the number of classes for which the attribute is relevant, i.e., the number of classes for which the class centroid of the attribute does not match the overall centroid. The class weight contains the distance $d_c(i)$ values of i-th feature for each class c.

2.2.2 Correlation-Based Feature Selection (CFS) and Minimum Redundancy Maximum Relevance (MRMR)

The Correlation-Based Feature Selection (CFS) (Hall, 2000) and the Minimum Redundancy Maximum Relevance (MRMR) (Ding and Peng, 2003) methods perform a sequential forward search, evaluating features with a correlation based or an information theoretic measure, respectively. They iteratively augment the set of chosen features $\mathcal{S}$, adding the best feature according to a quality criterion Q in each iteration:

$$\mathcal{S}_\ell = \mathcal{S}_{\ell-1} \cup \left\{ \underset{i \in \{1,2,\ldots,p\} \setminus \mathcal{S}_{\ell-1}}{\arg\max} \; Q(i) \right\}, \quad \ell \geq 1,$$

with $\mathcal{S}_0 = \emptyset$, where $Q(i)$ is either the difference

$$Q_{\text{MID}}(i) = Relevance(i) - \frac{1}{\ell} \sum_{j \in \mathcal{S}_\ell} Redundancy(i, j)$$

or the ratio between relevance and average pairwise redundancy of a feature:

$$Q_{\text{MIQ}}(i) = \frac{Relevance(i)}{\frac{1}{\ell} \sum_{j \in \mathcal{S}_\ell} Redundancy(i, j)}.$$

The $Relevance(\cdot)$ and $Redundancy(\cdot,\cdot)$ functions automatically map to one of the scoring functions (1), (2) or (3), depending on the types of provided data (nominal/numerical).

Operator: The operator SELECT BY MRMR / CFS implements these algorithms. Additionally, the operator has an option to produce stabilized selection results via a fast ensemble technique discussed later in Section 5, which bootstraps the selection process for e times to decrease the variance of the results. The operator returns the original EXAMPLESET and an ATTRIBUTEWEIGHTS object where the weights of selected features are set to one, and the rest are set to the zero value.

Remarks: The Q_{MID} and Q_{MIQ} functions are used in other operators as well. For instance, the operators PERFORMANCE (MRMR) and PERFORMANCE (CFS) use the two functions to evaluate feature subsets, possibly in a OPTIMIZE SELECTION loop. The PERFORMANCE (MRMR) operator also provides relevance and redundancy information as separate outputs, so that users can perform multi-objective optimization. To avoid multiple evaluations for the same features, users can create an MRMR-CACHE object using the MRMR CACHE CREATOR.

2.2.3 Fast Correlation-Based Filter (FCBF)

The Fast Correlation-Based Filter (FCBF) (Lei Yu, 2004) consists of two steps, one for choosing relevant features, and the other for removing redundant ones from the subset selected in the previous step.

To evaluate the relevance and the redundancy of features, FCBF uses the *symmetrical uncertainty* (SU) measure, which is the information gain of a random variable X provided by another random variable Y, normalized by the summation of their entropy values, i.e.,

$$SU(X,Y) := 2\frac{H(X) - H(X|Y)}{H(X) + H(Y)}.$$

The SU value of one indicates perfect correlation between features, whereas the value of zero represents independence of them. We can define the relevance of the i-th feature with respect to the class c by $SU(i,c)$, and the redundancy between two features indexed by i and j by $SU(i,j)$. FCBF first choose all features that have relevance values higher than a pre-defined threshold (between 0 and 1). Then, among the selected features, FCBF removes redundant ones that have *approximate Markov blankets* in the remaining features. For two relevant features indexed by i and j, the j-th feature is defined to form an approximate Markov blanket for the i-th feature if and only if $SU(j,c) \geq SU(i,c)$ and $SU(i,j) \geq SU(i,c)$.

Operator: This approach is implemented in FCBF - FAST CORRELATION BASED FILTER operator. The operator takes nominal inputs, and outputs the original input EXAMPLESET, an EXAMPLESET with only the selected attributes, and an ATTRIBUTEWEIGHTS object. (Users can discretize their inputs using the DISCRETIZE BY

ENTROPY operator in RAPIDMINER if necessary.) A threshold parameter is used to determine the relevance of features.

2.2.4 Backward Elimination via Hilbert-Schmidt Independence Criterion (BAHSIC)

The Backward Elimination via Hilbert-Schmidt Independence Criterion (BAHSIC) (Song et al., 2007a,b) considers feature selection using a relevance statistic defined in the Hilbert space, which can be estimated efficiently with a small number of samples.

Let us consider two domains $\mathcal{X}$ and $\mathcal{Y}$ where we draw samples and labels respectively. Given feature mappings $\phi : \mathcal{X} \to \mathcal{H}$ and $\psi : \mathcal{Y} \to \mathcal{H}'$, we define a cross-covariance operator $\mathcal{C}_{\mathbf{xy}} : \mathcal{H}' \to \mathcal{H}$ between feature maps, that is,

$$\mathcal{C}_{\mathbf{xy}} = \mathbb{E}_{\mathbf{xy}}\{(\phi(\mathbf{x}) - \mathbb{E}_{\mathbf{x}}[\phi(\mathbf{x})]) \otimes (\psi(\mathbf{y}) - \mathbb{E}_{\mathbf{y}}[\psi(\mathbf{y})])\}$$

where $\otimes$ is the tensor product. The square of the Hilbert-Schmidt norm of the cross-covariance operator (HSIC), $\|\mathcal{C}_{\mathbf{xy}}\|^2_{HS}$, is then used to evaluate the relevance of a feature $\mathbf{x}$ to the label $\mathbf{y}$. Given samples $Z = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, an unbiased estimate of HSIC can be computed by

$$\mathrm{HSIC}(\mathcal{H}, \mathcal{H}', Z) = \frac{1}{n(n-3)}\left[\mathrm{tr}(KL) + \frac{\mathbf{1}^T K \mathbf{1}\mathbf{1}^T L \mathbf{1}}{(n-1)(n-2)} - \frac{2}{n-2}\mathbf{1}^T KL\mathbf{1}\right],$$

where K and L are kernel matrices with zero diagonals, computed by $K_{ij} = (1 - \delta_{ij})\langle\phi(\mathbf{x}_i), \phi(\mathbf{y}_j)\rangle$ and $L_{ij} = (1 - \delta_{ij})\langle\psi(y_i), \psi(y_j)\rangle$ ($\delta_{ij} = 1$ if $i = j$, and zero otherwise). When all kernel entries are bounded by one almost everywhere, we can show that the gap between the estimate and the true value is bounded by $8\sqrt{\log(2/\delta)/n}$ with probability at least $1 - \delta$, for $n \geq 1$ and $\delta > 0$.

Feature selection is performed by backward elimination, starting with all features and removing features progressively. In each iteration, the algorithm removes features of a pre-specified fraction such that if they are removed, the HSIC evaluated on the remaining features will be maximized. The algorithm repeats this process until there will be no features left, adding the removed features to a list in order. The most recently added feature in this list provide the most relevant ones. Optionally, the parameters for the input/output kernels can be optimized in each elimination step by grid search, to maximize the HSIC score on the current set of remaining features.

Operator: The operator BACKWARD ELIMINATION VIA HILBERT-SCHMIDT INDEPENDENCE CRITERION implements this algorithm. The operator requires numerical input (the NOMINAL TO NUMERICAL in RAPIDMINER can be used if required, making sure that 'include special attributes' is checked to include labels). The operator outputs the original input EXAMPLESET, an EXAMPLESET with only the selected attributes, and an ATTRIBUTEWEIGHTS object, which contains the ranks of features according to their relevance if the ranks are higher than a specified value; otherwise the weights are set to zero.

Remarks: The operator supports liner, radial, polynomial, neural, anova, epachnenikov, Gaussian combination and multi-quadratic kernels. It also supports multi-label ExampleSets. The parameters `kernelx type` and `kernely type` denote the kernels to be used for the features and the labels respectively. Depending on the types of kernels, users can specify kernel parameter values or let the parameters be optimized by grid search within a specified range.

2.2.5 Dense Relevant Attribute Group Selector (DRAGS)

The Dense Relevant Attribute Group Selector (DRAGS) (Yu et al., 2008) finds all relevant features without removing highly correlated ones. This is done by identifying dense feature regions using the kernel density estimation (known as the Parzen window), selecting the dense regions that are relevant for classification. This helps improve the stability of feature selection in terms of input and dimension sampling.

To identify the dense feature regions, for given samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^p$, we consider the corresponding feature vectors $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_p$ in the n-dimensional sample space. We search for the modes (peaks) of the kernel density estimators given by

$$\hat{p}(\mathbf{f}) = \frac{1}{ph^n} \sum_{i=1}^{p} K\left(\frac{\mathbf{f} - \mathbf{f}_i}{h}\right),$$

where h is a fixed bandwidth. The modes of $\hat{p}$ correspond to the roots of $\nabla\hat{p}(\mathbf{f}) = 0$, which can be found efficiently by the mean shift procedure (Cheng, 1995) without estimating the density. The procedure produces a sequence estimated peaks $\mathbf{c}_1, \mathbf{c}_2, \ldots$ by

$$\mathbf{c}_{j+1} = \frac{\sum_{i=1}^{p} \mathbf{f}_i K\left(\frac{\mathbf{c}_j - \mathbf{f}_i}{h}\right)}{\sum_{i=1}^{p} K\left(\frac{\mathbf{c}_j - \mathbf{f}_i}{h}\right)}, \quad j = 1, 2, \ldots.$$

This sequence converges to a limit point if the kernel K satisfies mild conditions. (Examples of such kernels include a flat kernel $K(\mathbf{f})$ that returns 1 if $\|\mathbf{f}\| \leq \lambda$ or 0 otherwise for a given $\lambda > 0$, and the Gaussian kernel.) DRAGS computes all peaks starting from each feature vector, merging two peaks if their distance is closer than h.

After identifying all unique peaks representing dense regions, DRAGS clusters each feature vector to a peak that is closer than h in distance. (After this step, the groups with low density can be discarded optionally.) Then the groups are ranked by the average correlation of the features in each group to the class label. One representative feature is chosen from each group, which has the maximal correlation to the class label.

Operator: This selection method is implemented in the Dense Relevant Attribute Group Selector operator. It requires numerical inputs, which are normalized inside of the operator. (The Nominal to Numerical operator in RapidMiner can be used if necessary). The operator outputs the original ExampleSet, an ExampleSet with only the selected attributes and an AttributeWeights object. The weights of the most relevant features in each group contains their ranks, while the weights of the rest

are set to zero. The operator also returns a set of indicator weight vectors that represents the clustering of attributes into dense groups: in each weight vector, a weight is set to one (or two, if it is the most relevant in the group) if the corresponding attribute is clustered to the group associated with the vector; otherwise it is set to the zero value.

Remarks: The parameter `kernel type` determines the types of kernels to be used: the flat and the Gaussian kernel are supported. The parameter `eps` is used to declare the convergence of the mean shift procedure.

2.2.6 Consensus Group Stable Feature Selector (CGS)

The Consensus Group Stable Feature Selection (CGS) (Loscalzo et al., 2009) is an extension of the DRAGS algorithm in Section 2.2.5, based on the same idea of identifying dense feature groups. However, CGS is designed to overcome two major limitations of DRAGS: the fact that density estimation of features can be unreliable due to the shortage of samples, where a large enough number is required to observe feature correlation, and the fact that some relevant features can be ignored if they reside in relatively sparse feature groups.

To form consensus groups, CGS first identifies all dense feature groups from bootstrapped training samples, creating the similarity matrix of features W as follows:

$$W_{ij} := \frac{\text{the number of times the features } i \text{ and } j \text{ are grouped together}}{\text{the number of bootstrapping trials}}.$$

Then an agglomerative hierarchical clustering is performed on W to find consensus feature groups. Average linkage is used when merging clusters, to reduce the effect of outliers. Merging continues until there is no feature groups with an average similarity value larger than 0.5.

Feature selection is performed by choosing a representative feature from each consensus group which is closest to the group center. Then the relevance of the representatives are computed in terms of the correlation to the class label.

Operator: The Consensus Group Stable Feature Selector operator provides the CGS algorithm. It requires numerical inputs, and outputs the original ExampleSet, an ExampleSet with only the selected attributes, and an AttributeWeights object. The weights of the top relevant attributes contain their ranks, while the rest are set zero. The operator also returns the clustering of attributes into dense groups in the same way as DRAGS.

Remarks: In addition to the parameters of DRAGS, the CGS operator has the parameter `number of subsampling`, denoting the number of bootstrapping trials, and the `sample ratio` parameter which determines the number of training samples in each bootstrapping trial.

3 Wrapper Feature Selection Methods

In this section we discuss one wrapper approach, the SVM-REF (Guyon et al., 2002). Other traditional wrapper approaches can be implemented using the existing operators in RapidMiner, but they do not scale well for high dimensions due to their intensive computational requirements.

3.1 Recursive Feature Elimination using SVM (SVM-REF)

The Support Vector Machines (SVMs) (Vapnik, 1998) find decision functions with large margins by solving the following minimization problem. The solutions of the minimization are typically very dense, since the ℓ_2-norm in the objective tends to distribute weights over all dimensions.

$$\underset{(\beta,\beta_0)\in\mathbb{R}^{p+1},\xi\geq 0}{\arg\min} \frac{1}{2}||\beta||_2^2 + C\sum_{i=1}^{n}\xi_i \quad \text{s.t. } y_i(\langle\beta,\mathbf{x}_i\rangle+\beta_0)\geq 1-\xi_i,\ i=1,2,\ldots,n.$$

Therefore, it is hard to find important attributes by simply discarding the components with small magnitude in a solution. Instead, we can use a refined strategy called the *Recursive Feature Elimination* (SVM-RFE) (Guyon et al., 2002). SVM-RFE works in an iterative fashion, starting with the index set of all features $\mathcal{S}=\{1,2,\ldots,p\}$:

1. A linear SVM is trained on the features with indices in $\mathcal{S}$, resulting in β.
2. A fraction or a fixed number of features $j\in\mathcal{S}$ with small $|\beta_j|$ is removed from $\mathcal{S}$.
3. If $|\mathcal{S}|\leq k$ for a threshold value k, stop. Otherwise repeat from the step 1.

Note that the SVM can be replaced with other learning algorithms in this scheme.

Operator: We implemented two RFE operators in RapidMiner. The operator Select by Recursive Feature Elimination with SVM is an implementation of SVM-RFE, using the linear SVM code jMySVM, with a fixed parameter C for all iterations. To specify a different C value for each SVM round, or to use alternative learning algorithms, one can use the operator Recursive Feature Elimination. It contains a subprocess which can be filled with a chain of operators producing AttributeWeights objects.

4 Embedded Feature Selection Methods

Embedded feature selection methods make use of linear decision models such as $f(\mathbf{x}) = \langle \mathbf{x}, \beta \rangle + \beta_0$, where $\beta \in \mathbb{R}^p$ and $\beta_0 \in \mathbb{R}$ are the coefficients of a model. Training such models, we obtain the coefficients that capture the importance of corresponding variables by their magnitude, while the number of nonzero coefficients is controlled by sparsity-inducing norms such as the ℓ_1 norm.

4.1 Least Angle Regression (LASSO and LARS)

The *least absolute selection and shrinkage operator* (LASSO) (Tibshirani, 1996) produces sparse coefficient vectors β using the following ℓ_1 regularization problem,

$$\min_{\beta \in \mathbb{R}^p, \beta_0} \frac{1}{n} \sum_{k=1}^{n} (y_k - \mathbf{x}_k^T \beta - \beta_0)^2, \quad \|\beta\|_1 \leq t.$$

Note that if we omit the offset term β_0, then we have to standardized the input vectors.

The *least angle regression* (LARS) (Efron et al., 2004) algorithm provides stepwise regression models, as well as the solutions of LASSO with some modifications. Starting with $\beta = \mathbf{0}$, each iteration of LARS increases the coefficients whose corresponding features have the highest correlation with the target, until all coefficients have non-zero values.

Figure 1 shows the changes of solution coefficients for LASSO and LARS, for the Diabetes data set (`http://www.stanford.edu/~hastie/Papers/LARS/`).

Operator: The LARS - Least Angle Regression operator implements both LASSO and LARS algorithms. Once a model is trained for a given threshold value, we can extract the weights of features that correspond to another threshold value, or the nonzero weights whose number is no more than a specified value, by using the LARS - Change Model Parameters operator.

5 Stable Feature Selection Methods

An important question in feature selection is how to obtain feature subsets that are robust to sample variation (Saeys et al., 2008; Kuncheva, 2007; Meinshausen and Bühlmann, 2010). We discuss the stability of feature selection, introducing an ensemble approach we implement for RapidMiner to provide robust feature sets.

5.1 Stability Measures

The stability of a feature selection method can be measured by the similarity of feature subsets chosen by using different samples. We introduce two of such measures, the *Jaccard index* and *Kuncheva's index*.

Jaccard Index: The Jaccard index (Saeys et al., 2008) of two feature subsets F_a and F_b defined as follow:

$$S_J(F_a, F_b) = \frac{|F_a \cap F_b|}{|F_a \cup F_b|}.$$

The index is one if the two sets are identical, and the zero value if there is no feature shared by the two sets.

Kuncheva's Index: Kuncheva's index (Kuncheva, 2007) is defined for two subsets F_a and F_b of the same size k, taking into account the number of entire features p:

$$S_K(F_a, F_b) = \frac{|F_a \cap F_b| - \frac{k^2}{p}}{k - \frac{k^2}{p}} \tag{4}$$

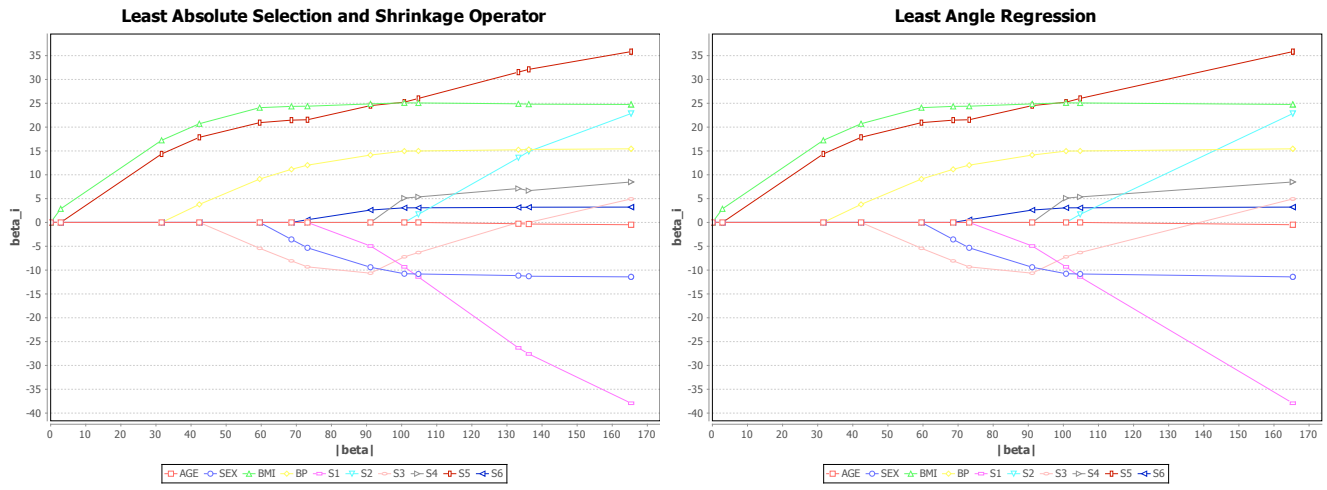


Figure 1: The coefficients in the solutions of LASSO and LARS on the Diabetes data set.

This index is one if the two sets are identical, and some negative value larger than -1 if there is no index shared by the two sets.

In both measures, the stability of more than two feature sets is computed by the average of all pairwise stability indices.

Operator: The stability of a feature selection method can be measured the FEATURE SELECTION STABILITY VALIDATION operator. The operator applies the specified selection method repeatedly on input samples (created by bootstrapping or cross-validation) of the original EXAMPLESET, and computes the stability values of resulting feature subsets. It can also compare the correlation of the associated coefficient vectors.

5.2 Ensemble Methods

Ensemble methods, in general, make use of multiple instance of learning methods to obtain better collective prediction than what can be expected from individual ones. We can extend this idea for feature selection, to improve the stability of feature selection as well as the predictive power of the selected features (Saeys et al., 2008; Kalousis et al., 2007; Meinshausen and Bühlmann, 2010). For RAPIDMINER we implement the approach developed in SFB 876, by Schowe and Morik (2010). This method runs a specified feature selection algorithm over bootstrap samples of input points, producing a consensus set of features combining different feature via ranking, weight thresholding, or simple summation.

Operator: The ENSEMBLE FEATURE SELECTION provides a meta-operator that can be filled with any feature selection method. The specified method is then applied repeatedly to bootstrap samples, similarly to the FEATURE SELECTION STABILITY VALIDATION operator.

Remarks: The ATTRIBUTEWEIGHTS object of each feature selection run is combined to a consensus ATTRIBUTE WEIGHTS object in three ways. (i) The `top-k` method counts how many times a feature has been selected in the top k features of each run. Then the k features with the highest count are returned. Users can also specify the minimum count required for each feature to be selected. (ii) The `geq-w` method works in similar fashion, counting how many times a feature received a weight greater than or equal to the specified value threshold. (iii) Finally, the `accumulate-weights` option simply adds up the weights over all iterations.

6 Utility Operators

In this section we introduce various utility operators implemented to help feature selection or model building tasks, simplifying the application of lengthy sub-processes or macros.

Select top k features: The SELECT TOP FEATURES operator takes an ATTRIBUTEWEIGHTS object as an input, and selects the top k or the top p percent entries in magnitude of the weight vector. The weight values of chosen entries are set to one, and the others are set to the zero value. This operator can be used inside of the WRAPPER-VALIDATION operator, for instance.

Log performance: The LOOP AND AVERAGE operator in RAPIDMINER allows logging of a single performance measure. To allow multiple performance measures for logging, we implement the MAKE PERFORMANCE LOGGABLE operator which can be attached to any LOOP AND AVERAGE operator. Our operator returns a PERFORMANCEVECTOR object which contains the measurements, along with their counts, mean, variance, and standard deviation values.

Convert Weights To Ranking: The CONVERT WEIGHTS TO RANKING operator sorts the weights of features and replaces the weight values with their ranks. The magnitude of the weights or their signed values can be used for sorting.

Rank by Selection: The RANK BY SELECTION operator extracts the intermediate ranking information of iterative feature selection procedures. The operator repeatedly runs the feature selection method specified by users as a subprocess.

Replace Missing Values: The REPLACE MISSING VALUES (WITH OFFSET) operator is an extension of the REPLACE MISSING VALUES operator in RAPIDMINER, allowing users to specify an offset for the values to be filled in missing entries. This can be used to distinguish the missing entries from the entries with maximum observed values, when we fill the missing entries with the maximum values.

7 Experiments

We present illustrative examples using our feature selection operators.

7.1 Performance Improvement

To demonstrate the benefit of feature selection in terms of prediction performance, we compare three learning models: Random Forest, Naive Bayes (NB), and One Nearest Neighbor (1NN). We compare these methods with and without feature selection, where feature selection is performed by the MRMR algorithm in Section 2.2.2.

Figure 2 shows the result for the colon data set[1] (Alon et al., 1999) ($n = 62, p = 2000$). We use five settings: two simple models with feature selection (NB+MRMR and 1NN+MRMR) and without feature selection (NB and 1NN), and the Random Forest without feature selection (Random Forest). We can easily find two facts: feature selection can improve the performance of learning models, and thus simpler models with feature selection can be used instead of complicated ones without feature selection.

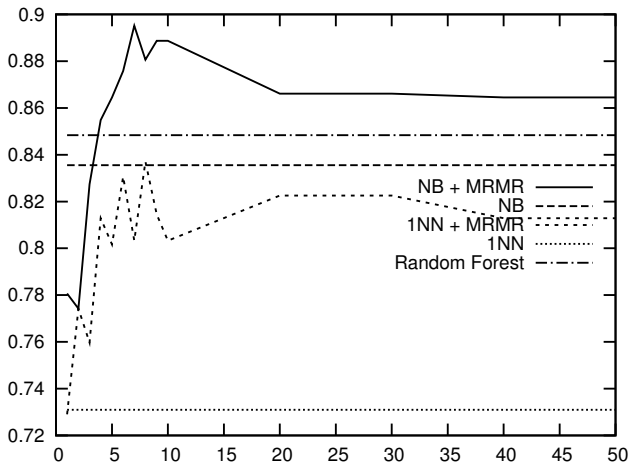


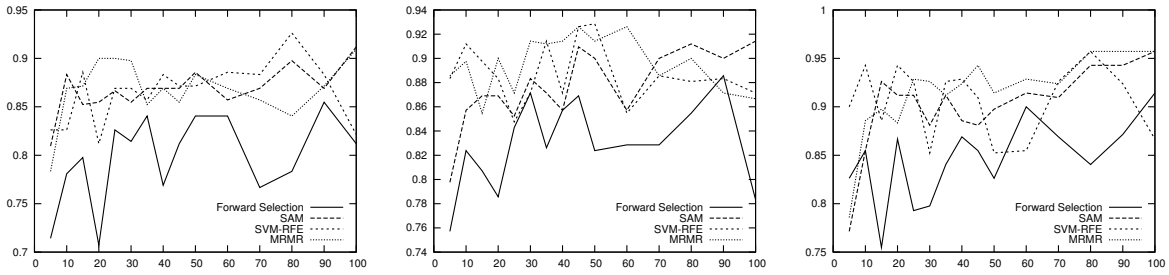
Figure 2: Comparison of learning models with and without feature selection, on the colon data set ($n = 62, p = 2000$). The x-axis represents the number of chosen features, and the y-axis shows the corresponding prediction accuracy values.

7.2 Filter and Wrapper Approaches

Now we show the potential benefits of our feature selection operators over the existing ones in RapidMiner. For comparison, we select features using four different approaches:

- A wrapper implemented using the Forward Selection operator and the Naive Bayes learner in RapidMiner, with ten fold cross validation.

[1] Available at `http://genomics-pubs.princeton.edu/oncology/affydata/index.html`



(a) Naive Bayes (b) 1NN (c) SVM

Figure 3: Classification accuracy (y-axis) of three learners, Naive Bayes, 1NN, and SVM, using the features selected by using four different strategies. The x-axis represents the number of chosen features.

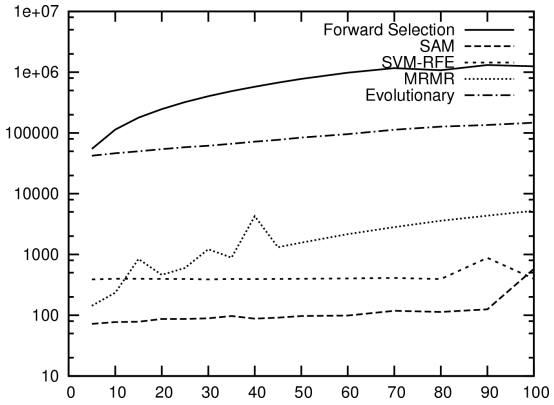


Figure 4: Runtime (y-axis, log-scale) of the different selection methods dependent on the number of selected features (x-axis).

- Three feature selection methods we implement, a univariate filter using the SAM statistic, a multivariate filter (MRMR), and a wrapper (SVM-RFE).

We compare the prediction performance of the four feature sets obtained by the above settings, on our miRNA expression data set with 67 examples and 302 features (not publicly available), where predictions are made by three learning algorithms, the Native Bayes, the One Nearest Neighbor (1NN), and the SVM.

Figure 3 shows the prediction accuracy of the three learning methods using the four feature selection approaches, where Figure 4 reports their runtime in seconds (log scale). The results suggest that our new feature selection methods (SAM, MRMR, and SVM-REF) lead to better prediction performance overall, even though they require much shorter computation time, than the existing methods in RapidMiner. (As shown in Figure 4, the computation time of Forward Selection can be reduced using another search strategy Optimize Selection (Evolutionary) in RapidMiner, but it is still much slower than ours.)

7.3 Benchmark of Multivariate Filter Methods

To compare the multivariate filter approaches presented in Section 2.2, we perform a small benchmark using three public data sets summarized in Table 1. We test the prediction accuracy of the Naive Bayes classifier using the feature sets obtained by PAM, FCBF, BAHSIC, DRAGS, and CGS filter methods. (Note that in the original paper of PAM (Tibshirani et al., 2002) the nearest centroid was used as the classifier, not the Naive Bayes. Their prediction performance were similar, but the Naive Bayes produced smaller feature sets in our experiments.)

The results are summarized in Table 2. The parameters of the filter methods that determine the number of features to be selected are optimized by grid search, using the Optimize Parameters (Grid) operator in RapidMiner, evaluating performance in each grid point by ten fold cross validation using the Wrapper-X-Validation operator in RapidMiner. The stability of feature selection is measured by the Jaccard index introduced in Section 5, for the COLON data set. The number of selected features in each setting is shown in square brackets.

The workflows for the experiments are available at `http://www.myexperiment.org/users/17770/workflows`. The actual RapidMiner processes, results (attribute weights, performance and parameter sets), and the logs of experiments can be downloaded from `http://www.myexperiment.org/files/537.html`.

Table 1: The benchmark data sets for testing multivariate filter approaches.

Name	Examples	Attributes	Classes	Source
COLON	62	2000	2	`http://www.cs.binghamton.edu/~lyu/KDD08/data/colon-std.arff`
SRBCT	63	2308	4	`http://www.cs.binghamton.edu/~lyu/KDD08/data/srbct-std.arff`
SONAR	208	61	2	RapidMiner sample repository

Table 2: The best cross validation prediction accuracy (standard deviation in parentheses) of the Naive Bayes classifier using the features selected by multivariate filter approaches on the benchmark data sets. Stability is measured by the Jaccard index on the COLON data set. The number of selected features is shown in square brackets.

	PAM	FCBF	BAHSIC	DRAGS	CGS	All Features
COLON	85.7 (14.2) [6]	86.9 (10.0) [5]	79.3 (12.3) [100]	84.3 (15.3) [91]	85.5 (13.8) [46]	55.0 (15.3) 2000
SRBCT	98.3 (5.0) [40]	98.3 (5.0) [40]	96.7 (6.7) [100]	87.4 (14.8) [250]	90.2 (16.9) [210]	93.3 (11.1) 2308
SONAR	74.1 (6.2) [3]	65.4 (9.7) [10]	72.7 (11.2) [11]	70.1 (12.6) [5]	70.2 (9.1) [5]	66.9 (7.3) 61
Stability	0.571	0.253	0.536	0.270	0.311	1.0

7.4 Stable Feature Selection

We use the ensemble feature selection method in Section 5 to show its stability profile over different numbers of subsampling. In Figure 5 we compare the stability of two methods in terms of Kuncheva's index, the MRMR and an ensemble of MRMR (using ten-fold cross validation subsampling). We selected features with different sizes in the range of $[0, 50]$ from the colon data set (Alon et al., 1999).

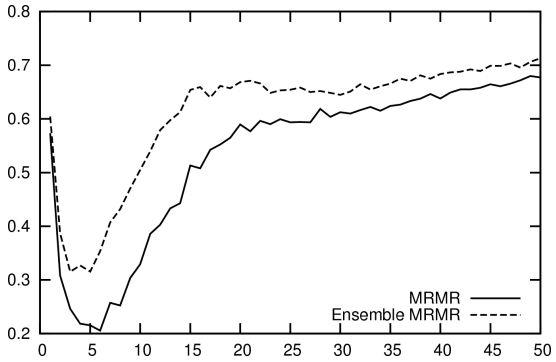


Figure 5: Stability of the MRMR and an ensemble of the MRMR, measured by Kuncheva's index (y-axis), for the different sizes of selected features (x-axis).

8 Conclusion

We presented an extension to RapidMiner which provides feature selection algorithms favorable for high-dimensional data. The operators implementing these algorithms usually performs much faster than the wrapper approaches that can be constructed combining the existing RapidMiner operators. We also provide operators implementing stability measures and an ensemble feature selection algorithm, to provide effective means to obtain robust feature sets.

Acknowledgements

The feature selection extension software (version 1.1.4) for RapidMiner is developed by Benjamin Schowe and Viswanath Sivakumar, and part of this work is based on their reports and the experiment results therein.

References

U. Alon, N. Barkai, D. A. Notterman, K. Gishdagger, S. Ybarradagger, D. Mackdagger, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, June 1999.

Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *Proceedings of the Computational Systems Bioinformatics*, pages 523–528, 2003.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407, 2004.

I. Guyon and 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*, 46:389–422, 2002.

M. A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *ICML*, pages 359–366, 2000.

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

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

L. I. Kuncheva. A stability index for feature selection. In *Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications*, pages 390–395, 2007.

H. L. Lei Yu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, Oct 2004.

S. Loscalzo, L. Yu, and C. Ding. Consensus group based stable feature selection. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 567–576, 2009.

N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4), 2010.

I. Mierswa and M. Wurst. Information preserving multi-objective feature selection for unsupervised learning. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 1545–1552, 2006.

T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

A. Y. Ng. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of 21st International Conference on Machine Learning*, 2004.

Y. Saeys, T. Abeel, and Y. V. de Peer. Robust feature selection using ensemble feature selection techniques. In W. Daelemans, B. Goethals, and K. Morik, editors, *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II*, volume 5212 of *Lecture Notes in Computer Science*, pages 313–325. Springer, 2008. ISBN 978-3-540-87480-5.

S. Sawilowsky. Fermat, schubert, einstein, and behrens-fisher: The probable difference between two means when $\sigma_1 \neq \sigma_2$. *Journal of Modern Applied Statistical Methods*, 1 (2):461 – 472, 2002.

B. Schowe and K. Morik. Fast-ensembles of minimum redundancy feature selection. In M. R. Oleg Okun and G. Valentini, editors, *Supervised and Unsupervised Ensemble Methods and their Applications - SUEMA 2010, ECML/PKDD 2010 Workshop*, pages 11–22, 2010. URL `http://suema10.dsi.unimi.it`.

L. Song, J. Bedo, K. M. Borgwardt, A. Gretton, and A. Smola. Gene selection via the bahsic family of algorithms. *Bioinformatics*, 23(13):i490–i498, 2007a.

L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo. Supervised feature selection via dependence estimation. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 823–830, 2007b.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, May 2002.

V. G. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences of the United States of America*, 98(9):5116–5121, 2001.

V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.

L. Yu, C. Ding, and S. Loscalzo. Stable feature selection via dense feature groups. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 803–811, 2008.