A Feature Selection Method Using Misclassified Patterns

D. M. Shawky and A. F. Ali

Abstract—**Feature selection (FS) is a key step in the data mining process. In FS, the objective is to select the smallest subset of features that reduces complexity and ensures generalization. In this paper, we present a combined filter-wrapper feature selection approach using misclassified data. The learning process starts with only one feature, which gives a large number of misclassified patterns. Only these patterns are used to select the next best feature which is added to the first one. By focusing on the misclassified patterns, the learner is undistracted and hence, it can select the relevant features more effectively and faster. The process continues until the classification results are within the required accuracy. The approach is applied to three datasets with high dimensional features using a variety of selection models and search strategies. Experimental results demonstrate the efficiency of the proposed approach in the two-class classification tasks.**

Index Terms—**Feature selection, misclassified patterns, pattern classification.**

I. Introduction

In classification tasks, the existence of a large number of features usually adds complexity and noise to the training process. Thus, it is important to choose the most relevant set of features to reduce the training time and increase the classification accuracy. FS has been an active field of research that is widely applied to many areas such as genomic analysis [1], text mining [2], image retrieval [3], and intrusion detection [4]. As new applications emerge in recent years, many new challenges arise. Thus, novel theories and methods addressing high-dimensional and complex features are needed. Stable FS [5], optimal redundancy removal [6] and the exploitation of auxiliary data and prior knowledge in FS [7], [8] are among the most fundamental and challenging problems in FS. In addition, FS faces problems that arise from linearly non-separable data, the effect of the sampling process, and the noise added during data generation [9], [10].

In this paper, an approach for FS using misclassified patterns (FSMC) is presented. The classification process starts with only one feature describing the data as a result of applying a FS criterion to the complete training data set. A classification process is applied using the extracted feature. The classification accuracy is then calculated. If it is not as desired, the set of misclassified patterns is calculated. We apply the FS criterion to the set of misclassified patterns only. The generated feature is added to the first one and the classification process is repeated but using the two features generated so far. Again, the accuracy is calculated and if it is not as desired, the new set of misclassified data is calculated and the process of FS is repeated until we reach the desired classification accuracy. The set of features used in the last classification process is the optimal set of features describing the data. This set generated by focusing only on the misclassified patterns. In this sense, we test the rule "We must learn from our mistakes" to see whether it can be beneficial in the context of machine learning.

To evaluate and to demonstrate the proposed method, we applied it to three complex datasets that were used in the NIPS 2003 FS challenge [11].We implemented FSMC with different learning classifiers. We compared the performances of these classifiers using FSMC and the popular sequential forward feature (SFF) selection method using all available training data. Experimental results show that, at the beginning of the classification process, the proposed FS approach has a higher classification accuracy compared to SFF selection method. Thus, the proposed approach can be used to generate a quick and efficient features set that can roughly describe the given data. This suggests the suitability of the proposed method for on-line classification problems.

The rest of this paper is organized as follows. Section 2 gives a review of the existing FS techniques. Section 3 introduces the proposed approach for FS. Section 4 presents the experimental setup, while Section 5 presents the experimental results and compares the proposed approach to the SFF. Finally, Section 6 draws conclusions, limitations, and outlines ideas for future work.

II. Feature Selection

There has been a great deal of research on FS, e.g., [12]-[14]. FS algorithms can be roughly grouped into two categories: filter methods and wrapper methods. Filter methods rely on general characteristics of the data to evaluate and to select the feature subsets without involving the chosen learning algorithm. Wrapper methods use the performance of the chosen learning algorithm to evaluate each candidate feature subset. Wrapper methods can be significantly slower than filter methods if the learning algorithm takes a long time to run [13]. As shown in the survey done by Hall [14], wrapper methods such as forward selection and backward elimination [15] have high performance with high computational costs. On the other hand, filter methods such as Relief [15], [16], Information Gain and FOCUS [17] can be executed more quickly with lower performance than that of wrapper methods. Some advanced wrapper methods such as CFS [18], which executes a substitute evaluator instead of a learned evaluator, have lower computational costs than wrapper methods. However, these performances are still non-practical, compared with wrapper methods. In [19], an extensive review of existing work on FS is provided.

Manuscript received May 5, 2011; revised September 22, 2011.

D. M. Shawky is with the Engineering Mathematics Department, Cairo University, Egypt, (e-mail: doaashawky@yahoo.com).

A. F. Ali is with the Biomedical Engineering Department, Helwan University, Cairo, Egypt, (e-mail: ahmed.farag@mcit.gov.eg).

III. Proposed Feature Selection Approach

Corresponding to the basic ideas behind the approach presented above, it consists of the following main steps:

1) Apply a suitable FS criterion to estimate the best feature describing a given data set. This set represents the initial set of features (IS).
2) Using IS, classify the data using a suitable classifier.
3) Calculate the classifier performance. If the performance is not accepted, go to step 4. Otherwise, stop.
4) Generate the set of misclassified patterns.
5) Apply the FS criterion that was used in Step 1 to the set of misclassified data and select its best feature. Add this feature to the set IS.
6) Re-classify the original set of data using the set IS.
7) Go to step 3.

In the rest of this section, details about the above steps are presented.

A. *Applied Feature Selection Criteria: Filters*

Filters are usually used as a pre-processing step since they are simple and fast. A widely-used filter method is to apply a uni-variate criterion separately on each feature, assuming that there is no interaction between features.

We used as filters three different commonly used filters that belong to different categories as reviewed in the literature [19]. These filters include the following:

1) T-test (TTEST): which is a parametric (it assumes that the data follows the normal distribution) uni-variate filter. Using this test, we calculate the maximum separation between classes and ranks the features according to the maximum p-value, which means more confidence about the differentiating power.
2) Wilcoxon rank sum test (RANKSUM): which is a model free uni-variate filter similar to the t-test except that it is a non-parametric test, i.e., it does not assume that the data comes from normal distribution.
3) Information Gain (IG): This is a multi-variate filter. It selects the features that make information gain maximum. It works as follows: Let S be a set of instances, p_i is the fraction of instances with class_i, S_v is the subset of S with A = v, and Values(A) is the set of all possible values of A, then:

$$\text{Entropy}(S) = -\sum_i p_i \log_2 p_i \quad (1)$$

$$IG = \text{Entropy}(S) - \sum_{v\in \text{Values}(A)} \frac{|S_v|}{|S|} . \text{Entropy}(S_v) \quad (2)$$

Using IG as a filter, the entropy and information gain is calculated for all features using (1) and (2). Then the one with highest IG is chosen. It should be noticed that TTEST and RANKSUM methods do not consider interaction between features. Thus, features selected based on their individual ranking may also contain redundant information. However, IG considers interaction between features.

B. *Used Classifiers*

We evaluated our algorithm using four different popular learning classifiers belonging to different categories as classified in [20].

C. *A logic-based classifier: Decision Tree*

Decision trees classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values [20].

In order to avoid over-fitting problems we can pre-prune the decision tree by not allowing it to grow to its full size [20]. Establishing a non-trivial termination criterion such as a threshold test for the feature quality metric can be used. Well-known pruning methods are presented in [21].

Many tree-splitting criteria are proposed in the literature [22]. These criteria include rules derived from information theory (e.g., Shannon entropy) [23], and rules derived from distance measures which measure the distance between class probability distributions. Gini's index of diversity is one of the most commonly used metrics. It is a measure of the inequality of a distribution where a value of zero represents total equality and a value of one represents maximal inequality [24]. In addition to the splitting criterion, the minimum number of observations (MinSplit) an impure node must have to be split has to be set.

D. *A perceptron-based classifier: A multi-layered Artificial Neural Networks (ANNs)*

Multilayered Perceptrons (Artificial Neural Networks) have been created to classify non-linearly separable data [25]. A multi-layer neural network consists of large number of units (neurons) joined together in a pattern of connections. Units in a net are usually divided into three classes: input units, which receive information to be processed; output units, where the results of the processing are found; and units in between known as hidden units. Feed-forward ANNs allow signals to travel one way only, from input to output. Generally, determining the size of the hidden layer is a problem, because an underestimate of the number of neurons can lead to poor approximation and generalization capabilities, while excessive nodes can result in over-fitting [26]. The weights of the net to be trained are initially set to random values, and then instances of the training set are repeatedly exposed to the net. The values for the input are placed on the input units and the output of the net is compared with the desired output. Then, all the weights in the net are adjusted slightly in the direction that would bring the output values of the net closer to the values for the desired output. There are several algorithms with which a network can be trained [27]. However, the most well-known and widely used learning algorithm to estimate the values of the weights is the Back Propagation (BP) algorithm [20]. A review of the existing work in ANNs is provided in [28].

E. *Support Vector machines (SVMs)*

Support Vector Machines (SVMs) [29] have been widely applied to pattern classification problems and non-linear regressions. The basic idea of the SVM algorithm is to find an optimal hyper-plane that can maximize the margin between two groups. The vectors that are closest to the optimal hyper-plane are called support vectors. Maximizing the margin and thereby creating the largest possible distance between the separating hyper-plane and the patterns on either side of it has been proven to reduce an upper bound on the expected generalization error. SVMs show excellent performance in handling large feature space and over-fitting problems [30]-[32]. An excellent survey of SVMs can be found in [33].

If the training data is linearly separable, then a pair (w, b) exists such that

$w^T x_i + b \geq 1$ for all $x_i \in$ class 1 (3)

$w^T x_i + b \leq -1$ for all $x_i \in$ class 2 (4)

with the decision rule given by:

$f_{w,b} = \mathrm{sgn}(w^T + b)$ (5)

where w is called the weight vector and b the bias (or threshold). For linearly separable two classes, an optimum separating hyper-plane can be found by minimizing the squared norm of the separating hyper-plane. This is a convex quadratic programming (QP) problem:

Minimize $\Phi(w) = \frac{1}{2}\|w\|^2$ (6)

Subject to $y_i(w^T x_i + b) \geq 1$, i=1,2,…,n (7)

In the case of linearly separable data, once the optimum separating hyper-plane is found, data points that lie on its margin are known as support vector points and the solution is represented as a linear combination of only these points. Other data points are ignored. Therefore, the model complexity of an SVM is unaffected by the number of features encountered in the training data. For this reason, SVMs are well suited to deal with learning tasks where the number of features is large with respect to the number of training instances. Many parameters must be set when using an SVM. Firstly, the function used to map the training data into kernel space (e.g., linear kernel, quadratic, or radial basis function). Secondly, we must determine the method that is used for finding the separating hyper-plane. These methods include Quadratic Programming (QP), Sequential Minimal Optimization (SMO), or Least Squares. Standard QP method is time-consuming and is mostly impractical for large problems. Meanwhile, SMO is a simple algorithm that can, relatively quickly, solve the SVM QP problem [32]. It should be mentioned that the training optimization problem of the SVM necessarily reaches a global minimum, and avoids ending in a local minimum, which may happen in other search algorithms such as neural networks [21].

1) A statistical learner: k-Nearest Neighbour (KNN)

K-nearest neighbour (kNN) is used to predict the response of an observation using a nonparametric estimate of the response distribution of its k nearest (i.e., in predictor space) neighbours. KNN classification is based on the assumption that the characteristics of members of the same class should be similar and thus, observations located close together in covariate (statistical) space are members of the same class or at least have the same posterior distributions on their respective classes [34].

Given a set of training samples {p_1, p_2,…, p_n} and an unknown sample p , kNN finds k training samples closest to p. Let these samples and their corresponding class labels be defined by the sets $\{p_1^m, p_2^m, \ldots, p_k^m\}$ and {y_1, y_2,…,y_k} respectively. KNN classifies p to the class which has the greatest number of representatives in the latter set. In other words, the classification is performed by taking the majority vote among k nearest neighbours of p. The choice of k affects the performance of the kNN algorithm [21]. Experiments showed that the performance of kNN was not sensitive to the exact choice of k when k was large. It was found that for small values of k, the kNN algorithm was more robust than the single nearest neighbour algorithm (1NN) for the majority of large datasets tested [21]. The relative distance between instances is determined by using a distance metric. The distance metric must minimize the distance between two similarly classified instances, while maximizing the distance between instances of different classes. Some commonly used metrics include Euclidean, Manhattan and Chebychev distance metrics [21]. In addition to distance metrics, a rule for classification is needed. Usually, the majority rule is used. That is, a sample point is assigned to the class the majority of the k nearest neighbours are from. When classifying to more than two groups or when using an even value for k, it might be necessary to break a tie in the number of nearest neighbours. We can either select a random tiebreaker, or nearest neighbour among the tied groups to break the tie.

IV. Experimental Setup

The used datasets are presented in Table I together with some important metrics. These data sets were used in the NIPS 2003 variable selection benchmark [35]. Details about how these data sets are processed and prepared can be found in [35]. The reason for using these datasets is that they are available. In addition, they span a variety of domains. The data are split into training set and test set. One dataset (MADELON) was artificially constructed to illustrate a particular difficulty: selecting a feature set when no feature is informative by itself [35].

TABLE I: Used Datasets

Dataset	ARCENE	GISETTE	MADELON
Domain	Mass Spectrometry	Digit Recognition	Artificial
# of Features	10000	5000	500
#of Training Examples	100	6000	2000
#of Test Examples	100	1000	600

Table II presents the set of parameters associated with the used classifiers.

TABLE II: Parameters of Used Classifiers

Classifier	Parameters
Decision Tree	Split Criterion: Gini's index, MinSplit=10.
ANN	20 layers feed-forward back-propagation network. Training function: gradient-descent. Transfer function: hyperbolic tangent sigmoid for hidden layers and linear for output layer.
kNN	K=3, Euclidean distance metric, and the majority rule.
SVMs	Linear kernel function with QP.

V. Experimental Results And Discussion

In order to evaluate the proposed approach, we applied it to three datasets. The obtained results are compared to those obtained when a sequential forward FS is applied to the complete training dataset. We compared the results obtained using these two modes. We call the former method FS using misclassified patterns (FSMC). Since the performance depends on the used classifiers, we combined the proposed method with the classifier's name. For example, using a decision tree classifier with sequential forward FS on the complete training data is denoted by DT-SFF. Meanwhile, using the same classifier but applying the used FS criterion to the misclassified patterns is denoted by DT-MC.

Fig.'s 1-3 show the testing accuracies on the ARCENE using the three used FS criteria TTEST, IG and RANKSUM, and the four used classifiers DT, ANN, kNN and SVMs respectively. Accuracies are shown for feature dimension 1 to 100. In each figure, the used classifier is tested against the two modes of FS; our method which is denoted by classifier-MC and the SFF denoted by classifier-SFF. Results obtained when using the other two datasets are shown in Fig.'s 4-9 which are presented in the Appendix.

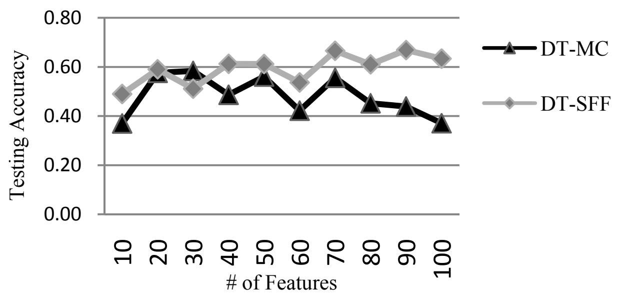
Regarding the FS criterion, IG outperforms TTEST and RANKSUM. The weakness of TTEST and RANKSUM is that selection ignores the redundancy and interaction among the features. On the other hand, IG filters detect features interactions.

In addition to the FS criterion, learning classifier is important to the testing performance. It is obvious from the shown figures that SVM-MC outperforms other combinations, especially at the beginning of the classification process where the number of misclassified patterns is still large. Testing accuracies are higher in all FSMC methods than they are in SFF methods at the beginning of the classification process. Thus misclassified patterns have important features that must be focused on during the classification process. However, the performance is decreased gradually as the number of features is increased. This is probably due to the over-fitting and generalization problems that occur in classification of small data size with high-dimensional features.

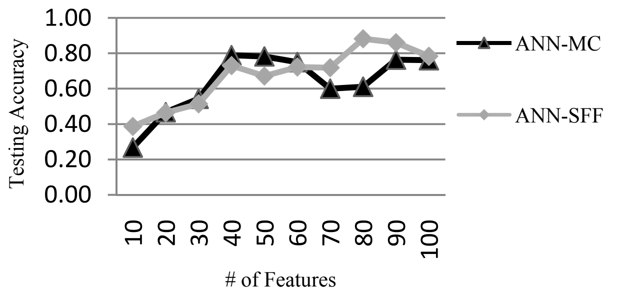
Overall, regarding the used classifier, the FS method SVM-MC performed the best, followed by DT-MC, kNN-MC, and ANN-MC.

Regarding the used data, experimental results show that the best testing accuracies were obtained when GISETTE was used. Since ARCENE dataset includes many irrelevant features, the overall testing accuracies were very poor; especially when ANN and kNN are used as the classifiers.

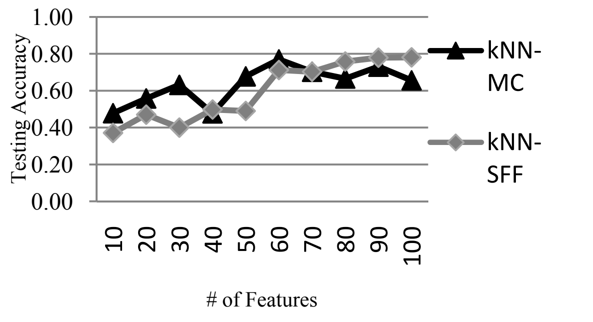
Irrespective of small size data problems, it should also be noticed that by forcing the learner to focus only on misclassified patterns, it can learn more accurately than being distracted by the complete set of the training data. Thus, the proposed technique can be used to generate a quick feature set that roughly describes the data with an accepted accuracy. This suggests the validity of the statement "We must learn from our mistakes" in the machine learning process.



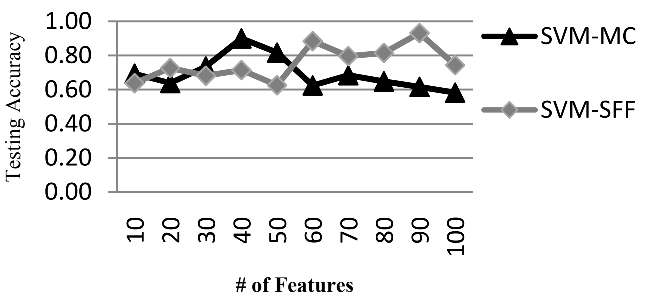
(a) Classifier DT



(b) Classifier ANN

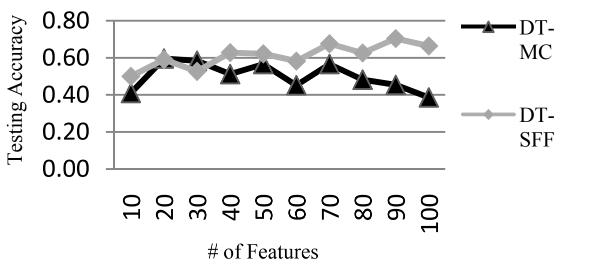


(c) Classifier kNN

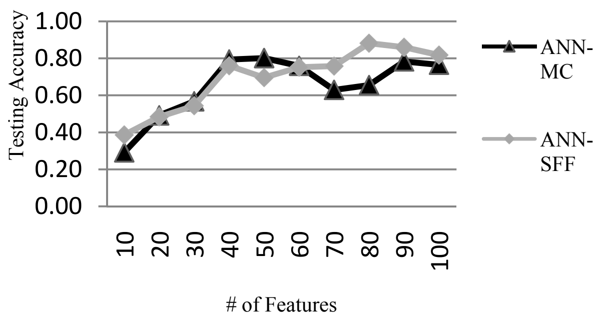


(d) Classifier SVM

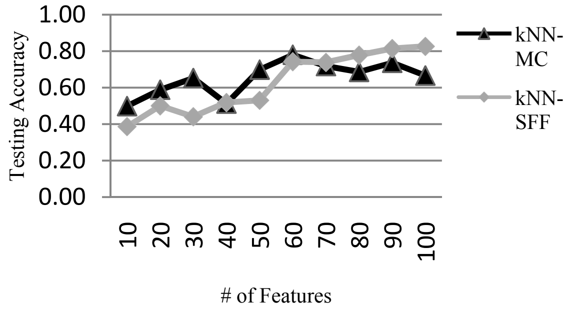
Fig. 1.Testing accuracies on ARCENE using TTEST and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.



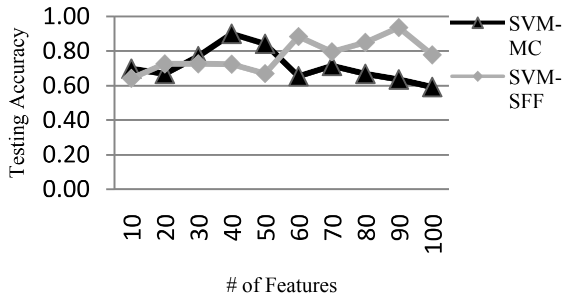
(a) Classifier DT



(b) Classifier ANN

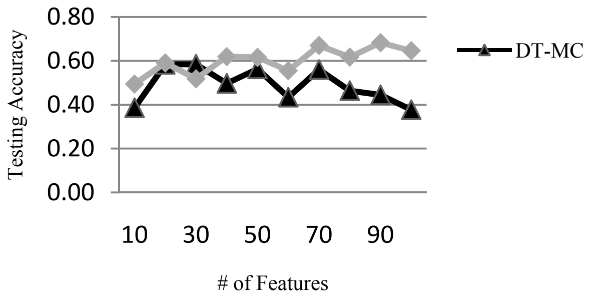


(c) Classifier kNN

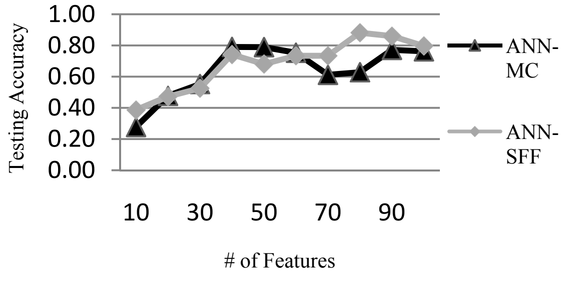


(d) Classifier SVM

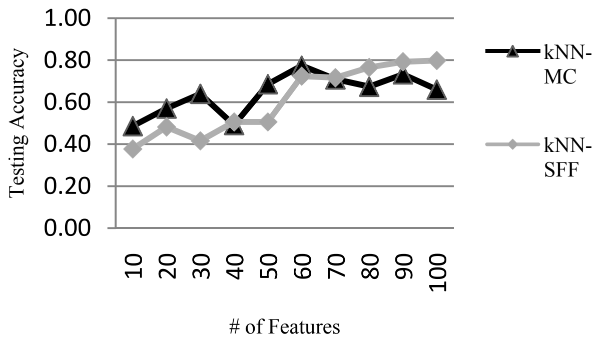
Fig. 2.Testing accuracies on ARCENE using IG and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.



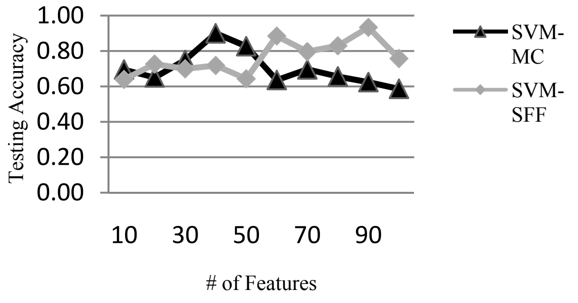
(a) Classifier DT



(b) Classifier ANN



(c) Classifier kNN



(d) Classifier SVM

Fig. 3.Testing accuracies on ARCENE using RANKSUM and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.

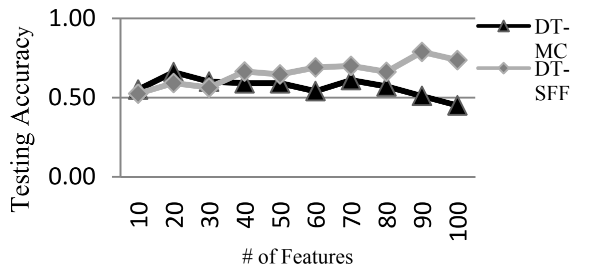
VI. Conclusions and Future Work

Since it is too expensive to use all available features in classification tasks especially when the dimension of features is in the order of thousands, we need advanced approaches to mine the minimum features with the highest prediction accuracy for complex datasets.

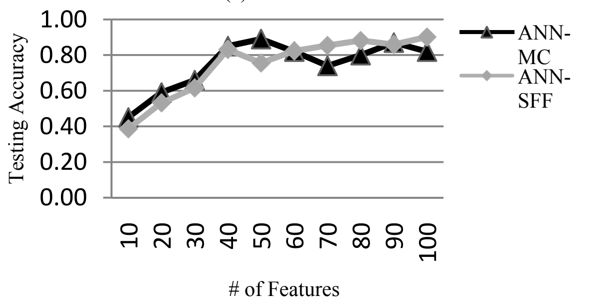
In this paper, an approach for FS using misclassified data is introduced. The presented approach is inspired by the analogy to human learning process where a stress on weak learning points can increase the learning activity. Our method of selecting features from misclassified patterns provides an efficient and inexpensive method of searching for the optimal or approximate optimal subset of features in high-dimensional data. We evaluated our approach against the popular sequential forward FS method, and experimental results on datasets with challenging difficulties were promising.

Further research issues may be concerned in studying threats to validity that our approach may suffer from. Especially those resulting from small sample size which may result in an inaccurate performance assessment as proposed in [36]. In addition, we intend to provide a mathematical justification of the obtained results by modeling the effect of extracting features from misclassified data on the classification error. In order to decrease the over-fitting problems, we may study the performance of the proposed method if -at some point- we stop focusing on the misclassified patterns only and use the complete training data instead. Thus, we intend to use an approach that combines the misclassified data with the complete training data.

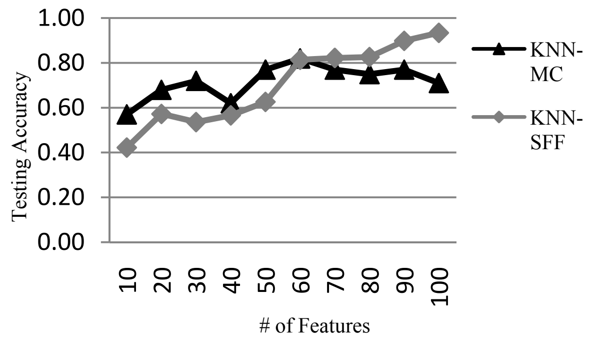
APPENDIX



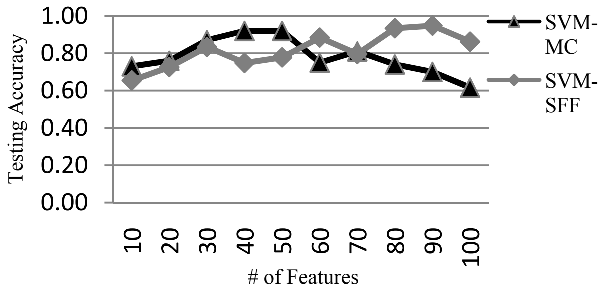
(a) Classifier DT



(b) Classifier ANN

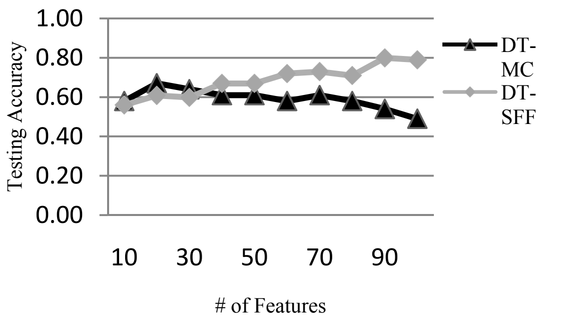


(c) Classifier kNN

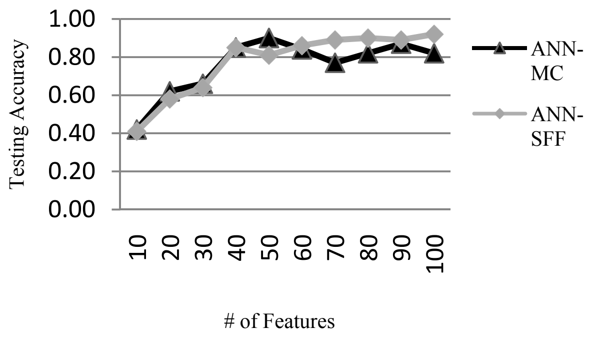


(d) Classifier SVM

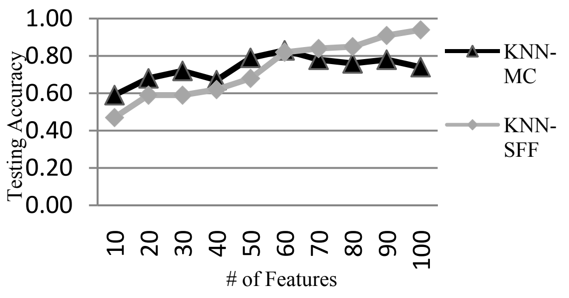
Fig. 4.Testing accuracies on MADELON using TTEST and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.



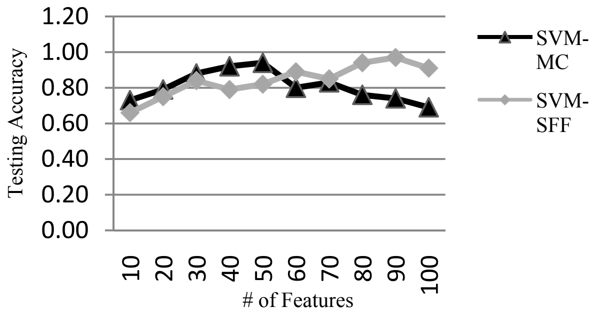
(a) Classifier DT



(b) Classifier ANN

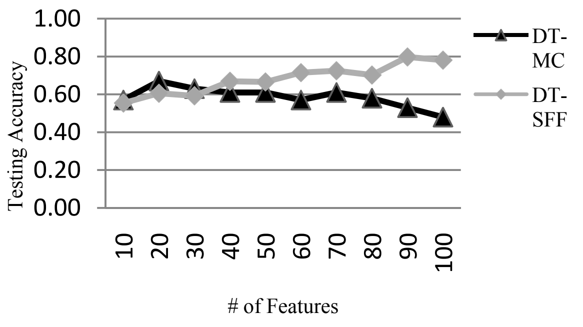


(c) Classifier kNN

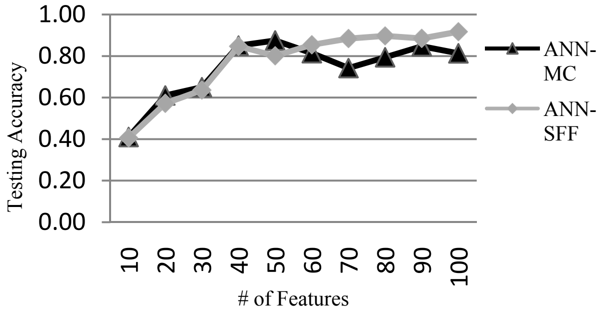


(d) Classifier SVM

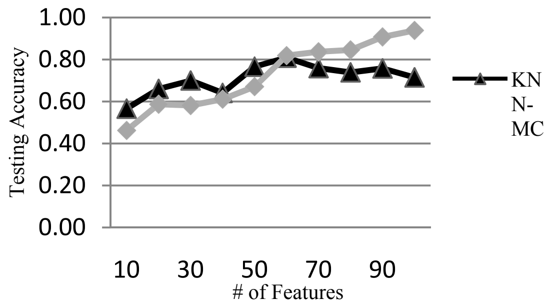
Fig. 5.Testing accuracies on MADELON using IG and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.



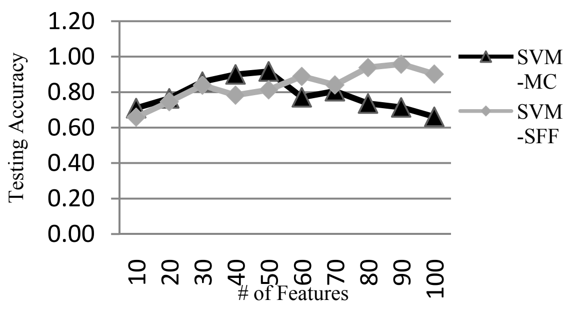
(a) Classifier DT



(b) Classifier ANN

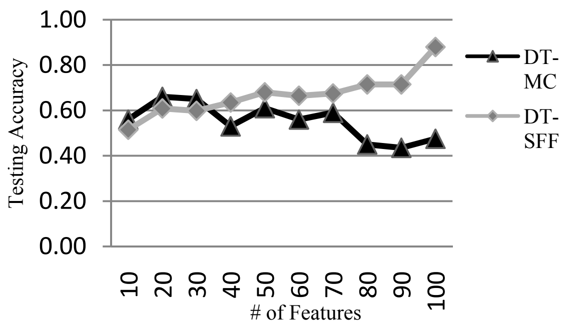


(c) Classifier DT

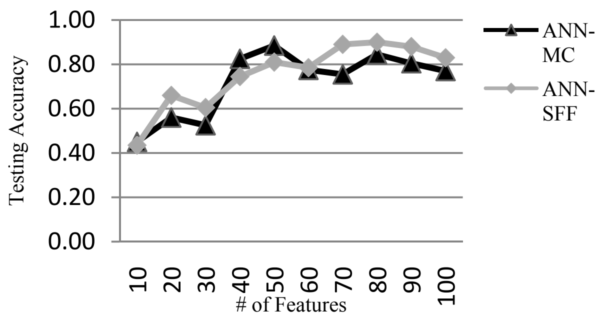


(d) Classifier SVM

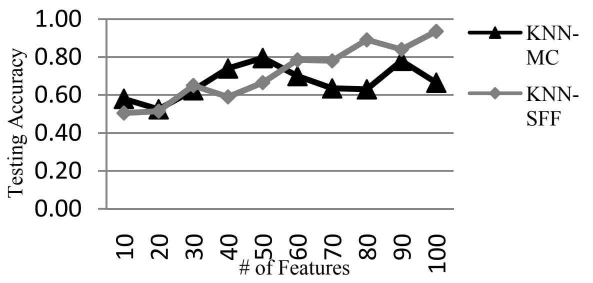
Fig. 6.Testing accuracies on MADELON using RANKSUM and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.



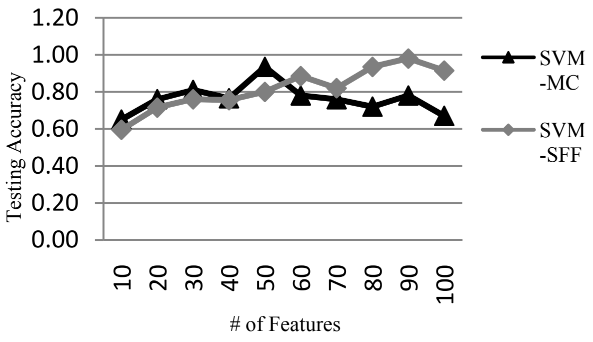
(a) Classifier DT



(b) Classifier ANN

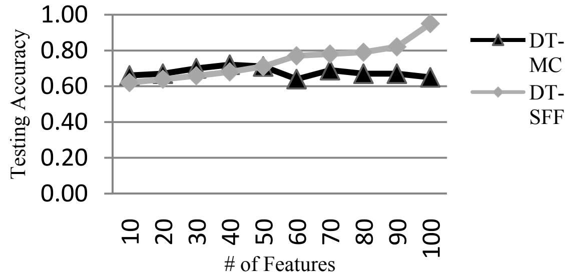


(c) Classifier kNN

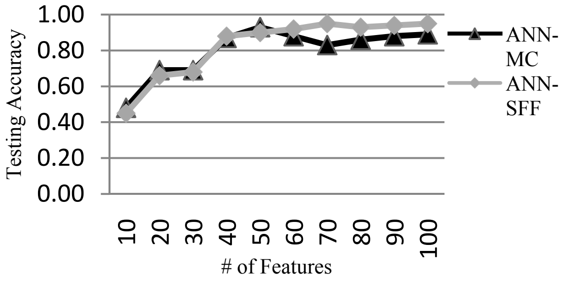


(d) Classifier SVM

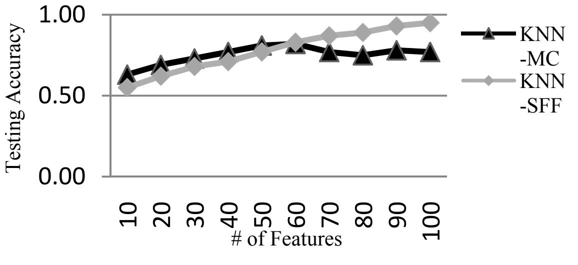
Fig.7.Testing accuracies on GISETTE using TTEST and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.



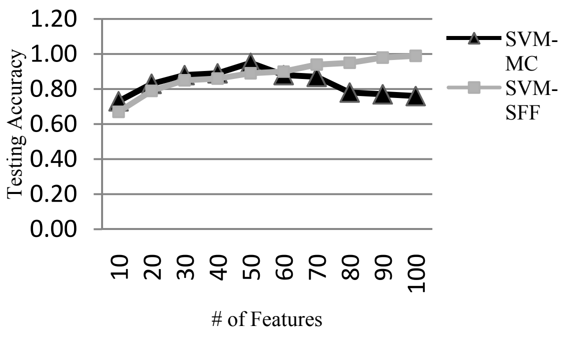
(a) Classifier DT



(b) Classifier ANN

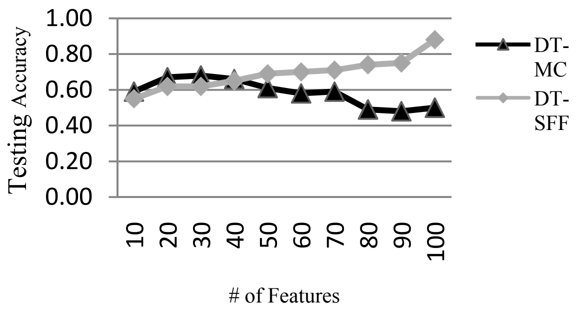


(c) Classifier kNN

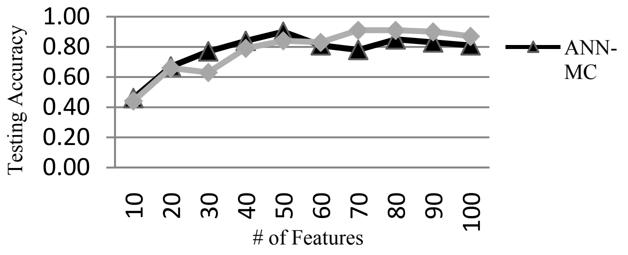


(d) Classifier SVM

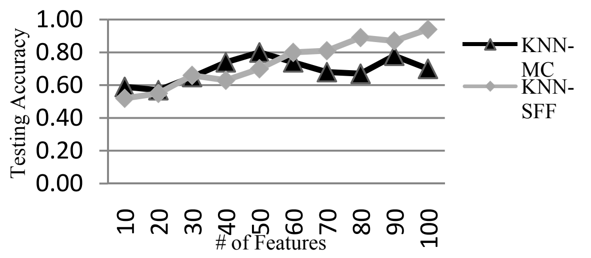
Fig. 8.Testing accuracies on GISETTE using IG and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.



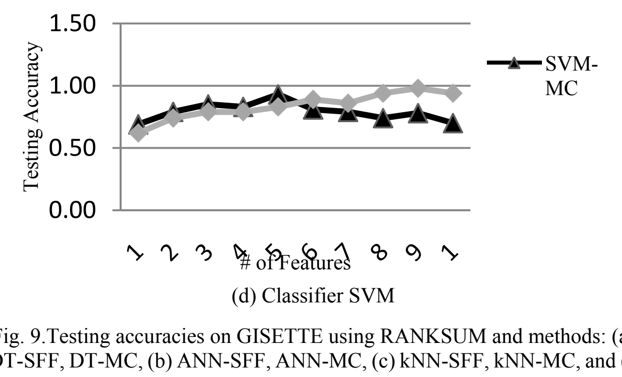
(a) Classifier DT



(b) Classifier ANN



(c) Classifier kNN



(d) Classifier SVM

Fig. 9.Testing accuracies on GISETTE using RANKSUM and methods: (a) DT-SFF, DT-MC, (b) ANN-SFF, ANN-MC, (c) kNN-SFF, kNN-MC, and (d) SVM-SFF, SVM-MC.

REFERENCES

[1] I. Inza, P. Larranaga, R. Blanco, and A. J. Cerrolaza, " Filter versus wrapper gene selection approaches in DNA microarray domains," *Artificial Intelligence in Medicine*, vol. 31, pp. 91-103, 2004.

[2] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Machine Learning Research*, vol. 3, pp. 1289-1305, 2003.

[3] D. L. Swets and J. J. Weng, "Efficient content-based image retrieval using automatic feature selection,"*IEEE International Symposium On Computer Vision*, pages 85-90, 1995.

[4] W. Lee, S. J. Stolfo, and K. W. Mok, "Adaptive intrusion detection: A data mining approach," *AI Review*, vol. 14, no.6, pp. 533- 567, 2000.

[5] 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, 2008.

[6] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," In *Proceedings of the Computational Systems Bioinformatics conference (CSB'03)*, pp. 523-529, 2003.

[7] Z. Zhao and H. Liu, "Multi-source feature selection via geometry-dependent covariance analysis," *JMLR Workshop and Conference Proceedings*, vol. 4, pp. 36-47, 2008.

[8] Z. Zhao, J. Wang, H. Liu, J. Ye, and Y. Chang, "Identifying biologically relevant genes via multiple heterogeneous data sources," *In Proceedings of the Fourteenth ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, 2008.

[9] K. Coombes, "Pre-processing mass spectrometry data," In Fundamentals of Data Mining in Genomics and Proteomics, M. Dubitzky, Ed., Boston: Kluwer, 2007,pp. 79–99.

[10] C. Ding, and H. Peng, " Minimum redundancy feature selection from microarray gene expression data," *In Proceedings of the IEEE Conference on Computational Systems Bioinformatics*, pp. 523–528, 2003.

[11] NIPS 2003 workshop on feature extraction and feature selection challenge, http://clopinet.com/isabelle/Projects/NIPS2003/.

[12] Y. Cai, J. He, and L. Lu, "Predicting Sumoylation Site by Feature Selection Method," *Bimolecular Structure and Dynamics*, vol. 28, no. 5, pp. 797-804, 2011.

[13] G. John, and R. Kohavi, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no.1-2, pp. 272-324. 1997..

[14] M. A. Hall, " Benchmarking attribute selection techniques for data mining," Department of Computer Science, University of Waikato, Tech. Rep. Working Paper 00/10, 2000.

[15] K. Kira and L. Rendell, "A practical approach to feature selection," *In Proceedings of the Ninth International Conference on Machine Learning*, D. Sleeman and P. Edwards, Eds., pp. 249–256,1992.

[16] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," In Proceedings of the 1994 European Conference on Machine Learning, pp. 171–182, 1994.

[17] H. Alumualim and T. G. Dietterich, "Learning Boolean concepts in the presence of many irrelevant features," *Artificial Intelligence*, vol. 69, no. 1-2, pp. 279–305, 1994.

[18] M. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, Department of Computer Science, University of Waikato, 1998.

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

[20] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques", *Informatica*, vol. 31, pp. 249-268, 2007.

[21] T. Elomaa, and J. Rousu, "General and Efficient Multisplitting of Numerical Attributes," *Machine Learning*, vol. 36, pp. 201–244, 1999.

[22] P. Kristin Bennett, "Decision tree construction via linear programming," In *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pp. 97-101, 1992.

[23] S. Schwartz, J. Wiles, I. Gough, and S. Philips, "Connectionist, rule-based and bayesian decision aids: An empirical comparison," London, Chapman and Hall, 1993, pp. 264-278.

[24] B. Saul Gelfand, C. S. Ravishankar, and J. Edward Delp, "An iterative growing and pruning algorithm for classification tree design," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 163-174, 1991.

[25] D. E. Rumelhart, G. E.Hinton, and R. J. Williams, "Learning internal representations by error propagation" in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart, and J. L. McClelland et al., Eds, Cambridge, MA: MIT Press, vol. 1, 1986, pp. 318-362.

[26] L. S. Camargo, and T. Yoneyama, “Specification of Training Sets and the Number of Hidden Neurons for Multilayer Perceptrons,” *Neural Computation,* vol. 13, pp. 2673–2680, 2001.

[27] C. Neocleous, and C. Schizas, “Artificial Neural Network Learning: A Comparative Review,” LNAI 2308, Springer-Verlag Berlin Heidelberg, pp. 300–313, 2002.

[28] Zhang, G. , “Neural networks for classification: a survey,” in *IEEE Transactions on Systems*, vol. 30, no. 4, pp. 451-462, 2000.

[29] C. Cores, and V. N. Vapnik, “Support Vector Networks,” *Machine Learning*, vol. 20, pp. 273-29, 1995.

[30] I. Guyon, and A. Elissee, “An introduction to variable and feature selection”, *Machine Learning Research, Special Issue on Variable and Feature Selection*, vol. 3, pp. 1157-1182, 2003.

[31] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, “Gene selection for cancer classification using support vector machines”, *Machine Learning*, vol. 46, pp. 389-422, 2002.

[32] J. Platt, “Using sparseness and analytic QP to speed training of support vector machines”, in *Advances in neural information processing systems,* M. Kearns, S. Solla, and D. Cohn, Eds., MIT Press, 1999.

[33] C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp.1-47, 1998.

[34] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, “ KNN Model-Based Approach in Classification,” *Lecture Notes in Computer Science*, vol. 2888, pp. 986 – 996, 2003.

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

[36] S. Lee, “Mistakes in validating the accuracy of a prediction classifier in high-dimensional but small-sample microarray data”, *Statistical Methods in Medical Research,* vol.17, pp. 635–642, 2008.