Tumor Classification from Gene Expression Data: A Coding-Based Multiclass Learning Approach

Alexander Hüntemann[1,⋆], José C. González[2], and Elizabeth Tapia[3]

[1] Katholieke Universiteit Leuven, Celestijnenlaan 300B, 3000 Leuven, Belgium
Alexander.Huntemann@mech.kuleuven.be

[2] E.T.S.I. Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain
jgonzalez@gsi.dit.upm.es

[3] Facultad de Ciencias Exactas, Ingeniería Agromesura, Escuela de Ingeniería Electrónica, Riobamba 245 bis, 2000 Rosario, Argentina
etapia@eie.fceia.unr.edu.ar

Abstract. The effectiveness of cancer treatment depends strongly on an accurate diagnosis. In this paper we propose a system for automatic and precise diagnosis of a tumor's origin based on genetic data. This system is based on a combination of coding theory techniques and machine learning algorithms. In particular, tumor classification is described as a multiclass learning setup, where gene expression values serve the system to distinguish between types of tumors. Since multiclass learning is intrinsically complex, the data is divided into several biclass problems whose results are combined with an error correcting linear block code. The robustness of the prediction is increased as errors of the base binary classifiers are corrected by the linear code. Promising results have been achieved with a best case precision of 72% when the system was tested on real data from cancer patients.

1 Introduction

Effective cancer treatment depends strongly on an accurate diagnosis of the type of tumor. Nowadays, the diagnosis of such malignancy relies strongly on histopathological and clinical data. Molecular tests have not yet been widely exploited to predict the type of cancer since molecular markers have not been identified for all possible tumors.

A promising technology has been introduced in molecular biology called microarrays, where thousands of genes can be analyzed simultaneously. On the surface of these devices, fragments of DNA or RNA are deposited. Then, a dyed sample of tissue is applied to the microarray for analysis. Hybridization occurs

⋆ This work has been done while the author was at the Department of Telematic Systems Engineering, Universidad Politécnica de Madrid, Spain

© Springer-Verlag Berlin Heidelberg 2005

at the spots where the sample genetic material matches the DNA/RNA on the surface of the microarray. The output of such an experiment is a colormap where the color intensity is related to the degree of expression of the genes under consideration. Since microarrays are built in a controlled fashion, it is possible to identify gene expression patterns from the obtained colormaps. This technique opens new ways for researchers to further investigate the existing relationships between gene expression patterns and cancer.

Following current trends in bioinformatics, the present article studies the classification of cancer tissues based on genetic information provided by a microarray. The analyzed data was obtained from the Whitehead Institute of Cancer Research in the USA [3]. It consists of two independent databases, i.e. one training set of 144 classified instances and one test set of 54 classified instances. Each instance is composed of 16063 gene expression values. The database is small in size since it is an extremely costly process to obtain sample tissues valid for later analysis. The employed samples correspond to primary biopsy tissues enriched by 50% in malignant cells in order to make the analysis of the data easier. They represent the 14 most common classes of cancer in human beings, i.e. breast, prostate, colon, lung, uterus, renal, ovary, bladder, pancreas, central nervous system cancer, leukaemia, lymphoma and mesothelioma. In every case independent medical experts of different cancer research centers in the USA verified the initial diagnosis twice. Once the samples were classified, a high throughput technique was used to profile the data genetically: Affymetrix's GeneChips.

The problem in the presented research is to train classifiers on a multiclass dataset of 144 instances and validate the results on the 54-instance test set. This problem's complexity is due to two reasons: first, there exist much more attributes than classified instances and second, multiclass machine learning is intrinsically more complex than biclass learning because most of the available algorithms are designed for the biclass case.

1.1 Related Work

Previous work has been published on the creation of a diagnosis system for cancer based on gene expression data and machine learning algorithms. We can compare for instance the results of Yeang et al. [15], Ramaswamy [9] and Golub [4]. Yeang and Ramaswamy have developed a system for multiclass diagnosis and evaluated it on the same dataset. They are considered, therefore, an important reference for the current research and their results establish a goal to achieve. Golub introduces in his article the statistical Signal to Noise Ratio (SNR) that we will also use for feature selection and correlation measurements. The approach followed by Yeang and Ramaswamy is similar to our system since they also decompose the multiclass learning problem into N binary and simpler problems. Their approach differs from ours on how to combine the individual predictions of the binary learners. They use a *One-Versus-All* code where every classifier is trained to distinguish each class from the rest. A disadvantage of their system is that contradictions may exist when several classifiers have positive output.

Instead, in our approach, linear codes are employed to combine the output of the individual binary learners to create a better prediction. Not every linear block code is suited for machine learning. We rely on the good properties of low density parity check codes (LDPC codes), as the Gallager code [6], [7]. An important reference for our work is Dietterich et al. [1]. Their publication explains how error correcting output codes can be used in multiclass learning as opposed to standard multiclass learning algorithms like, for instance, ID3. The foundations of how to apply LDPC codes in machine learning can be found in the Ph.D. dissertation of Tapia [12].

An important choice in multiclass learning based on linear coding theory is the underlying binary learning algorithm. As it is normally done in bioinformatics when dealing with gene expression datasets, an algorithm is selected capable of handling a number of attributes that is higher than the number of samples. This algorithm is called *Support Vector Machines* (SVM). In addition to the powerful SVM learners, the performance of the binary classifiers is improved by *boosting* them. A very good introduction to *boosting* and the employed *boosting* algorithm, AdaBoost.M1, can be found in the publication by Freund et al. [2].

The present article is structured in seven sections. After the brief introduction, the feature selection process dealing with the dimensionality problem of the data is explained. Then, an overview over all preprocessing steps is given. Later, the selected machine learning approach is detailed. In the fifth section, the results of the classification on a real dataset are presented. A conclusion and an acknowledgement section finish the article.

2 Feature Selection

The complexity of the problem is addressed partially by a feature selection process. The objective is to find out which genes are most correlated with the class distinction and use only these genes in the classification. Attribute selection is performed according to a parameter called Signal to Noise Ratio (SNR) [4] for its similarity to the SNR parameter used in communication theory.

$$SNR\left(\overrightarrow{gen}, \overrightarrow{class}\right) = \frac{\mu_{class+}\left(\overrightarrow{gen}\right) - \mu_{class-}\left(\overrightarrow{gen}\right)}{\sigma_{class+}\left(\overrightarrow{gen}\right) + \sigma_{class-}\left(\overrightarrow{gen}\right)} . \qquad (1)$$

The main advantages of the SNR statistic are that it respects the correlation structure of the data and it does not assume any hypothesis about the statistical distribution of the samples, which would have to be verified [9].

Another added value is that it can be computed empirically if the selected attributes are meaningful in a statistical sense by a hypothesis contrast. The goal of the hypothesis contrast is to state, with a given significance level, if a hypothesis is valid or not. In our case the null hypothesis is that the SNR ratio does not select the most correlated genes with the class distinction. This hypothesis must be rejected for the machine learning process to remain valid.

A hypothesis contrast can have two possible outcomes: the null hypothesis can be either true or false. The significance level α is the probability to reject a correct null hypothesis H_0. We call the error of rejecting a correct null hypothesis Type I error. Another possible scenario is to accept a false H_0. This type of error is called Type II and its probability is given by the power of the contrast, β. If we reduce the significance level α of a hypothesis contrast, we reduce the probability to reject a valid H_0 hypothesis. A *p-value* is the lowest significance level at which it is possible to reject the null hypothesis. The *p-value* also gives the lowest α for which the observed statistic is meaningful. It is possible to use the *p-value* in a decision rule for accepting or rejecting the null hypothesis.

$$p - value \leq \alpha \Longleftrightarrow \text{ reject } H_0; \; p - value > \alpha \Longleftrightarrow \text{ accept } H_0 \; . \tag{2}$$

In our case it is possible to calculate *p-values* experimentally[1] and compare them with the significance level of the null hypothesis in order to contrast its validity. The *p-value* of a gene is the probability that the SNR hypothesis contrast of a random permutation of the class labels is greater than or equal to the SNR observed. We calculate in some experiments the number of genes that exceed the real SNR ratios, when the class labels are permutated. Formally:

$$p - value\left(\overrightarrow{gen_i}\right) = \sum_{B=1}^{N_{PERM}} \frac{\#\left\{j \in \{1,2,\ldots,N_{genes}\} : |C_j| \geq |A_i|\right\}}{N_{genes} \cdot N_{perm}} \tag{3}$$

$$A_i = SNR_{\text{real}}\left(\overrightarrow{gen_i}, \overrightarrow{class}\right); \quad C_j = SNR^{B}_{\text{rand}}\left(\overrightarrow{gen_j}, \overrightarrow{class^*}\right) \; .$$

where C_j is the SNR statistic in random permutation B of the class labels, $\overrightarrow{class^*}$. If the *p-value* calculated in the permutation test is lower than the significance level α, the null hypothesis is rejected.

In figure 1 we can see that, even in the worst case, the null hypothesis can be rejected since there are enough genes with a small *p-value*. We can conclude that the SNR ratio is a good correlation measure with the class distinction.

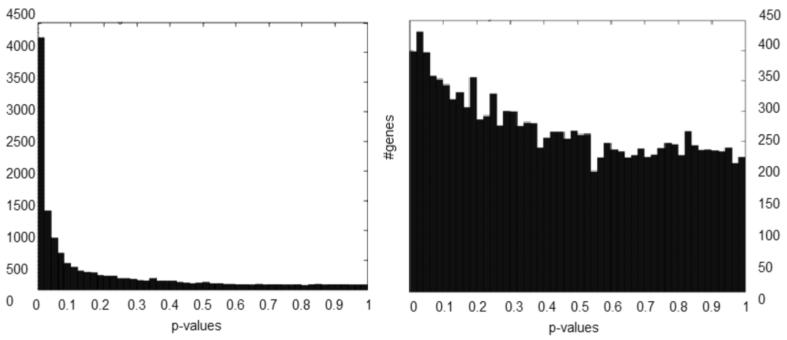

Fig. 1. Histograms representing the number of genes as function of the *p-values*. The figure shows on the left the best case (bladder cancer) and on the right the worst case (ovary cancer) scenario for the hypothesis contrast. In both cases the null hypothesis can be rejected since there are enough genes with small *p-values*

[1] For further explanation please refer to [11]

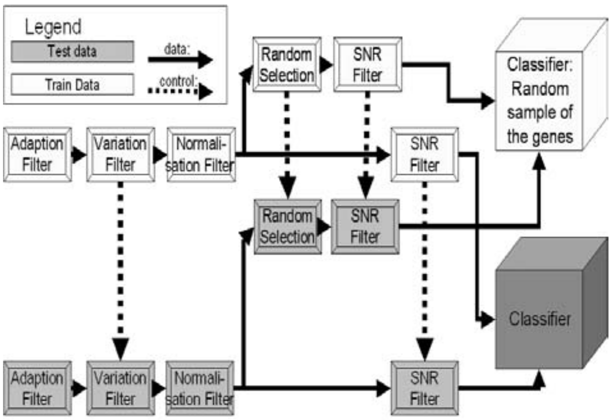

Fig. 2. Preprocessing and filtering stages. Two paths of data processing are shown: one affecting the training data and the other referring to the test set. Both databases must be compatible after modifying the attribute set in order to achieve meaningful results

3 Preprocessing

In figure 2 all stages of preprocessing and filtering are summarized. On top, the flow for the training set is represented and below the steps relative to the test set are shown. The process starts with an adaptation filter that transforms the data from its original format into an understandable format for the employed machine learning library, WEKA [14]. The adaptation of the data is followed by a variation filter that eliminates those genes with not enough dynamic variation among samples, i.e. the genes without marked difference in expression across different classes. Genes selected in the training dataset must also be selected in the test data by the variation filter for both datasets to be compatible. After the variation filter, the data is normalized to mean zero and standard deviation one. The data is normalized in order to avoid that genes with higher absolute expression values mask other genes with smaller values. At this point in the process, the tracks for both the test and training dataset split. With a random selection filter, attributes are chosen from the total set of attributes to verify if the attributes of the feature selection stage are truly marker genes. Both paths of training and test set contain as last stage of preprocessing a SNR filter that chooses the most correlated sets of attributes out of the total set. Note that again full compatibility between training and test set is maintained.

4 Machine Learning Approach

In order to deal with multiclass classification using biclass base algorithms, a novel approach is used related to coding theory. Binary classifiers are trained transforming the original fourteen class-learning task in N binary learning problems. The number of learners is higher than the amount of bits needed to code fourteen classes, i.e. four bits. This way, if one of the classifiers fails, the original class may still be retrieved if the codeword is decoded correctly using the properties of linear block codes. The underlying model employed is the transmission of codewords over a binary memoryless channel with additive white gaussian noise. The received codeword is decoded bearing in mind that it might have a small amount of errors. In this setting the errors are modeled according to the training error performance of the binary classifiers.

$$\overrightarrow{r} = \left(\overrightarrow{t} + \overrightarrow{e}\right) \bmod 2 = \left(G^T_{nxk} \cdot \overrightarrow{s_k} + \overrightarrow{e}\right) \bmod 2 \ . \tag{4}$$

In the above equation $\overrightarrow{s_k}$ represents a vector of the k possible source symbols and G^T_{nxk} is the generator matrix of the linear block code $C(n, k)$. The received codeword is the sum modulo two of the transmitted codeword and an error vector. At the moment of reception, both the transmitted and the error vectors are unknown. One of them has to be guessed based on the available data, i.e. the received vector and the properties of the linear block code. An optimum decoder estimates the transmitted vector with the maximum a posteriori probability of having transmitted this vector given the received vector and G^T_{nxk}. Formally:

$$\widehat{t} = \arg\max_t p\left(\overrightarrow{t} \,|\overrightarrow{r}, G^T_{nxk}\right) \ . \tag{5}$$

The main disadvantage of optimum decoding is its computational complexity that makes it impractical. Normally, the problem of optimum decoding is NP-complete [7].

Closely related to the generator matrix G^T_{nxk} is the parity check matrix $H_{(n-k)xn}$ that has the property of being orthogonal to the generator matrix. If this property is applied to the equation of reception in a memoryless channel, the syndrome relation is obtained:

$$syndrome : z = H_{(n-k)xn} \cdot \overrightarrow{e} \bmod 2 \ . \tag{6}$$

The syndrome of a vector can be used in the decoding process as it is done in the sum-prod algorithm that is used in the presented research.

4.1 Gallager Codes

A special kind of linear code that can be decoded iteratively is used as composing scheme. This code was developed by MacKay and Neal [7], [6]. Gallager codes have a parity check matrix with a very low density of '1's. Let's denote by $m = n - k$ the number of parity bits (rows of $H_{(n-k)xn}$) and by t the number of '1's of a column. The parity check matrix of the code is constructed by choosing randomly its bits. The number of '1's per column is constrained to t and the number of '1's per row is as uniform as possible.

Based on the syndrome relation, the following sets are defined:

$$L\left(m\right) \equiv \left\{l : H_{ml} = 1\right\}; \ M\left(l\right) \equiv \left\{m : H_{ml} = 1\right\} \ . \tag{7}$$

$L(i)$ represents the set of bits that participate in the syndrome equation z_i . $M(l)$ is the set of indexes of the parity check equations, which involve the bit l of the vector whose syndrome is calculated.

The sum-prod algorithm is a two-step process consisting of a horizontal and a vertical step. During these stages two parameters related to $H_{(n-k)xn}$, q^x_{ml} and r^x_{ml}, are updated iteratively until the syndrome condition (equation 6) is satisfied. The value q^x_{ml}, with $x = 1$ or $x = 0$, represents the probability that bit l of the vector whose syndrome is calculated is equal to x, given all the information obtained from all parity check equations except equation m. The value r^x_{ml}, with $x = 1$ or $x = 0$, gives the probability that syndrome equation m is satisfied if the bit l has the value x and the rest of the bits have a separable probability distribution given by:

$$\{q_{ml'} : l' \in L\left(m\right) \backslash l\} \ . \tag{8}$$

During the initialization stage, initial values are assigned to the variables q^x_{ml} with $H_{mxn} = 1$. These initial values are calculated as the a priori probabilities of the error vector. The horizontal step is devoted to calculate the r^x_{ml} variables for all bits of $L(m)$ iterating through the rows of the parity check matrix. During the vertical step the q^x_{ml} probabilities are updated using the r^x_{ml} values obtained in the horizontal step. For the details of how these values are calculated, we refer to the publications of MacKay and Neal [6], [7]. With q^x_{ml} it is possible to calculate the values of the a posteriori probabilities that are used to estimate the bits of the error vector. If the estimated error vector satisfies the syndrome condition (equation 6), the process is stopped. Otherwise, the horizontal and vertical steps are repeated, updating variables with the values of the previous iteration. The decoding is accomplished by setting a one on position l of the error vector if the a posteriori probability exceeds 0.5. As part of the decoding, the syndrome condition is always verified to see if all the constrains are met. An error occurs if a maximum number of iterations is exceeded. Nevertheless, the final values of an erroneous decoding process can serve as initial values for the next run of the sum-prod algorithm. Undetected errors can appear if the estimated error vector satisfies the syndrome equation (equation 6) but does not correspond to the actual transmission error pattern.

4.2 Relation Between Gallager Codes and Multiclass Learning

The learning system consists of the elements depicted in figure 3: channel coder, binary symmetric memoryless transmission channel and channel decoder. A binary source that produces binary symbols represents the training set. The supervisor is a novel element in the transmission model. This supervisor determines if the output of the system is correct. In the case errors are present, the system is updated accordingly. The supervisor calculates error probabilities of the binary classifiers, which later determine the behavior of the discrete memoryless channel of the model. The supervisor does not change the channel coder. Once obtained, the linear code, parity check and generator matrices remain fixed.

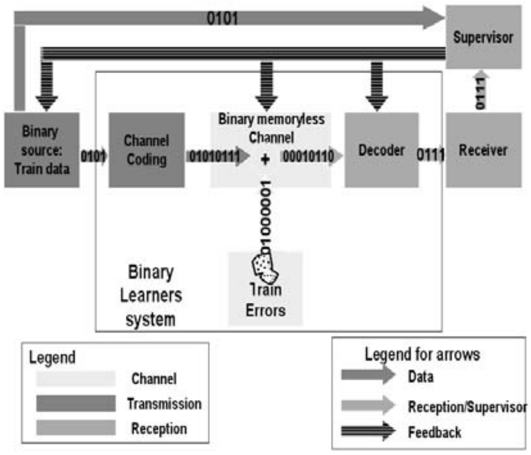

Fig. 3. Model for the multiclass learning problem. The original multiclass setting is transformed into N binary problems. The output of the N binary classifiers is later combined with a linear block code

The class of a new sample is obtained evaluating the output of all binary base classifiers. After all base learners have provided their output, a decoding process is started where the error probabilities are estimated according to the training errors of the binary classifiers. It is important to note that the system takes the performance of all base classifiers into account when doing a prediction of a new sample's class. This is achieved because the individual training error probabilities are used during the decoding process. Globally, the system adapts to the learning task by training binary learners and applying later a decoding approach.

Low Density Parity Check codes (LDPC codes), as for example the Gallager Codes, have an interesting property that makes them attractive for machine learning problems. There is a threshold for the crossover probability, p_0^*, that defines the maximum value at which the number of erroneous messages tends to zero as the number of iterations tends to infinity. If the number of '1's per column of $H_{(n-k)xn}$ is greater than three, the components of the error probability vector decrease exponentially. From this it is possible to infer that LDPC codes behave better if they are very long. The above condition can be satisfied more easily as p_0^* decreases with the channel rate. Therefore, if the performance of a code is not satisfactory, it is only necessary to reduce the channel rate to improve the results [12].

4.3 Support Vector Machines

So as to keep the training error low, strong binary classifiers are used: *boosted Support Vector Machines* (SVM). SVM is a novel machine learning algorithm widely employed for genomic data, where there are much more attributes than instances. The main advantage of SVM is that they perform a classification in high dimensionality feature spaces and do the computations in the original feature space by kernel functions. The main goal of the algorithm is to find the maximum margin hyperplane that separates the data, i.e. the best conceivable separation of samples so as to obtain the smallest possible error [8]. In the presented research, the SVM implementation of the WEKA machine learning library was used with polynomial kernel functions.

4.4 Boosting: *AdaBoost*

SVM is a powerful classification scheme but another algorithm called *boosting* is added to further improve the results. *Boosting* is a technique that minimizes the training error by performing several iterations on the training data. Normally, it is used in combination with weak learning algorithms in order to improve their performance. A *boosted* classifier is trained on different distributions of the initial training set. Every sample is assigned a probability related to the error probability when classifying it. On every iteration, a stronger effort is made on wrongly classified samples because the probability distribution of the samples is modified according to classification results. At the end all hypothesis are combined having a higher weight those hypothesis with smaller error. It is important to notice, however, that a small training error does not necessary imply small test errors. This is only true if the training and test databases have similar valued attributes for the same classes [2].

The *boosting* implementation employed in the presented research is called *AdaBoost.M1* [2] and forms part of the WEKA machine learning libraries [14]. In *AdaBoost.M1* a base classifier is trained for a fixed number of iterations on the training set. This base classifier returns in the i-th iteration a hypothesis that classifies the data minimizing the training error. The training error is calculated according to the probability distribution D_i that describes the difficulty to classify each sample. D_i is updated after every iteration proportionally to the training error. At first, D_0 is uniform for all the samples of the training set. D_{i+1} is calculated from D_i and the weak hypothesis h_i multiplying the weight of the sample by a number related to the training error. If a sample was classified correctly in the previous iteration, its weight is left unchanged for the next step. The dependency of the number that multiplies the weight distribution is such that erroneously classified samples get a higher probability in the next iteration. At the end of the process, the final hypothesis is obtained as a weighted sum of the weak hypothesis, h_i, being the weight related to the training error.

5 Results

In this section the results of the machine learning approach are presented. On two graphs the precision, i.e. the relation among correctly classified positives and the number of instances classified as positives, is displayed. Figure 4(A,B) clearly shows that with increasing decoding iterations of the LDPC recursive Gallager code, the error of the classifier is lower in average. The minimum error obtained

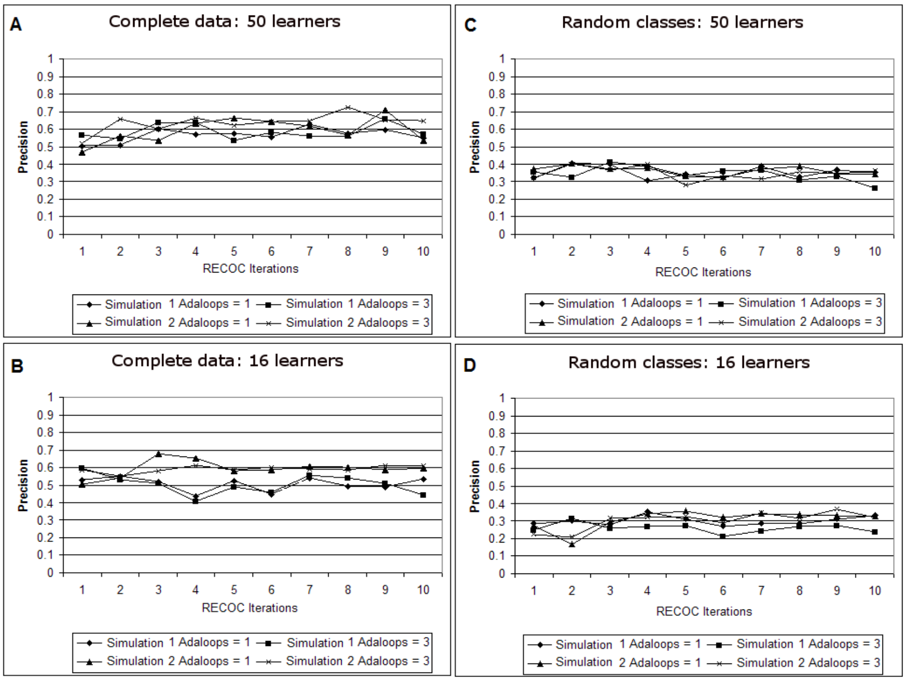

Fig. 4. Results of the classification process. The precision is plotted in function of the number of decoding and *boosting* iterations. Higher number of learners improve the performance of the classifiers (A,C). A best case precision of 72% is achieved (A). The results for random permutation data (C,D) are significantly worse than for the complete dataset (B,D)

in 10 simulations is of 40%, which is a little bit higher than the error achieved by Ramaswany et al. on the same database. It is important to mention however, that reducing the channel rate from 0.25 to 0.08, i.e. increasing the number of learners from 16 to 50, significantly improves the quality of the results. With a higher number of learners, the error correcting capability of a linear code increases and thus the overall error is reduced. Computational constraints only permitted to use at maximum 50 learners. Exceeding this number of classifiers significantly increased the computation time of the simulations to a non-tolerable limit given the available computers resources. With more powerful processors or even clusters we expect to further reduce test errors. Nevertheless, the results are comparable to other publications as for example [9] and [15] that worked on the same dataset and achieved a best-case precision of 78%.

In order to verify that the performance of the system is satisfactory, the obtained results are compared with the classification of randomly selected attribute sets. This way it can be proved that the genes selected by high SNR ratio are truly marker genes for the analyzed pathologies. From figure 4(C,D) it can be concluded that the classification provided by the most correlated genes in the sense of the SNR ratio is meaningful. In the case of the randomly selected genes, the best-case precision does not exceed 40%.

6 Conclusions

The addressed problem bears a high complexity due to the following reasons: first, the relation between number of attributes and instances is very low (curse of dimensionality) and second, multiclass machine learning is intrinsically very complicated[2]. The small size of the available sample databases increases the difficulty of training proper classifiers. In particular, only eight instances per category are available. Nevertheless, our results are comparable to the results published by other research groups [9], [15]. This is achieved by transforming the multiclass learning problem via a coding approach into several simpler biclass learning settings. The results of the binary classifiers are combined to a joint prediction with a linear block code that allows a small number of classification errors to be present. The learning problem is thus similar to a transmission through a binary symmetric memoryless channel. The output of each base binary classifier represents a bit of the received codeword and the training errors of the classifiers can be assimilated to the channel's error probability. Decoding is done with a recursive Gallager code with excellent performance. Due to the iterative decoding, error rates can be limited with sufficiently high number of learners.

It is also necessary to face the problem of dimensionality in the present research, which means that the number of attributes exceeds largely the number of available classified samples. This problem is common to gene expression datasets since modern high throughput techniques allow analyzing thousands of genes simultaneously. Still, it is a costly process to obtain many classified samples and, therefore, the size of the databases never exceeds a few hundred instances. This problem has been solved in our investigation using statistical feature selection algorithms based on the SNR ratio. It is verified statistically by a hypothesis contrast that the SNR ratio correctly measures the correlation to a class distinction before training the classifiers with the filtered datasets. The importance of the performed feature selection is not only related to machine learning requirements, it also offers insight into biological processes by identifying possible marker genes for a particular kind of tumor. As the number of attributes is still too high even after SNR filtering, a base learning algorithms is chosen able to deal with a huge number of features: *Support Vector Machines*.

Presently it is being investigated how to improve the computational efficiency in order to reduce the error rate of the process even more. The results obtained up to now encourage to continue researching on molecular diagnosis systems that may improve further the treatment of patients. The combined use of techniques from the areas of information and coding theory along with machine learning algorithms represent a new and encouraging approach to the use of gene expression data for medical diagnosis.

[2] In multiclass learning the random guessing probability is $\frac{1}{k}$ for k classes in comparison to the much higher random guessing probability for biclass learning of $\frac{1}{2}$

Acknowledgments

The authors wish to thank the Spanish Ministry of Culture and Education for funding this research that was carried out at the Universidad Politécnica de Madrid under a Research Collaboration Grant.

References

1. T. Dietterich and G. Bakiri: *Error-correcting output codes: A general method for improving multiclass inductive learning programs.* Proceedings of the 9^{th} National Conference on Artificial Intelligence (AAAI-91), AAAI Press, pp. 572-577, 1991.
2. Y. Freund and R. R. Schapire. *Experiments with a new boosting algorithm.* In Machine Learning: Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann, 1996.
3. www-genome.wi.mit.edu/MPR/GCM.html .
4. T.R. Golub et al., *Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression.* Science 1999 286: pp. 531-537, 1999.
5. S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1983.
6. D. J. C. MacKay, R. M. Neal, *Good Codes based on Very Sparse Matrices*; Cryptography and Coding the IMA Conference; 1995.
7. D. J. C. MacKay, R. M. Neal, *Good Error-Correcting Codes based on Very Sparse Matrices*, IEEE transactions on Information Theory, 1999.
8. S. Mukherjee, *Classifying Microarray Data Using Support Vector Machines.* In: A Practical Approach to Microarray Data Analysis, D. P. Berrar, W. Dubitzky and M. Granzow (Eds.), Kluwer Academic Publishers, pp. 166-185, 2003.
9. S. Ramaswamy et al., *Multi-Class Cancer Diagnosis Using Tumor Gene Expression Signatures*, PNAS 98: pp. 15149-15154, 2001.
10. B. Schölkpof, A. Smola, *Learning with Kernels Support Vector Machines, Regularization, Optimization and Beyond*, MIT Press, 2001.
11. J. Storey and R. Tibshirani, *Statistical Significance for Genome-Wide Experiments* http://www-stat.stanford.edu/˜tibs/ftp/fdringenomics.pdf, 2003.
12. E. Tapia, *New learning models based on recursive error correcting codes*, Doctoral Thesis, ETSI de Telecomunicación Universidad Politécnica de Madrid, Spain, 2001.
13. E. Tapia, J. C. González, A. Hüntemann, J. García-Villalba *Beyond Boosting: Recursive ECOC Learning Machines* In Multiple Classifier Systems, MCS 2004, Lecture Notes in Computer Science Vol. 3077, pp. 62-71. Springer 2004.
14. I. H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 1999.
15. C. H. Yeang et al. *Molecular classification of multiple tumor types*; Bioinformatics 17 (Suppl. 1): pp. 316-322, 2001.