New Preprocessing Methods for Better Classification of MR and IR Spectra

by
Alexandre Nikouline

A Thesis Submitted to the Faculty of Graduate Studies
In Partial Fulfillment of the Requirements for the
Degree of
Ph.D.

Department of Electrical and Computer Engineering
The University of Manitoba Winnipeg, Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

NEW PREPROCESSING METHODS FOR BETTER CLASSIFICATION OF MR AND IR SPECTRA

BY

ALEXANDRE NIKOULINE

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

TABLE OF CONTENTS

University of Manitoba

Abstract

New Preprocessing Methods for Better Classification of MR and IR Spectra

by Alexandre Nikouline

We introduce a global feature extraction method specifically designed to preprocess magnetic resonance spectra of biomedical origin. Such preprocessing is essential for the accurate and reliable classification of diseases or disease stages manifest in the spectra. The new method is Genetic Algorithm-guided. It is compared with our enhanced version of the Forward Selection algorithm ("Dynamic Programming"). Both seek and select optimal spectral subregions. These subregions necessarily retain spectral information, thus aiding the eventual identification of the biochemistry of disease presence and progression.

Both methods proved to be very useful for large datasets. The danger of overfitting related to the small number of samples in the datasets was demonstrated for both the artificial and real-life data. A bilinear regression model was used to quantitate the consequences of overfitting. Taking this in account, optimal parameters for the GA guided algorithm were recommended.

Acknowledgments

I would like to acknowledge my supervisor Dr. R.L. Somorjai for his support and training, but most of all for his patience. I also acknowledge the Institute for Biodiagnostics of the National Research Council Canada for financial support and for the excellent research and computing facilities.

I would like to thank all the great people in the Informatics group who helped me during the completion of this project, and especially Walter Roberson for his invaluable help with writing and debugging of my program.

I gratefully acknowledge my PhD committee for their support and advice.

List of Figures

List of Tables

List of Abbreviations

MR	Magnetic Resonance
IR	Infrared
PCA	Principal Component Analysis
GA	Genetic Algorithm
QDA	Quadratic Discriminant Analysis
LDA	Linear Discriminant Analysis
RDA	Regularized Discriminant Analysis
SNR	Signal to Noise Ratio
FID	Free Induction Decay
MRS	Magnetic Resonance Spectroscopy
FT	Fourier Transform
DP	Dynamic Programming
GA-ORS	GA-based Optimal Region Selection
MSE	Mean Square Error
MSE_{tr}	Mean Square Error for the Training Set
MSE_{ts}	Mean Square Error for the Test Set
TotalSS	Total Sum of Squares
LR-RSS	Residual Sum of Squares for the Linear Regression model
BR-RSS	Residual Sum of Squares for the Bilinear Regression model
LCL	Lower Control Limit

RSS	Residual Sum of Squares
PLS	Partial Least Square
wn	White Noise
rpp	Randomized Peak Position
rb	Random Baseline

1. Introduction and Literature Review

Noninvasive diagnostic techniques are finding their way into medicine. In many cases, e.g., MR or IR spectroscopy, the diagnosis (classification) is done based on a relatively small number of samples of high dimension. Thus, a typical classification problem in a biomedical application of MR involves 1024-dimensional spectra with at best 100 samples per class. From the mathematical point of view we are trying to classify an extremely sparse data set in a high dimensional space. Many classification methods applied to the raw data will fail, or will not be reliable. The problem is aggravated if we take into account that real life data are noisy, and classes are often assigned ambiguously.

Fortunately, the intrinsic dimensionality is usually smaller than the data size. This thesis is devoted to developing new methods of dimensionality reduction. The most popular current method is Principal Component Analysis (PCA). This method does not need a detailed introduction, and we'll only mention some basic facts in the literature review, since we use our own routine to carry out the PCA.

PCA is an unsupervised method, i.e., we do not need to know the class labels (diagnosis) to reduce the dimensionality. This is its main advantage, but frequently it is also the main disadvantage. Very often the magnitude of discriminating features of our spectra are comparable to noise in their contribution to the data. PCA is not capable of extracting such features.

Linear and Quadratic Discriminant Analysis are by far the most popular supervised methods. This is why the next part of our literature review will be devoted to them and to the relatively new Regularized Discriminant Analysis.

Recently, powerful Genetic Algorithms (GAs) have been used as feature selection methods /18,19/. Since a GA implementation will be the main preprocessing method

considered and developed in this work, the next topic of the literature review will be the GA.

A short review of the feature selection/extraction methods will follow the GA, since many new implementations are based on the GA.

The developed method can be used for data of different origin, but it was designed with NMR spectra as the input data in mind, so we will finish the literature review with the description of typical problems in spectra analysis.

1.1 Principal Component Analysis

We do not need to describe the history of PCA, or PCA itself in great detail. It can be found in any book on this subject, /1/ for example. Our purpose here will be to define some terms, which will be used throughout the work, and to discuss the limitations of the PCA.

Let ξ be a random n-dimensional vector, a sample from a distribution with covariance matrix Σ. Since Σ is a covariance matrix, it is symmetric and positive semidefinite. Thus it can be represented in the form $\Sigma = \Psi \cdot \Lambda \cdot \Psi^t$, where Ψ is an orthogonal matrix and $\Lambda = diag(\lambda_1,\ldots,\lambda_n)$ with $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n \geq 0$. Since $\Psi = (\psi_1,\ldots,\psi_n)$ comprise an orthonormal basis in $\mathbf{R}^n$, ξ can be rewritten in the form $\xi = a_1\psi_1 + \ldots + a_n\psi_n$, with $\|\xi\|^2 = a_1^2 + \ldots + a_n^2$. The $(\psi_1,\ldots,\psi_n)$ are called Principal Components. Vectors a and ξ are connected by the simple equation $a = \Psi^t \cdot \xi$. $\mathrm{Cov}(a) = \Psi^t \cdot \Sigma \cdot \Psi = \Lambda$ and this explains what we need the principal components for. In the new system of coordinates (the principal components) our random vector is uncorrelated. Moreover, we can reduce the dimensionality of our problem, if some eigenvalues equal zero or are very close to it, since the fraction of total variance explained by the first p dimensions of a equal $\frac{\sum_{i=1}^{p}\lambda_i}{\sum_{i=1}^{n}\lambda_i}$, which can be very close to 1. The problem of choosing p (stopping rule) is extensively explored in the literature. In /2/ several stopping rules are discussed, including the mentioned fraction of total variation explained. This rule tends to overestimate the number of nontrivial dimensions, but as we will see later, we must keep some small eigenvalues and corresponding eigenvectors for classification purposes.

In applied statistical analysis, the covariance matrix is usually unknown and replaced by its estimate. The popular choice of this estimate is the sample covariance matrix. Let $\mathbf{x}_1,\ldots, \mathbf{x}_N$ denote independent realizations of the random vector ξ; then

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \qquad 1.1$$

and

$$\mathbf{S} = \frac{1}{N-1}\sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})\cdot(\mathbf{x}_i - \bar{\mathbf{x}})^t \qquad 1.2$$

are unbiased estimates of the mean vector and the covariance matrix. Now we can apply eigenvector decomposition to the matrix S. The principal components are $\Psi^t \mathbf{x}_1,\ldots,\Psi^t \mathbf{x}_N$.

In addition to the known problems of PCA, we have yet another one, the quality of estimates. It has been shown /3/ that the sample covariance matrix increases the spread of eigenvalues. It tends to overestimate the highest and to underestimate the lowest. Even if we had a positive definite covariance matrix, the sample covariance matrix will be singular if the number of samples is less than or equal to the number of dimensions.

In such cases, the larger eigenvalues and corresponding eigenvectors can be calculated using Lanczos' method /4/. An implementation of the method can be found in /5/.

PCA for the classification problem has two severe drawbacks. First, it is an unsupervised technique. This means that to all our errors in the sample covariance matrix estimate we add another one, that of combining several classes into one, classes with different mean vectors and different covariance matrices. It distorts the estimate further. Second, the larger the eigenvalue the more variance is explained by the corresponding component, which does not necessarily mean better classification. Moreover, in MR or IR spectroscopy, the first principal component is usually non-discriminating.

As a result of these drawbacks, PCA is usually combined with feature selection /47/. Of the first k principal components, the best p are selected using a supervised classifier that will be described in the following part of Chapter 1.

1.2 Discriminant analysis

The purpose of classification or discriminant analysis is to assign objects to one of q classes, based on a set of measurements $\mathbf{x} = (x_1,...,x_p)$ obtained for each object. We will suppose that the k-th class can be described by the normal distribution with mean vector μ_k and covariance matrix Σ_k. Let us also assume that we know the unconditional prior probability π_k. The probability density function of the k-th class is given by

$$p_k(\mathbf{x}) = (2\cdot\pi)^{-p/2}\cdot|\Sigma_k|^{-1/2}\cdot\exp\left[-1/2\cdot(\mathbf{x}-\mu_k)^t\cdot\Sigma_k^{-1}\cdot(\mathbf{x}-\mu_k)\right] \qquad 1.3$$

and the classification rule is: choose $\hat{k}$ such that

$$p_{\hat{k}}(\mathbf{x})\cdot\pi_{\hat{k}} = \max_{1\le k\le q} p_k(\mathbf{x})\cdot\pi_k \qquad 1.4$$

1.3 and 1.4 lead to the classification rule

$$d_{\hat{k}}(\mathbf{x}) = \min_{1\le k\le q} d_k(\mathbf{x}) \qquad 1.5$$

where

$$d_k(\mathbf{x}) = (\mathbf{x}-\mu_k)^t\cdot\Sigma_k^{-1}\cdot(\mathbf{x}-\mu_k) + \ln|\Sigma_k| - 2\ln\pi_k \qquad 1.6$$

Using this rule is called quadratic discriminant analysis (QDA). If we assume that the covariance matrix is the same for all classes, we end up with the classification rule for Linear Discriminant Analysis (LDA), where the rule is the same, but the discriminant function is simplified significantly:

$$d_k(\mathbf{x}) = \mu_k^{\tau}\cdot\Sigma^{-1}\cdot\mu_k - 2\cdot\mathbf{x}^{\tau}\cdot\Sigma^{-1}\cdot\mu_k - 2\ln\pi_k \qquad 1.7$$

Terms common to all classes were canceled.

A detailed description of QDA and LDA can be found in /6/.

In practical applications we have to solve some additional problems. First, we have to estimate the mean vectors and covariance matrices for each class, or the pooled covariance matrix in the case of LDA. The estimates 1.1 and 1.2 are usually adequate, assuming that summation is done over all the samples of the same class.

The pooled covariance matrix is a weighted sum of the sample covariance matrix of each class, with π_k used as the weights.

Another problem to solve is the reliability of classification. Lachenbruch suggested an elegant way to assess the reliability without high computing cost /7/. He suggested excluding each sample from the training set before classifying it, the so called leave-one-out (LOO) cross-validation approach. However, we do not need to recalculate our sample covariance matrix and (more importantly) invert it *ab initio* every time. We need to do it only once and afterwards update these matrices, with a cost which is comparable to the calculations of the *a posteriori* probabilities.

Since the same formulae are used to estimate the parameters in discriminant analysis as in PCA, the same problem persists. The estimate, despite being unbiased, still can be badly distorted. The consequences of such distortion are very severe now, because of matrix inversion. As it has been said already, we typically have more dimensions in our data than samples in the training set. Therefore the sample covariance matrix for the entire data set is unavoidably singular. One can use a generalized inverse in place of the nonexistent inverse matrix, but this creates problems of its own. First, to get a robust generalized inverse matrix one must estimate the rank of the matrix precisely. In /2/ it has been shown for PCA applications that even sophisticated methods tend to overestimate the rank. It creates only marginal problems in PCA, but can be intolerable in discriminant analysis, as we shall see later.

Even if the matrix is not singular, the small eigenvalues can be underestimated, and they can dominate the inverse matrix. This means that discrimination will be done, based on factors such as accuracy of calculations, order of the data etc., which are unrelated to the classification problem.

To prevent this from happening Friedman suggested an approach called Regularized Discriminant Analysis /9/. He introduces two new parametric estimates of the covariance matrix. Let $\mathbf{m}_k$ be the sample mean vector, $\mathbf{S}_k$ be the sample covariance matrix of the k-th class and $\mathbf{S}$ be the pooled covariance matrix. Then

$$\mathbf{S}_k(\lambda) = (1-\lambda)\cdot\mathbf{S}_k + \lambda\cdot\mathbf{S} \qquad 1.8$$

and

$$\mathbf{S}_k(\lambda,\gamma)=(1-\gamma)\cdot\mathbf{S}_k(\lambda)+\frac{\gamma}{n}\cdot tr(\mathbf{S}_k(\lambda))\cdot\mathbf{I}\ , \qquad 1.9$$

where $0\le\lambda\le 1, 0\le\gamma\le 1$, are these estimates.

The four corners defining the extremes of the λ,γ plane represent well-known classification procedures. The lower left corner $(\lambda=0,\gamma=0)$ represents QDA, the lower right corner $(\lambda=1,\gamma=0)$ represents LDA. The upper right corner $(\lambda=1,\gamma=1)$ corresponds to the nearest-means classifier: an observation is assigned to the class with the closest (in Euclidean distance) mean. The upper left corner $(\lambda=0,\gamma=1)$ represents a weighted version of that classifier, with the class weights inversely proportional to the average variance of the variable within the class.

A good pair of values λ,γ can be found based on the training set. After testing RDA both on simulated and real life data, Friedman concludes that RDA can increase classification accuracy dramatically. Another interesting conclusion of his analysis is that in many cases LDA outperforms QDA "on its own turf", i.e., with significantly different covariance matrices for the different classes. The reason behind such a result is that the individual class covariance matrices are estimated poorly, whereas the pooled covariance matrix is generally more reliable.

Friedman has used error counting as the objective function when choosing λ,γ. In /10/ some other candidates for the objective function are discussed. Their choice of "appreciation" (objective) function is

$$A=\sum_{i=1}^{N}a(\mathbf{x}_i) \qquad 1.10$$

where

$$a(\mathbf{x}_i)=\begin{cases}\frac{1}{2}\cdot\prod_{k=1,k\neq m}^{q}\left[P(\omega_m|\mathbf{x}_i)-P(\omega_k|\mathbf{x}_i)\right]+\frac{1}{2}, P(\omega_m|\mathbf{x}_i)>P(\omega_k|\mathbf{x}_i), \mathbf{x}_i\in\omega_m\\ -\frac{1}{2}\cdot\prod_{k=1,k\neq m}^{q}\left[P(\omega_m|\mathbf{x}_i)-P(\omega_k|\mathbf{x}_i)\right]+\frac{1}{2}, P(\omega_m|\mathbf{x}_i)>P(\omega_k|\mathbf{x}_i), \mathbf{x}_i\notin\omega_m\end{cases}$$

and $P(\omega_m|x_i)$ is the conditional probability of class m given the observation $\mathbf{x}_i$. This objective function will make classification as crisp as possible for the correctly predicted observations and as fuzzy as possible for the misclassified observations.

Yet another objective functions was suggested in /11/. In essence, the authors are minimizing

$$d(\lambda,\gamma)=\sum_{i=1}^{q}\sum_{j=1}^{N_i} d(\mathbf{x}_j^i,\lambda,\gamma) \qquad 1.11$$

where $d(\mathbf{x}_j^i,\lambda,\gamma)=(\mathbf{x}_j^i-\mathbf{m}_i)^T\cdot\mathbf{S}_i^{-1}(\lambda,\gamma)\cdot(\mathbf{x}_j^i-\mathbf{m}_i)+\ln|\mathbf{S}_i(\lambda,\gamma)|$ and $\mathbf{x}_j^i$ is the j-th observation from class i.

As we will see later, the problem of selecting the objective function for RDA optimization resembles that for feature selection.

Another interesting modification of the LDA was suggested in /44/. The authors assumed that each class is a Gaussian mixture with different subclasses differing only in theirs means, while having a common covariance matrix. Moreover, the covariance matrix is common for all the classes. To solve the problem the authors suggested a method resembling fuzzy clustering /46/. They initialize the numbers of subclasses and their centroids, then iteratively estimate the covariance matrix, *a posteriori* probabilities, the updated centroids, and the updated covariance matrix. Such an approach will depend heavily on the correct estimate of the number of subclasses. The authors use two strategies to obtain the estimates and starting values for the means, the covariance matrix and the cluster probabilities. These are either the k-means /46/ clustering algorithm or the LVQ /47/ algorithm.

The method performed favourably against a range of competitors the authors used for comparison. As does the LDA, it also allows to reduce number of features in a dataset.

The major problem with the method is that it is slow.

1.3 Genetic Algorithms

After the publication of Holland's book /12/, Genetic Algorithms (GA) started to conquer the world of optimization. They have been successfully implemented for many optimization problems. This is a rapidly evolving area of research and we can find in the literature descriptions of a wide variety of GAs, but in this thesis we will restrict ourselves to the classical variant.

Any GA application consists of three components: a method of encoding the real life problem in the GA's terms, an objective function, and a GA engine. The first two components tie the problem to the GA engine and the latter solves it. Since in our GA program we implement both a novel encoding and a new (for GA) objective function, we will not pay too much attention to these components in the literature review. We will concentrate instead on the GA engine.

Before stating Holland's schema theorem, we need some definitions. A classical GA is working with bit strings. A set of bit strings comprise the population. Let $\mathbf{s}_i = (s_i^1, ..., s_i^p)$ be a bit string or a binary vector of length p, Ω the population of bit strings, n the size of the population. The population is evolving with time, therefore $\Omega(t) = \{\mathbf{s}_i(t), i = 1,...,n\}$. As we have mentioned, the evaluation function is part of the GA. We will denote this function by **F**. The function operates on each member of the population (chromosome) and returns the fitness of that member.

In /13/ the following description of GA is given:

1. **Initialize a population of chromosomes.**
2. **Evaluate each chromosome in the population.**
3. **Create new chromosomes by mating current chromosomes; apply mutation and recombination as the parent chromosomes mate.**
4. **Delete members of the population to make room for the new chromosomes.**
5. **Evaluate the new chromosomes and insert them into the population.**
6. **If time is up, stop and return the best chromosome; if not, go to 3.**

Of course, this description is too general. We have to clarify some aspects and the clarification will give us the classical GA. First, we have to define the genetic operators, mutation and crossover.

Mutation requires a probability of mutation $\mathbf{p_m}$. Given this probability, we generate a random number uniformly distributed in $[0,1]$ for each bit in the chromosome. If the number is less or equal of $\mathbf{p_m}$, we mutate the bit. There are two different approaches to this process. We can either "flip" the bit, or generate the new value randomly. The actual mutation rate is less than $\mathbf{p_m}$ with the second approach.

The crossover operator requires two parent chromosomes and returns two children. We can illustrate it with the following scheme:

Crossover

Fig 1.1

After parents have been chosen, we choose a random position (uniformly distributed) in the chromosome and exchange the parts of chromosomes, as it shown above. Again, there is a probability $\mathbf{p}_c$ assigned to this operator, which is the probability of a crossover to occur. A typical value for $\mathbf{p}_c$ is 60-90%.

Now we discuss how to evaluate the population. The fitness function is a very important part of the GA. However, without preventive measures we can end up with a population in which each chromosome has almost the same fitness as the best one. This will essentially mean no selection pressure at all. To avoid this, a linear normalization is introduced, where the probability a chromosome to be selected for breeding depends on its rank rather than the fitness. We order the population by decreasing fitness, and replace the initial fitness function by the final one:

$$f(\mathbf{s}_i) = a - i \cdot b \qquad 1.12$$

with appropriate constants a and b. Of course, we must ensure that $a - n \cdot b \geq 0$. This linear normalization is a very important part of the classical GA, since the probability of a chromosome being chosen as a parent is proportional to its fitness.

Now we can discuss a very critical issue concerning any population: how to delete old chromosomes and insert new ones. The classical GA is very "cruel" and makes almost no exception. The entire population is replaced by its offspring at once. The only exception in the classical GA is the best solution. This strategy of keeping the best chromosome(s) intact is called elitism.

GA is inspired by natural evolution and its success. It cannot be called genetic without using the concept of the gene. The analog of a gene in GA is called schema. We will define schema as a string of three symbols:

$$\mathbf{h} = (h^1,\ldots,h^p) \in \{0,1,*\}^p \qquad 1.13$$

We will say that a binary string $\mathbf{s}$ satisfies a schema $\mathbf{h}$ if for each $1 \leq i \leq p$, either $h^i =$ '*' or $h^i = s^i$. For example, for $\mathbf{h} = (11*00)$ there are only two satisfying bit strings, namely

(11100) and (11000). We can pick all the chromosomes satisfying a given schema **h** out of the entire population. Sometimes this subpopulation is also called schema.

Let $m(\mathbf{h},t)$ be the number of the chromosomes in that subpopulation. Now we are ready to formulate the schema theorem:

$$m(\mathbf{h},t+1) \geq m(\mathbf{h},t)\cdot\frac{f(\mathbf{h})}{\bar{f}}\cdot\left[1-p_c\cdot\frac{\delta(\mathbf{h})}{p-1}-p_m\cdot o(\mathbf{h})\right] \qquad 1.14$$

where $o(\mathbf{h})$ is the order of the schema (the number of defining symbols, 0 or 1, in the schema), $\delta(\mathbf{h})$ is the defining length of the schema, i.e., the physical distance between the outermost defining symbols of the schema, $f(\mathbf{h})$ is the average fitness of the chromosomes satisfying the schema and $\bar{f}$ is the average fitness of the entire population. In the example above, $o(\mathbf{h}=(11*00)) = 4$, and $\delta(\mathbf{h}) = 5$. All other functions in 1.14 will be determined by the entire population.

This theorem is the cornerstone of the GA; it says in essence, that schemata with an above average fitness will grow in the population, provided that they are not diluted by crossovers and mutations. In short, the schema theorem guarantees the convergence of the GA.

However, it says nothing about the quality of the solution. Will it converge to the first local minimum, in case of a minimization problem, will it go a little bit further, or will it find a narrow global maximum? The schema theorem does not answer those questions. The clue to the quality of the solution lies in the diversity of the population. It will help to explore a high dimensional space using a relatively small population.

An interesting way to maintain diversity in the population is suggested in /14/. The authors proposed a modification of the original fitness function to prevent overcrowding near some good solution. They called it a sharing function.

Let the fitness function be f. We introduce the sharing radius σ_{sh}, and the distance between two chromosomes:

$$d(\mathbf{s}_i, \mathbf{s}_j) = \sqrt{\sum_{k=1}^{p} \left(s_i^k - s_j^k\right)^2} \qquad 1.15$$

The sharing function is defined by

$$\phi(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{sh}}\right)^{\alpha}, d < \sigma_{sh} \\ 0, \text{otherwise} \end{cases} \qquad 1.16$$

where α and σ_{sh} are predetermined constants. Then, the sharing-modified fitness function becomes:

$$f_{sh}(\mathbf{s}_i) = \frac{f(\mathbf{s}_i)}{\sum_{j=1}^{n} \phi\left(d(\mathbf{s}_i, \mathbf{s}_j)\right)} \qquad 1.17$$

Changing the control parameters α and σ_{sh} with time we can either prevent the stagnation of population at the cost of accuracy, or ignore the stagnation and reduce the sharing-modified fitness function to the original one.

In /15/ the authors suggested this strategy for the entire GA design. This means that at the first stage we are looking for the first apprpoximation with reduced chromosome size; as time passes, the chromosome size is increased and the fitness function is also refined. This approach is a direct analog to the simulated annealing method.

One of the implications of inequality 1.14 is that there are two competing tendencies in the population. On one hand, the selection process is trying to improve the fitness of our population. On the other hand, mutation and crossover are working against it. Actually, they are working independently of any fitness considerations. Since to improve the fitness if it is already high is quite improbable, they work against the fitness. However, the main purpose of the genetic operators is to give flexibility to the population. A proper combination of these two tendencies will determine the success of the GA implementation. This means that parameter optimization is an important part of any GA realization. This is, of course, problem-dependent. Many papers are devoted to this aspect of GA implementation. In /16/ the authors suggest a way to determine the

population size adaptively. The size depends on the schema fitness variance. The results are not particularly interesting for us, since they are specific for a certain GA architecture. In /17/ a nonlinear differential equation was suggested to optimize the GA parameters. One can even find GA applied to the GA parameter optimization. Again, most of these results are applicable only for specific GA architectures.

Another theoretical aspect of the GA is an assessment of its convergence. As we mentioned, the schemata theorem was the major step in that direction. In /36/ a Markov chain framework was suggested to explore the convergence of the GA. The authors have used a nonstationary GA, with the probability of mutation reducing asymptotically to zero via a schedule. They have obtained a bound on the probability of mutation to facilitate the convergence. The simulation results have shown that convergence for GA is faster than for simulated annealing.

Similar results were obtained in /35/, where the convergence of GA and simulated annealing was compared.

D.E. Goldberg in /33/ brings the well-developed techniques from simulated annealing into the GA area to prove the asymptotic convergence of GA.

As a logical development the authors of /34/ have suggested a hybrid of those two methods to solve some NP-hard problems.

1.4 Feature selection/extraction

J. Kittler in /47/ gives an overview of the feature selection/extraction methods. If the number of original attributes is relatively low, forward selection or backward elimination would be reasonable choices. These two method are closely related, except that in the former we start from the best attribute and expand the feature set by adding the attribute that improves the objective function the most, whereas for backward elimination we start with the entire set of attributes and eliminate attributes sequentially.

Of course, the first approach is much faster because it works with low-dimensional data. The second allows to monitor continuously the amount of information loss incurred.

Different objective functions were described in this review but all of them are monotonic on the set of subsets of the attributes. In that case a branch-and-bound algorithm obtains the best subset of attributes much faster than an exhaustive search.

Most of the feature extraction methods considered by J. Kittler are based on the Karhunen-Loeve expansion, which is based on the eigenvectors of the covariance matrix. As we already mentioned, this approach will not allow easy interpretation of the results.

In /18/ a simple GA was used to select the best subset of attributes out of the original set, best with respect to the known classification. An interesting aspect of its implementation is the objective function. The authors are minimizing the number of attributes to achieve a given accuracy of classification. A similar approach was used in /19/ for selecting an optimal subset of principal components. In /28/ the author exploits

the inherent parallelism of the GA applied to the feature selection. Two GA-based methods were also compared to a random search and proved to be superior.

Another area of application of the feature selection/extraction methods is the regression problem. In /31/ many aspects of the subset selection in regression are covered. There is a significant overlap in the methods used in the classification and the regression problem. The main difference is in the objective function, because the "probability of error" concept does not make sense in the regression problem. GA-based feature extraction methods /27,30/ are relatively new and were not mentioned in /31/. Both implementations use an encoding similar to the one in /18/, i.e., "1" in a chromosome represents a selected attribute, "0" represents an ignored attribute. This implies that they cannot work with the entire spectral range in most cases, because uncontrollably high number of attributes in the analysis will undermine the result's relevance.

The authors of /29/ demonstrate that the feature subset selection methods are susceptible to overfitting. They suggest the use of a "wrapper method", such as bootstrap or cross-validation, to estimate the prediction accuracy.

1.5. MR spectra, ideal and noisy

Magnetic resonance spectroscopy (MRS) is the primary field of application of our methods. Therefore we must describe these spectra in some detail and show the problems we are encountering.

The contemporary MRS is pulse spectroscopy. The sample is irradiated by a radio frequency pulse and then the same equipment that was used to generate the pulse is used to register the sample response. The ideal impulse response (free induction decay or FID) is a sum of decaying complex exponentials:

$$F(t) = \sum_{i=1}^{p} I_i \cdot \exp\left(-R_{2i} \cdot t + j \cdot \left(2\pi\nu_i \cdot t + \varphi_i\right)\right) \qquad 1.18$$

where I_i is the amplitude of the i-th resonance, ν_i is its chemical shift, R_{2i} is its transverse relaxation rate, and φ_i is its phase. It is expected that $\mathbf{I}_i$ reflects the amount of the compound present with chemical shift ν_i, which itself is determined by the chemical structure of that compound. The Fourier transform of 1.18 gives us a frequency response function

$$S(\nu) = \int_{-\infty}^{\infty} F(i) \cdot \exp(-j \cdot 2\pi\nu t) dt =$$

$$\sum_{i=1}^{p} I_i \cdot \frac{\exp(j \cdot \varphi_i)}{R_{2i}} \cdot \frac{1}{1 - j \cdot \left(\frac{2\pi(\nu_i - \nu)}{R_{2i}}\right)} \qquad 1.19$$

which is just a sum of phased Lorentzians.

If one would like to compare the relative concentrations of two compounds, one isolates the corresponding peaks in the spectrum and integrates the real part of 1.19 in the vicinity of the peaks. But this is ideal and unrealistic. First, white noise is present in the spectrum. It can have different sources, due to the sample, electronic noise, or something

else. To increase the signal to noise ratio (SNR), the MR spectrum of the same sample is obtained repeatedly. The SNR increases $\sqrt{n}$ -fold with n repetitions.

As we have mentioned already, the same equipment is used to generate the pulse and to record the response. Therefore, after the pulse we must wait until all residual oscillations have decayed (dead time), losing the first and most informative points of the FID, thus losing in SNR.

The dead time has another implication for the spectra, i.e., phase shift. Since we have lost the first points of the FID, the real part of the spectrum appears as if each peak had its own phase. The simplest way to avoid the phasing problem is to work with magnitude spectra. This can be costly if one is trying to fit spectra, because for the real part the signal drops proportionally to $\frac{1}{(\nu-\nu_i)^2}$, whereas in magnitude only to $\frac{1}{|\nu-\nu_i|}$. Thus the accuracy of fitting the amplitude spectra suffers dramatically. For an attempt to compensate for the loss see /20/.

Another typical problem in MRS is the baseline. Again, baseline distortions can have different causes. One of them is the discreteness of our calculations /21/. Indeed, we do not have a continuous FID, we sample it over a finite period of time and carry out a discrete FT. As a result, we obtain a slightly different spectrum:

$$\hat{S}(\nu)=\sum_{i=1}^{p} A_i \cdot \frac{1-\exp\left(\left(j2\pi(\nu_i-\nu)-R_{2i}\right)\cdot T_{aq}\right)}{1-\exp\left(\left(j2\pi(\nu_i-\nu)-R_{2i}\right)/SW\right)} \qquad 1.20$$

where $A_i = I_i \cdot \exp\left(\left(j2\pi\nu_i - R_{2i}\right)T_{in} + j\varphi_i\right)$, T_{in} is the dead time, T_{aq} is an acquisition time, and SW is the sweep width, which is determined by the sampling rate. It can be shown that

$$\hat{S}(\nu) \cong S(\nu) + \sum_{i=1}^{p} A_i \qquad 1.21$$

thus giving a baseline. Unfortunately, the main reason for the baseline is not the data processing, but the hardware itself, and we cannot predict the shape of the baseline. In /22/ it is shown that the baseline can be eliminated simply by the appropriate choice of the sampling time, i.e., moving the initial point within one sampling interval can reduce baseline artifacts dramatically. We can also reformulate the last statement, i.e., that the inappropriate choice of the sampling points distorts the spectra dramatically. That paper was published in 1983, but we still have the very same problem in many experimental settings. Furthermore, there are problems even beyond hardware. Medical applications of MRS are complicated by the fact that the human body is mostly water. In proton spectroscopy, a spectrum of any human tissue will have just one peak, a water peak. All other information is present as tiny irregularities on the shoulders of the water peak. There exist experimental techniques to suppress the water signal; nevertheless, there is always some residual influence even in case of a successful implementation of those techniques.

Finally, the last problem. Due to fluctuation of hardware parameters or to individual preferences of the experimentalist the spectra may be linearly distorted, i.e., peak positions (chemical shifts) may be shifted with respect to the canonical ones. Such a distortion does not create any problem if one is analyzing an individual spectrum, but in the case of comparative spectral analysis it creates additional difficulties. This problem can be alleviated to some extent by aligning the spectra, but can not be eliminated completely, due to the presence of the white noise.

Since the different peaks in the spectra represent different metabolites or groups of metabolites, a logical way to extract the information from the spectra is to fit them by a sum of Lorentzians. The fitting can be done either in the frequency or in the time domain. In /39/ different methods of fitting were compared. Namely, HLSVD /40/, VARPRO /41/ and FITPLA /42/. The first two methods work in the time domain. HLSVD is a linear prediction method. It is very fast and requires minimal user interaction. The main

disadvantage of the method is its sensitivity to noise. This problem is aggravated in the typical MR experimental setting by the "deadtime".

HLSVD is not a maximum likelihood method. In contrast, the VARPRO (variable projection) method is a maximum likelihood method and is, in essence a nonlinear least squares fitting. The disadvantages of the VARPRO are the need for increased user-interaction and longer calculations times.

The last method is a nonlinear fitting based on the equation 1.20.

The comparison of these three methods has shown that they give similar results in demanding conditions with many overlapped peaks. They predict well the peak positions, but fail to estimate accurately the peak width /43/. Because of the possibility of baseline distirtions, this can lead to severe errors in the metabolite concentrations.

Thus arises the need to develop a method that allows us to extract relevant features from the spectra primarily for the purpose of classification (medical diagnosis). The method must retain the spectral identity of the features and must be robust enough to work with the highly variable and noisy spectra obtainable from human tissues and biofluids.

2. Preprocessing methods

This Chapter is devoted to the preprocessing methods we have used in our data analysis. The objective function used in these methods is given by

$$F = \sum_{c=1}^{q} \sum_{i=1}^{N} (p_{ic} - l_{ic})^2 \qquad 2.1$$

where p_{ic} is the probability that sample i belongs to class c, l_{ic} is a class indicator, i.e., 1 for the true class and 0 for all others. The term "fitness function" is commonly used in the GA literature, but other terms like "objective function" or "evaluation function" have their own niche /13,32,38/. Since we are going to use different feature selection/extraction methods, we have chosen a term, that is not connected to the GA specifically.

The objective function 2.1 is not monotonic on the set of feature subsets, and as a result the branch-and-bound algorithm not only will give a suboptimal solution, but can be misleading. Therefore, a different approach is used for feature selection.

These methods all use weighted LDA as the primary classification method. In the case of weighted discriminant analysis (each observation comes with its own weight reflecting its reliability or importance) IMSL's estimates /8/ are biased. Therefore, we had to derive unbiased estimates ourselves. The results are presented in Appendix A. This chapter will open with a brief description of the DP-based feature selection method. Subsequently, the GA-based algorithm will be described.

2.1. DP as a feature selection method

The Dynamic Programming (DP)-based feature selection was inspired by Viterbi's algorithm /23/. This is the reason it was named DP, being actually a forward selection algorithm.

We have a training set with known class labels and the following parameters: q is the number of classes, p is the number of attributes in the original data set. Our goal is to select a subset of attributes, which optimizes the objective function 2.1. For a fixed, desired number of attributes k we have $C_p^k = \frac{p!}{k! \cdot (p-k)!}$ possible combinations of attributes. Thus, exhaustive search is not feasible, even for moderate values of p. We have chosen a faster way of selecting a suboptimal subset of attributes.

On the first step of the algorithm, p sets of attributes are initialized by placing the individual attributes in corresponding sets ($\Omega_1 = \{1\}, \ldots \Omega_p = \{p\}$).

Now for i=2,...,k we repeat the same process, each Ω_i, i=1,p is updated by the attribute that gives the best result in combination with those already included in Ω_i.

This approach was implemented in combination with PCA, when a few principal components were selected out of those corresponding to positive eigenvalues, and gave promising results /24/. The method was also applied to the spectra themselves. However, first we had to reduce the feature space dimensionality by averaging several adjacent spectral points, since it is still impractical to use DP for 1000 attributes. The reduction had its own merit, since we had averaged out some noise. The main drawback of this approach is its inflexibility, i.e., we cannot move the border of the averaging window freely, they are fixed as soon the size of the window is determined.

This gave us the idea to use GA for the same purpose, where now we have more freedom to select the regions of interest.

2.2. GA-guided optimal attribute selection for spectra

Because our problem is special, we did not use the standard GA but designed and implemented a problem-specific version. There are two aspects of our GA implementation. 1) **Mapping** the original attribute space onto a bit string set. 2) Designing an overall scheme to create the **population** and allowing its evolution with subsequent generations. (The **objective function** F that drives the algorithm has already been defined.) Before we describe these in detail, we present a simple pseudo code for the overall operation of the algorithm:

1. Select M, the maximum number of desired attributes/subregions, G, the number of *generations*, P, the size of the *population*;

2. Create P binary strings of length L (P *chromosomes*), each containing M subregions, i.e., different sets of contiguous but non-overlapping *ones*; the remaining chromosome locations are filled with *zeroes*;

3. Process the M subregions to derive the M features;

4. For each of the P strings evaluate the previously defined fitness function F, applying an M-feature LDA/LOO classifier to the training set. Sort the P fitness values in ascending order (because of our definition of F, the lower its value the fitter the chromosome). In this sorted list the P_{elite} best *chromosomes* are referred to as the *elite*.

5. "Breed" the population by *mutation* and/or *crossover*. Steps 3-5 constitute one *generation*.

6. Go to step 3 and repeat until the number of generations equals the preset G.

Mapping

The problem of mapping the original feature space onto a bit string (zeroes and ones) has not received much attention in the literature. A possible reason is that the typical feature set for the standard GA application is a collection of numerical values, each represented by its own bit string in the computer. We have a somewhat different problem. A spectrum is a set of L different intensity values, one at each of L frequencies. The natural and simplest mapping onto a bit string is to put L "zeroes" into a logical array (a "chromosome") of L dimensions. Thus our initial input parameters are the *positions* of the spectral frequencies. Selecting subregions from the spectrum translates to converting some of the "zeroes" into "ones". However, because we plan to do more than simply eliminate a subset of the frequencies, we have extended the above simple representation. Thus, further dimension reduction is possible if in each subregion we replace the individual intensities by fewer attributes via additional processing. Such processing would create and associate a single attribute, e.g., the average value, or the variance, with each subregion.

To achieve such flexibility, we treat a spectrum as a set of segments; each comprised of a pair of adjacent data points. Hence, an L-point spectrum becomes an (L-1)-point "chromosome". Now a "one" in the i th position means that points i and $i+1$ are

connected. A "zero" means the opposite. A set of consecutive "ones" corresponds to an attribute range. Therefore, any given chromosome is a combination of a set of connected spectral subregions, and unconnected spectral points.

This is illustrated in Fig 2.1 below. The chromosome 10011100110 is encoded into three spectral regions (positions 1-2, 4-7, 9-11). The 12-point spectra would be replaced by three attributes, one per region. For example, the points 1-2, 4-7, 9-11 could be replaced by their averages, but any other function of the original attributes in these ranges could be used.

Encoding of the spectral points

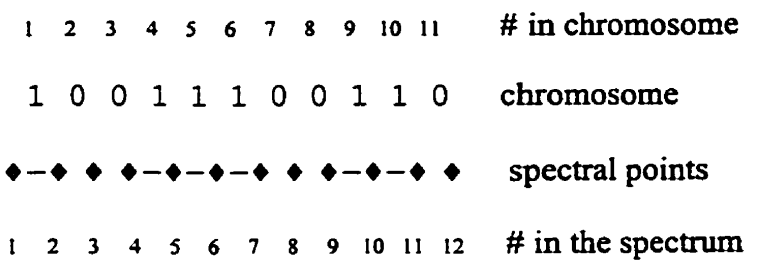


Fig 2.1

With the above construction we have two opportunities to reduce dimensionality: by selecting subregions, and by further processing (by some transformation) the data points within the subregions. If this transformation is averaging, then the influence of noise is also decreased.

The GA Architecture

We now introduce the genetic operators we use, and the details of how the population is created for any given generation.

There are two canonical genetic operators, mutation and crossover. Our implementation of *crossover* is the same as that described in the literature. We select two parents randomly from the current population, and choose a random crossover point at which the chromosomes are to be cleaved. Exchanging the parts of the parent chromosomes creates two new chromosomes. This operation makes GA very flexible and enables the search to move far away from the initial locations in the high dimensional feature space. The other genetic operator, *mutation*, is specifically tailored to our problem. Its distinguishing characteristics are that it is not a single-point operator, and that its size evolves with evolving generations. Extensive experimentation showed that in early generations we must change more than one bit in our chromosomes, hence the introduction of a k-block mutation. A large mutation size (large k) allows a rapid but coarse-grained initial exploration of the feature space, but as the process evolves, we need finer tuning, hence the size k of the k-block mutation is decreasing with increasing generations. The initial size of point mutation is 1/64th of the full spectral range, i.e., k = L/64, and k is decreased gradually as the optimization proceeds.

The input parameter set consists of the size P of the population, the maximum number of subregions allowed, the number of generations G, and the mutation p_m and crossover p_c probabilities.

To help create an initial population that is representative of the data, we build a *separability profile* using the t^2 statistic. For two-class problems this computed statistic is stored for each spectral position. For many-class problems all pairwise statistics are computed and the profile produced as their sum. Then the initial subregions are chosen randomly but with relative probabilities proportional to the profile.

The algorithm starts by generating a population of P random bit strings according to the above initialization. A *generation* consists of 1) changing members of the population by the above outlined *mutation* and *crossover* rules ("breeding"), 2) sorting the entire population according to their fitness (F) values. The best P_1 strings (the elite) are kept intact, the rest, $P - P_1$, is produced randomly by a process governed by the assigned mutation and crossover probabilities, and by the current fitness ranking. Two random chromosomes are chosen, with probabilities depending linearly on the rank of the chromosomes in the ordered generation list. They are first mutated with probability p_m, and either mated via crossover with probability p_c, or added directly to the new generation. This process is continued until a new generation is formed. We stop when either the size of point mutation is zero or the pre-set number of generations G has been reached.

Of course there are some drawbacks to this scheme. First, the best fitted strings are not changed, i.e. we do not explore the vicinity of the best solutions; second, there is danger of stagnation, when the entire population consists of copies of the same string. However, if we solve the second problem, we also solve the first. Indeed, in the evolving process, the best solutions will become next best on the subsequent generations, therefore will be subject to change in some way or another. Moreover, the best solutions usually have "close relatives" beyond the elite, therefore stagnation is the only real problem. There

are different solutions to this problem. The most elegant is the sharing function introduced in /14/.

We have chosen a very simple and primitive way of preventing stagnation. We do not allow two copies of the same string in the elite group. This means, that after sorting, we replace all the duplicates in the elite by the next best chromosomes.

Our mapping allows a huge number of potential features for classification, up to half the size of the full spectrum. However, any classifier that uses many attributes will be unstable and slow. Therefore, in our GA implementation we restrict the maximum number of features allowed, treating it as an input parameter under user control. All the regions in a chromosome are sorted according to size, distance to the neighbours, and other tie-breaking criteria and only best of them are used.

The question arises immediately, what to do with those regions that were not chosen. One solution is to eliminate them from a chromosome altogether (to zero all the corresponding positions). However, another option is more interesting. We can keep the unselected regions in the chromosome and allow mutations to take place there, thus imitating life. Indeed, in "real life" evolution, the silent mutations play a very important role, especially in a changing environment, preparing some species for extinction and others for prosperity. Currently, only the first way is implemented, but it will be interesting to compare the two in terms of time required to achieve the same fitness.

3. Algorithm testing and tuning.

This chapter is devoted to testing the methods described previously. To test the algorithm thoroughly we have designed artificial datasets. The design will be described in detail at the beginning of the chapter 3.

One of the biggest problems we encountered during the testing is the predictive power of the selection process. To illustrate what can happen, let us consider two identical classes of spectra. This means that each spectral position has identical distribution for both classes. If we had known the parameters of the distributions, we could conclude that the classes are undistinguishable. Unfortunately, in a typical situation the parameters in question are unknown and estimated based on the training set. Of course, the estimated parameters are not equal anymore. Since we are averaging spectral points in some region, we can expect that the estimation error will be less. It will be so for most of the regions. However, for some the estimation error may be correlated and we can get well-defined "features" for theoretically identical classes. The selection, guided by the LDA performance, guarantees that such regions will be found. When true discriminating features exist, the false ones will not present serious danger, however they can still influence the predictive power of the entire selection process.

We shall explore this problem, which we call overfitting, in greater detail in this chapter. We shall estimate the severity of the problem based on the dependence of the results for the training and test sets.

It is known that the probability of error in LDA is determined by the Mahalonobis distance d_{ij} between i-th and j-th class centroids /45/. Let us assume that there is a minimal value d_{min} that implies a pair of classes is different. Then, for a 2-class problem we must satisfy the following inequality: $d_{12} > d_{min}$. For a 3-class problem we must satisfy three inequalities: $d_{12} > d_{min}$, $d_{13} > d_{min}$, and $d_{23} > d_{min}$. Of course, these inequalities are not independent, and some of our assumptions are unnecessarily strong, but this reasoning suggests that the 3-class problem is less vulnerable with regard to overfitting.

After we have stated the problem of overfitting and explored it for both artificial and real life data, we will try to estimate the optimal parameters for our methods, taking the overfitting into account.

Ways to prevent overfitting or minimize its severity will be discussed in the Conclusions.

3.1. Artificial data sets.

Artificial data sets were created to test the GA-based attribute selection. The data sets were designed to imitate typical problems in MR spectroscopy. We have started from a two-class data set, but a three-class data set was also generated, since in classification a two-class problem has very often its own flavor and differs from many-class problems.

For each problem a set of peak heights was generated, and all the modifications of the spectral data were based on the same peak heights. The peak heights were normally and independently distributed around a given mean value h, the variance was chosen to create approximately 5% of class overlapping. To separate the classes, some designated peaks have the mean value shifted (h+d or h-d for different classes). The designation is done according the following table (the values in the table are the multipliers of d by which the mean value is shifted from h):

Table 3.1 Design of the datasets with peak heights.

		Peak Number
	Class	1 2 3 4 5 6 7 8 9 10
2-class problem	1	0 1 -1 0 1 1 0 -1 -1 0
	2	0 -1 1 0 -1 -1 0 1 1 0
3-class problem	1	0 1 -1 0 1 1 0 -1 -1 0
	2	0 1 -1 0 -1 -1 0 1 1 0
	3	0 -1 1 0 1 1 0 1 1 0

For both problems 10 peaks were used, 4 of which had identical distribution for all classes. Six others served to distinguish the classes. For the 3-class problem each pair of classes differed only in 4 peaks.

900 samples were generated, 300 for the training set and 600 for the test set, thus giving 100 samples per class in the training set for the 3-class problem and 150 for the 2-class one. The peak height data were saved and used to generate the spectral data, thus for each type of spectral noise the same underlying peak heights data were used, providing that our analysis is concerned the spectral noise only, not the fluctuations in the underlying peak heights data (the latter imitates the "real life" variety within a given class).

The Kolmogorov-Smirnov (KS) /48/ test was used to confirm the correspondence of the generated sets to design criteria. Table 3.2 summarizes results for the 2-class problem

Table 3.2 Kolmogorov-Smirnov test for the 2-class problem (p-values)

Peak	Training set Class1 ↔ Class 2	Class 1 Train set ↔ Test set	Class 2 Train set ↔ Test set
1	p = n.s.	p = n.s.	p = n.s.
2	p < 0.001	p = n.s.	p = n.s.
3	p < 0.001	p = n.s.	p = n.s.
4	p = n.s.	p = n.s.	p = n.s.
5	p < 0.001	p = n.s.	p = n.s.
6	p < 0.001	p < 0.05	p = n.s.
7	p = n.s.	p = n.s.	p = n.s.
8	p < 0.001	p = n.s.	p = n.s.
9	p < 0.001	p = n.s.	p = n.s.
10	p = n.s.	p = n.s.	p = n.s.

n.s. stands for not significant, i.e.; the difference in distribution is not statistically significant. The KS test confirms that all peaks, except one were generated correctly, i.e., that in the training set classes are separated very well for those peaks which were selected to be discriminative. The irrelevant peaks have identical distribution for both classes. For the second class peaks are distributed identically for the training and test sets. The same is true for all but the 6th peak of class 1. Due to sampling fluctuations arising from the use of random samples the 6th peak is distributed differently for the test set, compared to the training set. Nevertheless, this difference is much less significant than the between class difference and the test data of different classes are still very well separated for this peak (data not provided).

The following table summarizes results for the 3-class problem:

Table 3.3 Kolmogorov-Smirnov test for the 3-class problem

Peak	Training set			Training set ↔ Test set		
	1 ↔ 2	1 ↔ 3	2 ↔ 3	Class 1	Class 2	Class 3
1	p = n.s.	p = n.s.	p = n.s.	p = n.s.	p = n.s.	p = n.s.
2	p < 0.05	p < 0.001	p < 0.001	p < 0.05	p = n.s.	p = n.s.
3	p = n.s.	p < 0.001	p < 0.001	p = n.s.	p = n.s.	p = n.s.
4	p = n.s.	p = n.s.	p = n.s.	p = n.s.	p = n.s.	p = n.s.
5	p < 0.001	p = n.s.	p < 0.001	p = n.s.	p = n.s.	p = n.s.
6	p < 0.001	p = n.s.	p < 0.001	p = n.s.	p = n.s.	p = n.s.
7	p = n.s.	p = n.s.	p = n.s.	p = n.s.	p = n.s.	p = n.s.
8	p < 0.001	p < 0.001	p = n.s.	p = n.s.	p = n.s.	p = n.s.
9	p < 0.001	p < 0.001	p = n.s.	p = n.s.	p = n.s.	p = n.s.
10	p = n.s.	p = n.s.	p = n.s.	p = n.s.	p = n.s.	p = n.s.

Again 9 peaks out of 10 were generated according to specifications, the only exception for the 3-class problem is the second peak, and again we can ignore the fluctuation.

After the satisfactory peak heights generation, we switch to generating the spectra.

The spectral data (magnitude spectra) were generated according to 1.20. Since the peak width was chosen to be identical for all the peaks and the phase can be ignored for the magnitude spectra, the ideal spectra were just linear combinations of the peak heights.

Each spectrum contained 1024 points, the peak positions were distributed uniformly along the spectral range.

In addition to the ideal noise-free spectra, we have considered three types of "noise" we have mentioned in chapter 1. The white noise was modeled by adding a normally distributed random number (with zero mean and a given σ) to the each spectral point. To imitate a randomized peak position for any given spectrum we have replaced by a linear function **a+b*x** the canonical peak position **x**, where **a** and **b** were generated randomly for each spectrum. Baseline was modeled by the sum of two random, small, but very broad peaks. Fig. 3.1 illustrates this for a particular spectrum. In addition to the magnitudes of the noise free spectrum, baseline, and white noise, the magnitude of the difference between the noise free spectrum and one with randomized peak positions is also displayed. We can see readily that this particular kind of noise can be the largest, but it is localized. One comment regarding the baseline simulation. It creates bigger distortion than it appears, since only its magnitude was plotted. The "wave height" is much bigger for the real or imaginary part of the baseline.

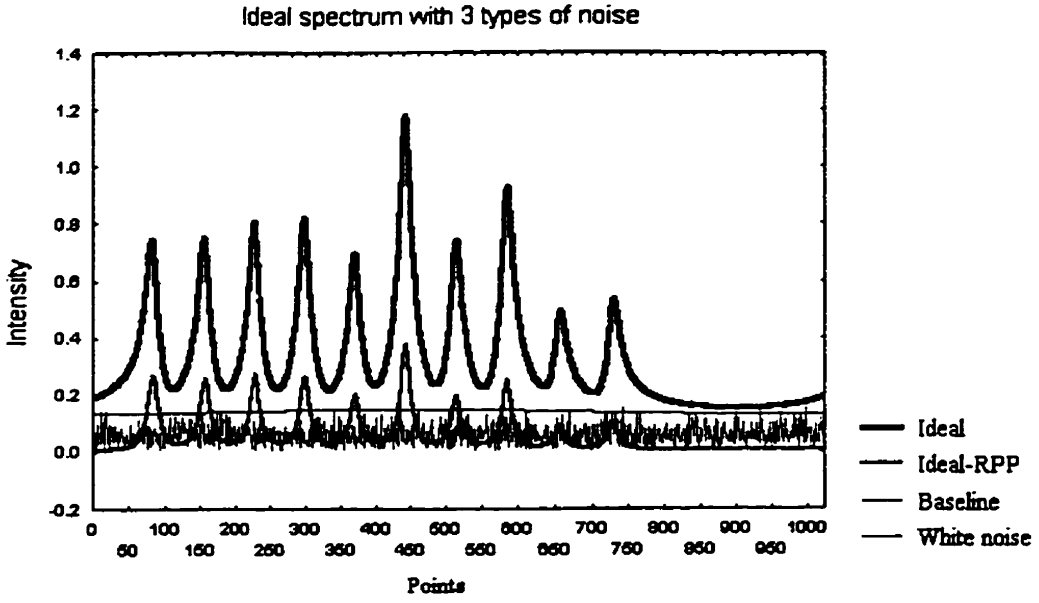


Fig 3.1

The data were generated separately for each type of distortion as well as for some combinations, thus giving us 7 types of spectral data:

1) noise-free spectra

2) white noise (wn)

3) randomized peak position (rpp)

4) random baseline (rb)

5) wn + rpp

6) wn + rb

7) wn + rb + rpp

To summarize:

- Despite many simplifications, the data were generated so that even for the noise-free spectra the classes had a significant overlap.
- We know the number of features in the data and where the features are located.
- The classification of the peak height data gives us a gold standard for the spectral data analysis.
- The test set is twice as large as the training set, allowing for reliable results.

Given this "tool", we can explore the behavior of our feature selection/extraction methods, optimize their parameters, and answer the main question: are the methods adequate for the problems we encounter.

All random numbers were generated using the functions genunf and gennor of the freeware statistical library RANLIB. Normally, the seed value was not changed. To achieve the same effect, the corresponding modules of the program were executed consequently within one run of the program, thus a new run will start with a seed values which were left from the previous run. This provides with a completely reproducible results.

3.2. Overfitting with GA-ORS for artificial data sets

Before parameter optimization, we must decide what criterion to use. The most natural candidate is the outcome of the feature selection/extraction. Unfortunately, because of the possibility of overfitting, this is not advisable. To show this we have calculated mean squared error (MSE) not only for the training set (MSE_{tr}) but also for the test set (MSE_{ts}).

We plot the MSE_{ts} (y axis) against MSE_{tr} (x axis) in Fig. 3.2.

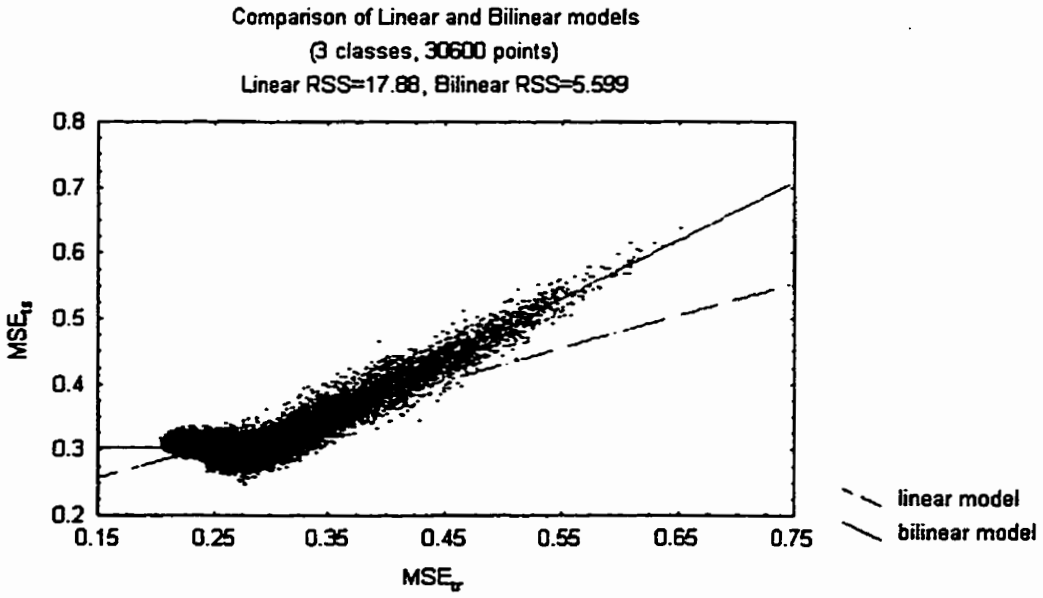


Fig. 3.2

As can be seen linear regression is not the best model to describe the dependence. As an alternative, we decided to use a bilinear regression (also shown on the plot), where one of the lines is a constant. The detailed mathematical description of the model is given in Appendix B. Essentially, this model implies that after some critical value, any further improvement in the training set does not do improve the test set. If we were to use a more complicated bilinear model, allowing an arbitrary slope for both lines, we would

probably get negative correlation for the low values of MSE_{tr}. This would mean that the further we improve MSE_{tr}, the worse MSE_{ts} would become. However, for this work the simpler model is sufficient.

In this chapter we shall attempt to show that our model describes better the dependence between MSE_{tr} and MSE_{ts}. Thus we'll assume that:
$MSE_{ts} = y_0 + a\,(MSE_{tr} - x_0) + e$, where e is assumed to be a normally distributed error term.

We'll start with the artificial data sets. As can be seen in Fig. 3.2, for any model MSE_{ts} has significant variance for similar values of MSE_{tr}. Therefore, we cannot base our analysis on a single run of the feature selection/extraction process. To reduce the variance, 10 runs for the same dataset and same parameters were used. The only difference was in the seed value for the random number generator. Each run consisted of 50 generations, with population size equal to 300. Taking into account the initial generation, we get $51 \cdot 300 = 15300$ datapoints for our analysis. We required to find the 10 best regions and ran LDA in the 10-dimensional space to get each point. Multiply this by 7 different types of noise generation and two types of problem (2-class and 3-class).

Thus, more than 10 runs for each dataset was found to be too timeconsuming. The only solution was to use robust statistics, namely median and lower/upper quartiles, because mean value and variance are too sensitive to outliers. The statistics were computed in Statistica 4.3.

To compare different models we have used three numbers: the total variance of MSE_{ts} (TotalSS) and residual sums of squares for the linear regression (LR-RSS) and for the bilinear regression (BR-RSS). The results are given in the Tables 3.4 and 3.5.

Table 3.4. Total Sum of Squares and Residual Sums of Squares for the 2-class problem

	TotalSS			LR-RSS/TotalSS			BR-RSS/LR-RSS		
Noise type	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile
No noise	.1880	.1590	.2018	.5795	.4722	.6179	.6989	.6458	.7101
wn	.9630	.6569	1.2957	.8646	.5961	.9543	.9057	.8088	1.0000
rpp	14.2886	12.5986	15.4808	.1318	.1067	.1626	.9091	.6211	.9724
baseline	3.3353	2.9463	3.5232	.1472	.0987	.1717	.8894	.7855	.9283
wn+rpp	11.1840	9.9458	12.4846	.2518	.1520	.2781	.4736	.3944	.5280
wn+b	3.6551	3.5495	3.9306	.3492	.2844	.3931	.4773	.4245	.5177
wn+rpp+b	17.4901	14.8459	18.6228	.1341	.0773	.1965	.3656	.3259	.4725

The first conclusion we can draw from the Table 3.4 is that randomization of peak positions dramatically increases the variance of MSE_{ts}. This is an interesting observation and we shall discuss it later. For now, we are more interested in the quality of both the linear regression and the bilinear regression models.

For noise-free data neither linear nor bilinear regression work well, nethertheless the bilinear regression reduces the residual sum of squares by approximately one third compared to the linear regression. This is confirmed by the fact that correlation between MSE_{tr} and MSE_{ts} at the linear level is only 70% (see *a* in table 3.5).

We have obtained very peculiar results when only white noise was added to the data. This peculiarity is better seen from table 3.5. Out of ten runs 4 gave negative value for *a*; in essence this means that practically all the points were in the overfit area and that the dependence between MSE_{tr} and MSE_{ts} in this region is negative, i.e., the smaller MSE_{tr} the larger MSE_{ts} grows. Thus the general bilinear model would probably describe the dependence better.

Table 3.5. Parameters of the bilinear model for the 2-class problem

	y_0			a			x_0		
	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile
noise free	.0368	.0361	.0373	.6907	.6600	.7408	.0450	.0437	.0468
wn	.0590	.0443	.0607	.3884	-.2343	.4974	.0287	.0050	.0460
rpp	.0486	.0416	.0539	.9435	.9092	.9606	.0386	.0293	.0466
baseline	.0378	.0369	.0404	.8894	.8455	.9243	.0440	.0403	.0467
wn+rpp	.0646	.0610	.0662	.8456	.8292	.8563	.0593	.0493	.0610
wn+b	.0696	.0666	.0728	.8809	.8241	.8871	.0492	.0476	.0519
wn+rpp+b	.1041	.0948	.1129	.8545	.8335	.8637	.0808	.0644	.0893

The other cases of a single type of noise are described quite well by a linear model (more than 85% of the total variance was explained by the model), nevertheless the bilinear model gave on average ("on median") 10% improvement.

Results for the combined noise are even more impressive. Despite a moderate to good fit with a linear model, the bilinear model significantly improved this. The residual sum of squares was reduced by a factor of 3. Thus our bilinear model explains the dependence much better for the data with complex "noise". This could be partially due to the fact that data with a single type of noise are either too easy to optimize and we get to the overfit area very soon, or too hard to optimize and we get only a few points in the overfit area. Parameters of the bilinear model are also more stable for the complex "noise". This conclusion is very important since we are going to use y_0 as the main factor in the GA_ORS parameter optimization.

Results for the 3-class problem given below in Tables 3.6 and 3.7 confirm these conclusions. Again, randomization of the peak positions increased the variance of MSE_{ts}.

White noise did not create such a severe case of overfitting as for the 2-class problem and the dependence in this case is explained quite well by a bilinear regression model. The only dataset for which bilinear regression did not give significant improvement was the one with randomized peak positions. Combined with the fact that the linear model is working very well, this "abnormal" behavior can be easily explained by the lack of overfitting.

As for the 2-class problem, most complex data with all three types of noise added are explained much better by the bilinear model. The residual sum of squares decreased threefold compared to the linear model.

Table 3.6. Total Sum of Squares and residual Sums of squares for the 3-class problem

	TotalSS			LR-RSS/TotalSS			BR-RSS/LR-RSS		
Noise type	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile
No noise	1.6464	1.3409	1.8741	.6374	.5311	.6968	.5447	.4219	.5747
wn	3.1857	2.6937	3.4030	.8909	.7643	.9655	.5369	.4944	.5670
rpp	36.9197	29.3748	38.8872	.0418	.0370	.0485	.9613	.9111	.9708
baseline	12.0569	9.3862	13.9694	.1056	.0819	.1581	.6106	.5451	.8400
wn+rpp	25.1771	23.0065	26.6477	.1453	.0875	.1640	.6045	.4328	.6734
wn+b	9.6476	8.2257	13.2014	.2644	.1638	.2742	.5439	.3800	.5841
wn+rpp+b	25.6281	22.9355	27.3072	.2214	.1255	.2671	.3639	.2837	.4998

The parameters of the bilinear model for the 3-class model are also more stable, with all but two quartiles for y_0 within 5% of the median. These two exceptions are the lower quartiles for the "wn+rpp" (8%) and "wn+b" (6%) datasets. For x_0 there are three such exceptions: the "baseline" (7.4%), the "wn+rpp" (7.6%) and the "wn+b" (6.6%).

Table 3.7. Parameters of the bilinear model for the 3-class problem

	y_0			a			x_0		
	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile
noise free	.1754	.1738	.1771	1.0489	.9055	1.0718	.2170	.2077	.2223
wn	.1987	.1932	.2014	.9548	.8952	.9790	.2200	.2130	.2217
rpp	.1787	.1717	.1867	1.0092	.9530	1.0289	.1978	.1956	.2025
baseline	.1827	.1744	.1887	1.0752	1.0135	1.0834	.2222	.2058	.2255
wn+rpp	.2216	.2037	.2243	.8331	.7748	.8586	.2081	.1923	.2137
wn+b	.2174	.2047	.2249	.8700	.7749	.8840	.2184	.2039	.2240
wn+rpp+b	.2953	.2864	.2991	.8839	.8373	.8949	.2899	.2752	.2953

We have shown in this chapter that for the artificial data with complex noise the bilinear model is superior to the linear one. It greatly reduces the residual sum of squres, and has stable parameters. One of those parameters (x_0) is essentially the mean of the minimal value of MSE_{ts}. There is no sense to continue optimization once $MSE_{tr} \leq x_0$, since further improvement in MSE_{ts} is unlikely.

3.3. Overfitting with DP for artificial data

Dynamic programming requires some modifications to the described scheme to explore the possibility of overfitting. First, we cannot afford to start with 1024 variables, therefore the datasets were condensed to 128 attributes each, by averaging each 8 adjacent points. Second, subsets of variables have been chosen in a deterministic way. As a result, each run of the program gives identical results, unless the parameters are changed. To overcome the second obstacle we have created 8 condensed datasets instead of one using each time a different starting point. Thus we got 8 points to estimate each parameter in question.

For a fair comparison, the same number (10) of attributes were requested as in GA_ORS. MSE_{tr} and MSE_{ts} were recorded for all considered combinations of ten attributes and were used to estimate the model parameters. The results for the 2-class problem are given in the Tables 3.8 and 3.9.

As we can see from Table 3.8, the bilinear regression model does not give any significant improvement over the linear one. Moreover, the linear regression model practically does not change the total sum of squares. This means that there is no correlation between MSE_{tr} and MSE_{ts} i.e., all our points fell into the overfit area.

This conclusion is confirmed by parameter a in Table 3.9. It is either extremely unstable, changing from negative to large positive values, or is very stable but close to zero, as it happened for the noise-free data. There are two exceptions (the "rpp" and "baseline" datasets), but the fact that the linear model only marginally decreased the sum

of squares tells us that we have only few points in the linear area, all the others falling into the overfit area.

Table 3.8. Total Sum of Squares and Residual Sums of Squares for the 2-class problem

	TotalSS			LR-RSS/TotalSS			BR-RSS/LR-RSS		
Noise type	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile
noise free	.0566	.0458	.0634	.9868	.9779	.9918	.9979	.9972	.9994
wn	.4711	.3615	.7505	.9550	.9160	.9860	.9934	.9425	.9988
rpp	2.2781	2.2387	2.3878	.9134	.8405	.9685	.9956	.9887	.9994
baseline	.2836	.2700	.2973	.9341	.9246	.9651	.9778	.9696	.9898
wn+rpp	1.1184	1.0453	1.9376	.9342	.9015	.9657	.9956	.9570	1.0000
wn+b	.7528	.5673	1.3259	.9778	.9498	.9931	.9962	.9540	.9999
wn+rpp+b	1.5272	1.0476	2.7020	.8771	.8331	.9526	.9909	.9355	.9932

Table 3.9. Parameters of the bilinear model for the 2-class problem

	y_0			a			x_0		
	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile
noise free	.0373	.0373	.0375	-.1010	-.1116	-.0632	.0339	.0332	.0349
wn	.0597	.0553	.0653	-.3819	-.5174	1.3899	.0232	.0190	.0336
rpp	.0590	.0546	.0609	.6810	.5055	.8617	.0416	.0382	.0503
baseline	.0462	.0458	.0464	.5167	.4597	.7127	.0479	.0476	.0494
wn+rpp	.0810	.0786	.0893	.2607	-.3682	1.1218	.0579	.0383	.0600
wn+b	.0774	.0746	.0811	.3708	-.1675	.9912	.0487	.0314	.0526
wn+rpp+b	.1259	.1151	.1286	.0931	-.6437	.7635	.0729	.0664	.0779

Results for the 3-class problem (Tables 3.10 and 3.11) confirm the trend. Despite the relative stability of the a parameter, there is no improvement over the most trivial model $y = y_0$, the model which gives the total sum of squares as the residual sum of squares. In other words MSE_{ts} is essentially independent of MSE_{tr}.

Table 3.10. Total Sum of Squares and Residual Sums of Squares for the 3-class problem

	TotalSS			LR-RSS/TotalSS			BR-RSS/LR-RSS		
Noise type	Med.	Lower Quartile	Upper Quartile	Med.	Lower Quartile	Upper Quartile	Med.	Lower Quartile	Upper Quartile
noise free	.4415	.4163	.6682	.9852	.9455	.9907	1.0000	.9988	1.0000
wn	1.4863	.7482	2.5264	.9897	.9850	.9916	.9997	.9774	1.0000
rpp	2.5429	2.2219	2.8877	.6146	.4811	.7134	.9967	.9907	.9984
baseline	1.3942	1.1749	1.5921	.9079	.8536	.9375	.9873	.9812	.9973
wn+rpp	2.8399	2.4754	4.3573	.8859	.7496	.9501	.9662	.8955	1.0000
wn+b	3.1103	1.7541	4.4845	.9645	.8220	.9933	.9371	.8460	.9950
wn+rpp+b	3.1630	2.7083	4.3403	.9895	.9815	.9970	.9709	.9635	.9946

Table 3.11. Parameters of the bilinear model for the 3-class problem

	y_0			a			x_0		
	Med.	Lower Quartile	Upper Quartile	Med.	Lower Quartile	Upper Quartile	Med.	Lower Quartile	Upper Quartile
noise free	.1692	.1688	.1704	-.1681	-.2819	-.1119	.1766	.1751	.1819
wn	.2086	.2078	.2133	.0079	-.2182	.5045	.1819	.1776	.1901
rpp	.1937	.1891	.1958	.7400	.6945	.8734	.2140	.2110	.2161
baseline	.1801	.1773	.1818	.9718	.6823	1.4144	.2215	.2125	.2236
wn+rpp	.2325	.2316	.2371	.6315	.3031	1.2032	.2169	.2004	.2310
wn+b	.2349	.2294	.2432	1.1790	.3618	1.5646	.2333	.2259	.2363
wn+rpp+b	.3090	.3062	.3133	1.1564	.7754	1.9007	.3047	.2917	.3111

Unlike for the 2-class problem, we now have got very stable x_0, and what is even more impressive, its values for the different noise types are close to those obtained for the GA-based attribute extraction. This is promising, since it shows that the parameters of the bilinear model are intrinsic properties of the data set, not only of the method of attribute selection.

Thus we have shown that DP is also susceptible to overfitting, even to a greater degree than GA_ORS. This is due to the algorithm itself, since to obtain the 10th attribute only "good" combinations of 9 attributes were used. Therefore we come to the final step of algorithm with the burden of overfitting at the previous steps. Thus all points of a possible linear trend in the model can be eliminated even before we started to collect them. Unfortunately, we cannot start collecting at the previous steps, since the parameters of the model depend most likely on the number of requested attributes.

We have several ways to overcome the overfitting problem. One way (which shall be discussed later) is to reduce the number of features sought. The other two involve either increasing the number of samples or enlarging the averaging window. The former will never fail, but generally it is very expensive and therefore not always realistic. The latter will simply reduce the power of the method, since with a large window there is a good chance that a discriminative feature will be averaged with irrelevant information and will disappear.

3.4. Overfitting with real life data

Artificial data sets were a very convenient tool for the regression model comparisons, but without real life data our results could be attributed to the way the data were generated. Unfortunately, it is quite difficult to get real life data with a reliable test set, especially MR spectral data.

We have a data set with infrared (IR) spectra at our disposal. It has1362 IR spectra of blood samples belonging to 3 different classes. Each spectrum is 1816 points long. Classes are of different sizes, varying from 320 to 640 samples per class. Half the samples chosen randomly comprise the training set, the rest were included in the test set. Since the full data set did not show a tendency for overfitting, 3 other versions of the data were created with 50, 100, and 150 samples per class in the training set, selecting the samples randomly.

The results are given in the Tables 3.12 and 3.13. The bilinear regression model does not have any advantage over the linear one, but this could be for various reasons. Either there is no overfitting at all (all $MSE_{tr} > x_0$), or all points are in the overfit area (all $MSE_{tr} < x_0$). Analyzing the ratio LR-RSS/TotalSS we see that for the full data set and the one with 150 samples per class, there is no overfitting. The ratio is low, meaning that the linear regression model describes the dependence very well, a is large and stable.

When the number of samples was decreased to 100 per class, neither model worked well; with 50 samples per class, we were entirely in the overfit area. The parameter a is extremely unstable, switching from negative to positive values.

Table 3.12. Total Sum of Squares and Residual Sums of Squares for the IR data.

	TotalSS			LR-RSS/TotalSS			BR-RSS/LR-RSS		
Data sets	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile
full data	66.165	50.607	74.427	.1494	.1282	.1646	1.0000	.9993	1.0000
150	79.757	63.439	85.784	.2627	.2056	.2728	.9717	.9003	.9999
100	88.399	35.772	107.279	.4653	.2908	.5129	.9999	.9651	1.0000
50	20.8896	17.1616	33.4280	.8066	.7406	.8683	1.0000	.9933	1.0000

Table 3.13. Parameters of the bilinear model for the IR data.

	y_0			a			x_0		
Data sets	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile	Median	Lower Quartile	Upper Quartile
full data	.2069	.1825	.2131	1.3518	1.3081	1.3923	.1735	.1638	.1770
150	.2165	.2009	.2282	1.3349	1.2871	1.4486	.1225	.1134	.1261
100	.2685	.0000	.3055	1.4161	.7831	1.6675	.0926	-.0423	.1061
50	.5566	.0000	.5640	-.3756	-.4075	.4309	.06652	-.4787	.0751

We did not succeed in showing any advantage of the bilinear model for real life data, since practically all the points in the MSE_{ts}-MSE_{tr} plane were either in the overfit or in the linear area. The advantage of the bilinear regression model is best seen when we have a balanced set of points, i.e. with both the overfit and the linear range present significantly. Nevertheless we can still see the danger of overfitting. As soon as the training set became small enough (50 samples), the MSE_{ts} is not longer linearly related to MSE_{tr}, thus indicating that the predictive power of the algorithm was lost.

3.5. Parameter optimization for GA-ORS.

2-class problem

The bilinear regression model was introduced to find a way to optimize the parameters. GA-ORS has 4 major parameters defining the behavior of the algorithm. These parameters are: the number of generations (n), the population size (l), the probability of mutation (p_m), and the probability of crossover (p_c). Due to the extensive computations required, we cannot afford a 4-dimensional optimization, therefore the problem will be solved in two steps. First we find the "optimal" generation size and the number of generations; subsequently, the "optimal" probabilities will be obtained. The word optimal is in quotes since, due to the relatively large variance of the results, we cannot always differentiate reliably between two sets of parameters and only a few of the possible combinations could be explored.

On the first step of optimization, the population size was selected from the set {50, 100, 200, 300, 400, 500, 600}. For each value of the population size 10 runs of GA-ORS were carried out with 50 generations each. The probabilities were set to the normally recommended values p_m=0.001 and p_c=0.66.

Since GA-ORS is time consuming, the parameter optimization for artificial datasets was carried out with only one dataset, with all types of noise present.

MSE_{tr} for the 2-class problem is plotted in Fig. 3.3. For reference the median value and the quartiles of the x_0 parameter in the bilinear regression model are also

plotted. As we can see, 20 generations is enough for MSE_{tr} to go below the lower quartile of x_0 regardless of the population size. Unfortunately, x_0 is probably the least stable parameter of the model, so any conclusion based on it would be the least reliable. Nevertheless, even here we can see that l=200 is the best choice; further increasing l either made things worse (l=300) or gave insignificant improvement.

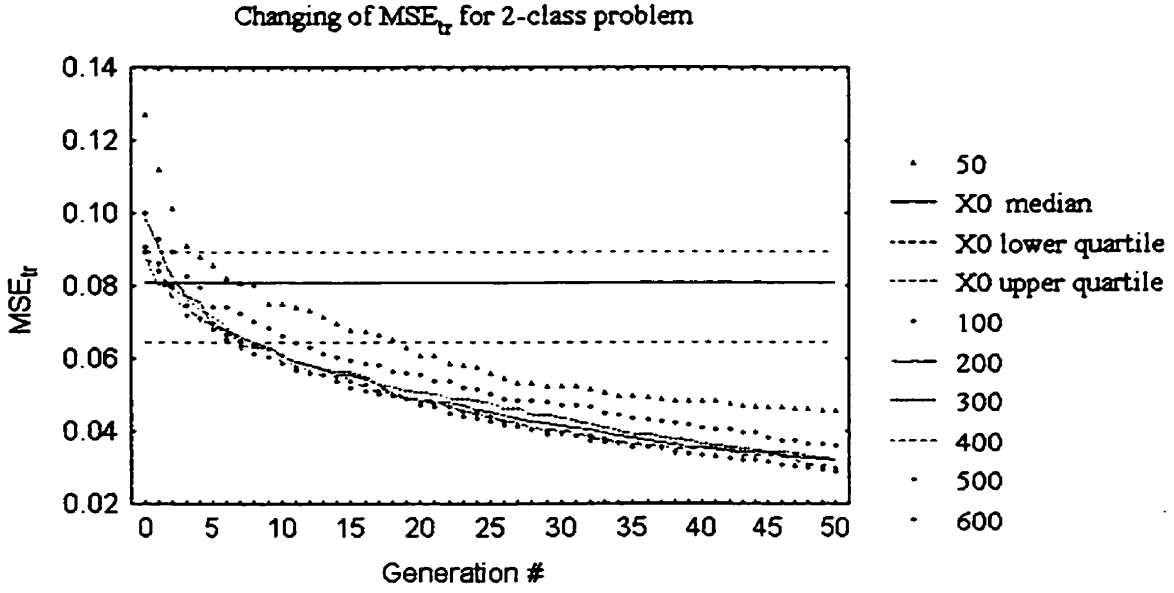


Fig. 3.3

Unlike x_0, y_0 is the most stable parameter of the model, and MSE_{ts} would be a better predictor for the quality of the resulting classifier. Unfortunately, as can be seen in Fig. 3.4, MSE_{ts} is very noisy. As a result, we must make our decision by exclusion. l=100 is unsatisfactory since the median of MSE_{ts} does not decrease below y_0, thus indicating underfitting. For l=50 it does go slightly below y_0, but this result is compromised by the one for l=100. l=500 and l=600 do not give improvement compared to l=200 or l=400, therefore these choices simply waste computational time. l=300 is definitely worse than l=200 and can be eliminated. Out of the remaining two values, surprisingly enough, l=400

requires more generations to achieve its minimum and this minimum is higher than for l=200. Thus l=200 is the only reasonable choice.

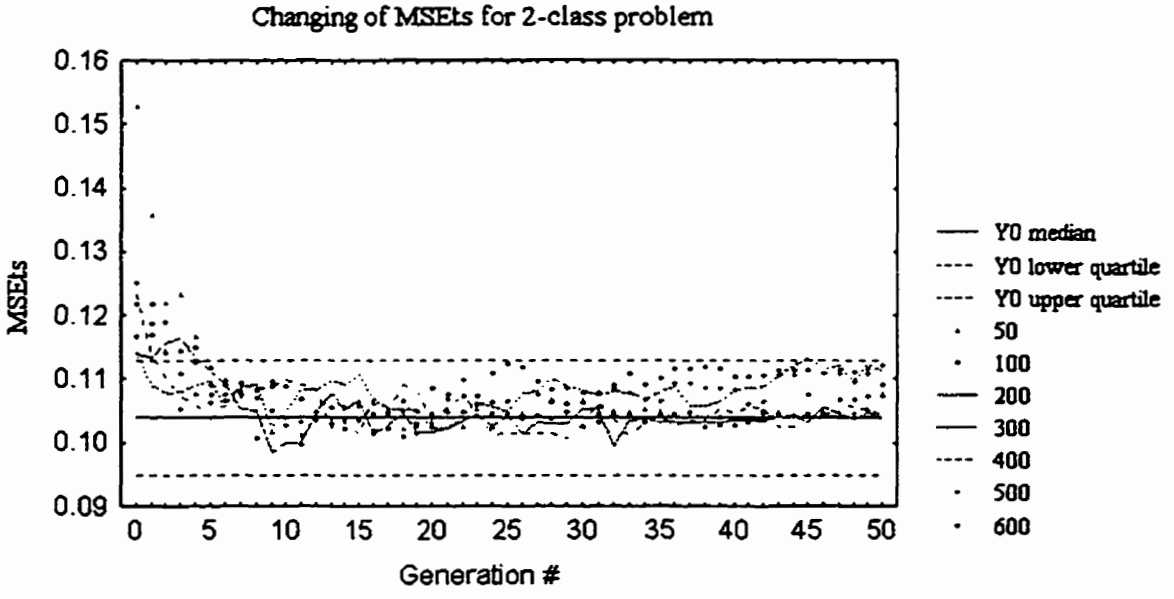


Fig. 3.4

To select the optimal value for n we have plotted not only the median of MSE_{ts}, but also the quartiles (Fig. 3.5). The distinct rise of the median MSE_{ts} between the 9th and 32nd generations is caused by random factors, since its lower quartile, aside from some fluctuations, is close to the lower quartile of x_0. Only after the 32nd generation is there a robust trend of increasing. Thus any value between 9 and 32 is acceptable for n. We have chosen n=20, in the middle.

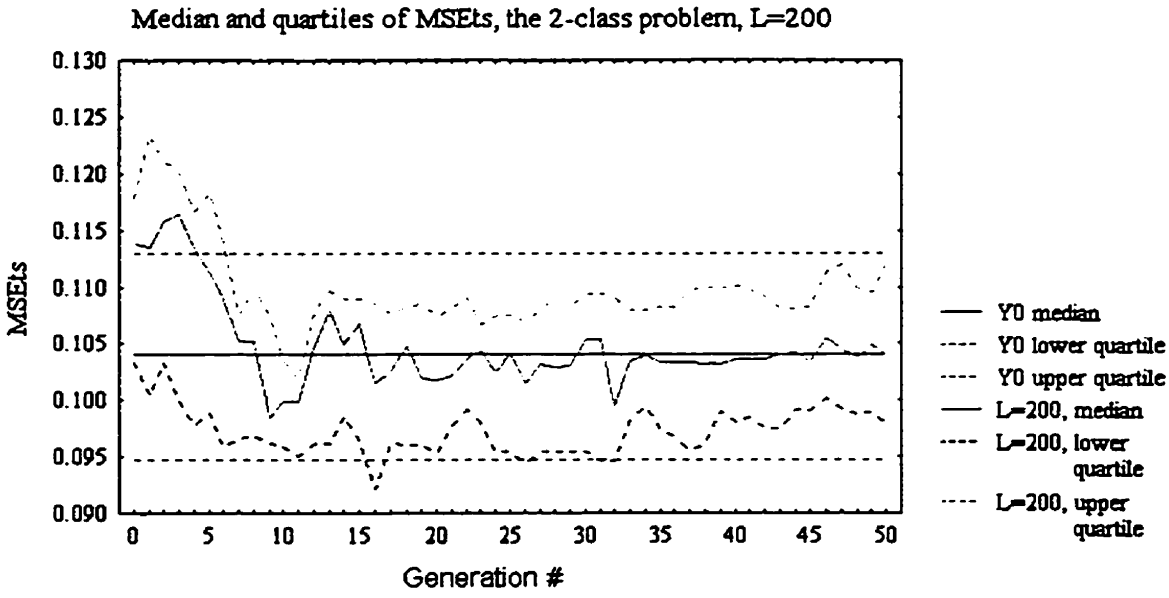


Fig. 3.5

To optimize the values of p_m and p_c, we have chosen 5 values for p_m -{0.0001, 0.0003, 0.001, 0.003, 0.01} and 5 values for p_c - {0.60, 0.63, 0.66, 0.69, 0.72}. A 2-dimensional grid was created with these values and GA-ORS was run 10 times for each node of the grid.

To choose the best combinations of probabilities, the quality control methods of Statistica were used. Thus we have switched to mean values instead of medians. There are two justification for this choice. First, the problem resembles a typical quality control problem. We have 250 observations overall, for a given combination of probabilities, we must decide, based on ten samples, if there is a significant difference between the distribution of these ten samples and the rest. Second, all but a few samples are distributed normally (see Fig. 3.6). We realize that ANOVA is more appropriate tool for the problem, but the *de facto* implementation of the quality control charts demonstrated that the artificial data are almost indifferent to the change in the values of the probabilities.

The upper and lower control limits for the X-BAR chart were chosen at 2 sigmas. Results are given in the Fig. 3.7. It can be easily seen that there are three combinations of probabilities with MSE_{ts} beyond the control limits. For two of them, (0.63,0.01) and (0.66,0.01), MSE_{ts} is higher than the upper control limit which means that these combinations are particularly bad. The combination (0.60,0.003) gave MSE_{ts} lower than the lower control limit (LCL), two others ((0.66,0.003) and (0.69,0.0003)) produced values which are very close to the LCL.

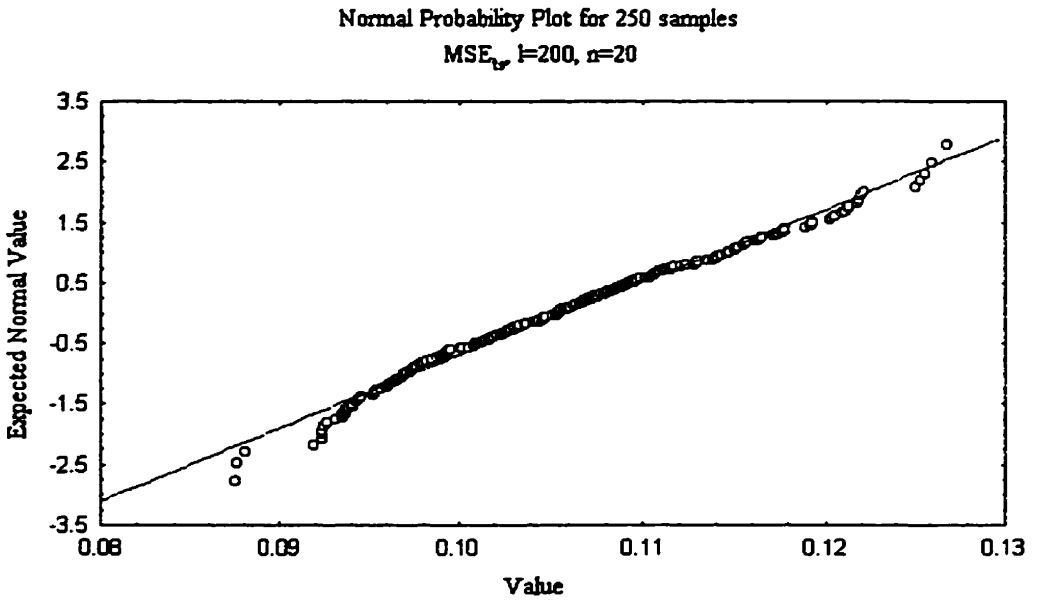

Normal Probability Plot for 250 samples
MSE$_{t,r}$, l=200, n=20
Expected Normal Value
3.5
2.5
1.5
0.5
-0.5
-1.5
-2.5
-3.5
0.08
0.09
0.10
0.11
0.12
0.13
Value

Fig. 3.6

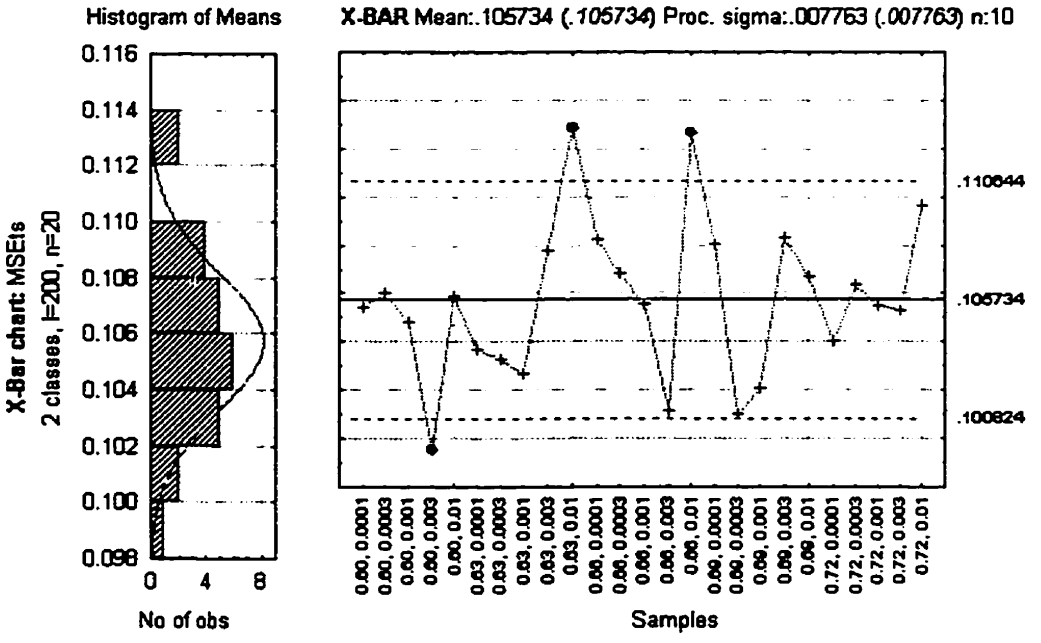

Histogram of Means
X-BAR Mean:.105734 (.105734) Proc. sigma:.007763 (.007763) n:10
X-Bar chart: MSEIs
2 classes, l=200, n=20
0.116
0.114
0.112
0.110
0.108
0.106
0.104
0.102
0.100
0.098
0
4
8
No of obs
.110644
.105734
.100824
0.60, 0.0001
0.60, 0.0003
0.60, 0.001
0.60, 0.003
0.60, 0.01
0.63, 0.0001
0.63, 0.0003
0.63, 0.001
0.63, 0.003
0.63, 0.01
0.66, 0.0001
0.66, 0.0003
0.66, 0.001
0.66, 0.003
0.66, 0.01
0.69, 0.0001
0.69, 0.0003
0.69, 0.001
0.69, 0.003
0.69, 0.01
0.72, 0.0001
0.72, 0.0003
0.72, 0.001
0.72, 0.003
0.72, 0.01
Samples

Fig 3.7

3-class problem

Like for the 2-class problem, after a few generations MSE_{tr} is below the lower quartile of x_0, thus indicating overfitting. Therefore we switched our attention to the MSE_{ts}'s behavior. Again, out of the 7 possible l values, only 3 are worth analysing in detail (see Fig.3.8), namely l=200, l=300, or l=400. The erratic behavior of MSE_{ts} for l=300 allows us to choose from the two remaining values.

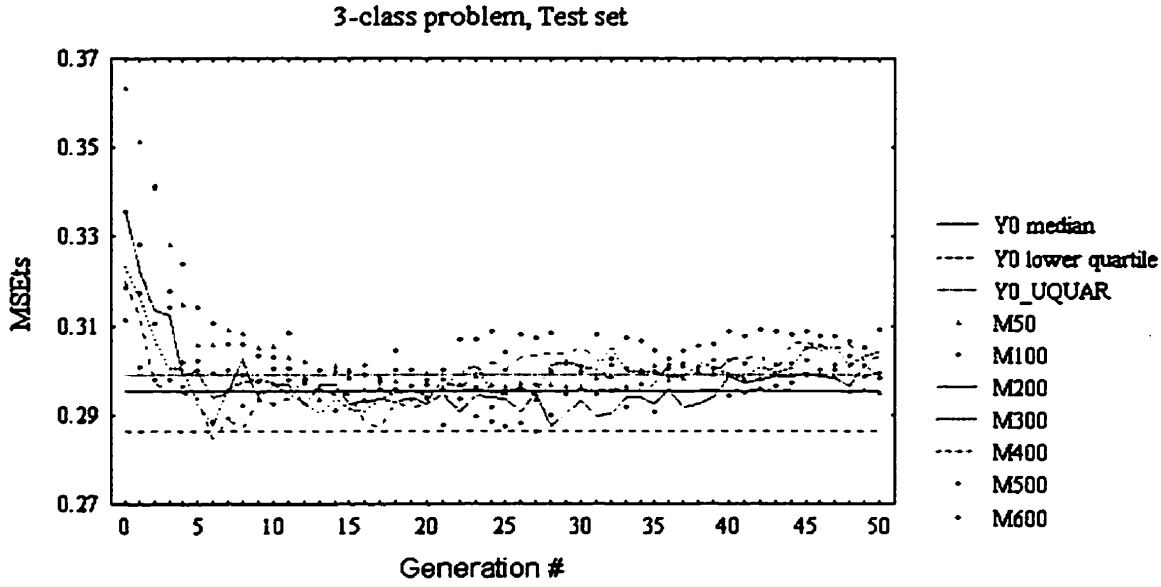


Fig 3.8

For l=200, the minimum of MSE_{ts} is achieved at n=28. For l=400 there are two minima, at n=6 and at n=17. Again, we pick a value in the middle (n=11) and we see that the latter case requires 400*(11+1)=4800 runs of LDA, while the former requires 200*(28+1)=5800 runs, thus leaving us with l=400 and n=11.

Repeating the analysis done for the 2-class problem, we discover (Fig. 3.9) that all but two combinations produced MSE_{ts} values within the control limits. The two

exceptions are (0.60,0.0001) and (0.72,0.01), the two extremes. We must avoid either of these.

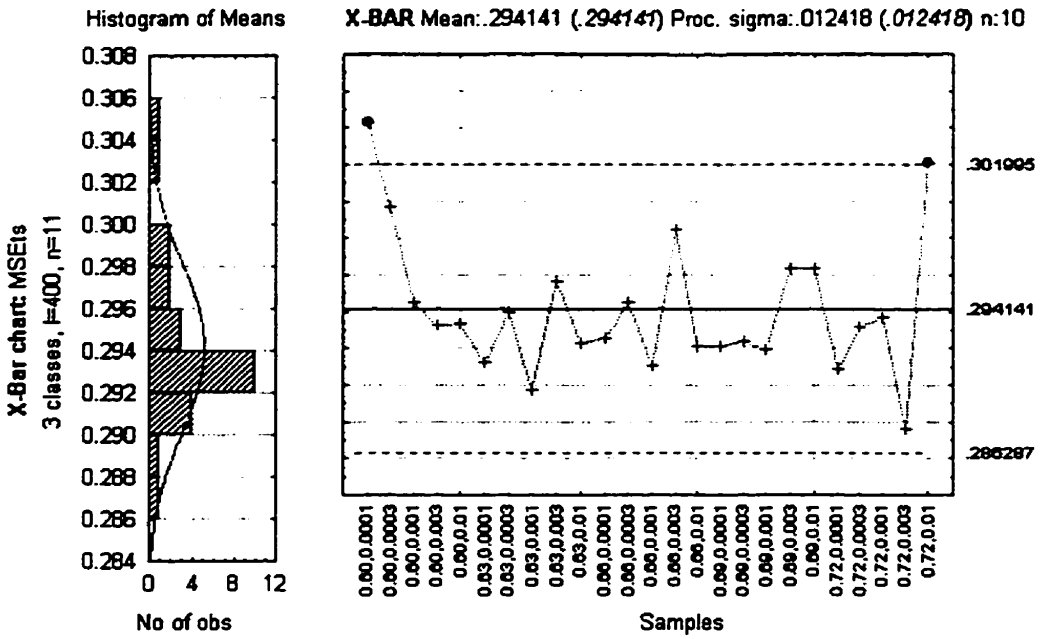


Fig 3.9

Although we have found five values beyond the control limits (2-class problem and 3-class problem combined), to do so we had to set the control limits to 2 sigma rather than the more common 3 sigma. Therefore we can give only cautious recommendations regarding the choice of probabilities. The only definite recommendation is to avoid too high or too low probabilities of mutation. The range [0.0003,0.003] will probably work well. All the analyzed values of p_{cross} are acceptable. The difference between the best combinations and the others is marginal.

Thus, recommended values are: the population size – 200-400, the number of generations – below 30, the probability of crossover – 0.60-0.72, and the probability of mutation – 0.0003-0.003. We have increased slightly the recommended number of

generations, compared to the optimal values we had chosen, since those number were obtained for two very specific artificially generated data sets. The real life data might need more generations to converge and might do so without overfitting.

3.6. Number of selected regions.

Although the number of selected regions is one of parameters, we decided to analyze its influence in a separate chapter, because of the special role it plays in the algorithm.

It is obvious that the more regions are selected, the greater is the danger of overfitting. Less obvious is the algorithm's behavior in the case when number of the regions is equal to or less than the number of features in the data. In our case, only 6 peak heights out of 10 were discriminative, while the other 4 were generated identically distributed for all classes (see Table 3.1). Therefore we have 6 discriminatory peaks and we need at least 6 regions to use all the information contained in the spectra. With greater number of regions (L) some of them can be chosen in the nondiscriminatory areas; this also can happen with a smaller number.

To test the method we have run the program 10 times for each value of L and followed the parameters of the bilinear model and the residual sum of squares. One would expect that with increased overfitting the results would be better described by the bilinear model, whereas without of overfitting the bilinear model would not decrease the residual sum of squares and the results for the training and test sets would be highly correlated (a~1). The last condition is very important, because we saw that the bilinear model can work as well as the linear when all the points on the MSE_{tr}-MSE_{ts} plane lie in the overfit area.

The mean values of the parameters of the bilinear model for the 2-class problem are plotted in Fig. 3.10. As we can see, x_0 and y_0 are very stable and practically identical for all L values. The a parameter decreases slightly with increasing L.

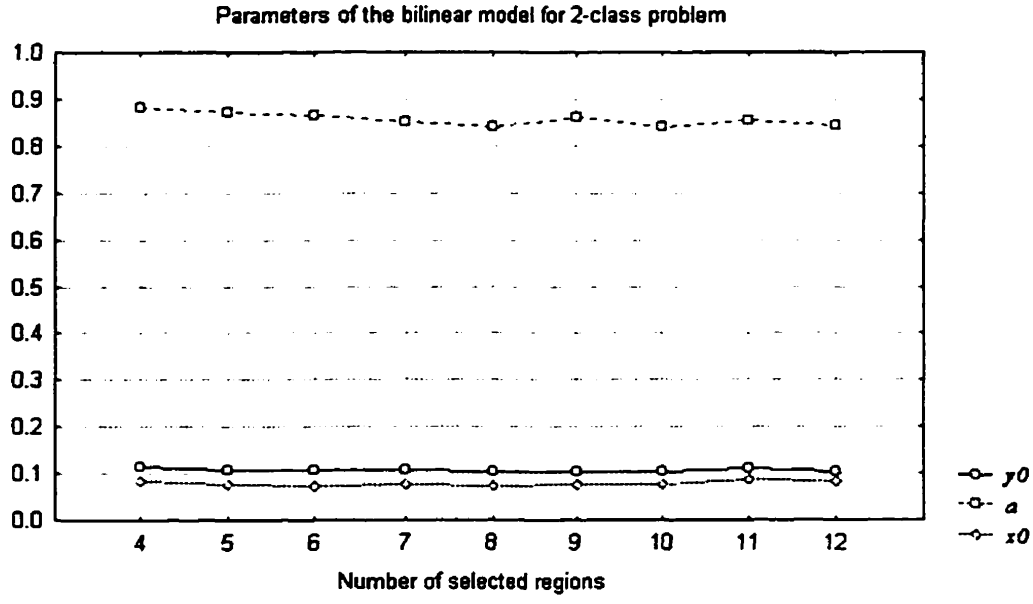


Fig. 3.10

Dependence of the residual sum of squares on the number of regions.
(2-class problem)

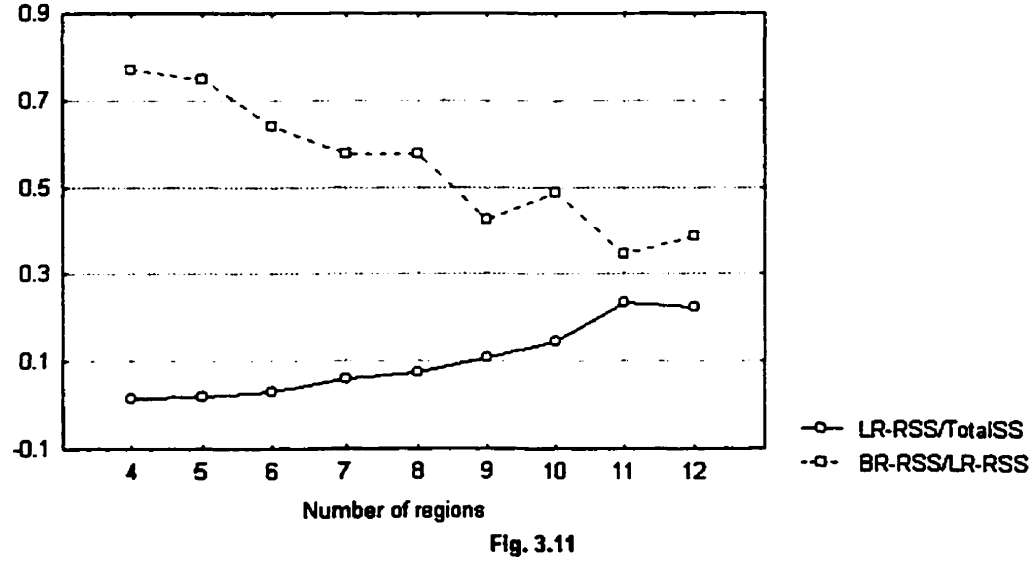


Fig. 3.11

An analysis of the residual errors reveals the true picture regarding the overfitting (Fig. 3.11). The ratio of the residual sum of squares (RSS) in the linear model and the total sum of squares (LR-RSS/TotalSS) increases dramatically with L, implying that the quality of the linear model decreases. In contrast, the ratio of the RSS in the bilinear and linear models decreases significantly, therefore, the bilinear model describes the data better for greater values of L. Thus, while the overfitting influences the results for small L, the severity of the problem increases with L.

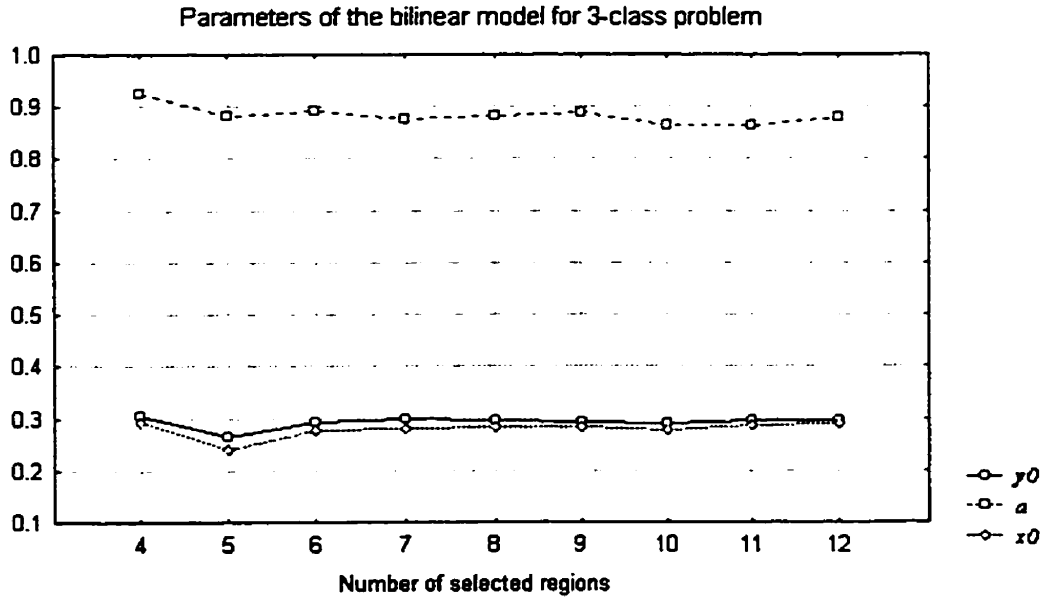


Fig. 3.12

The parameters of the bilinear model for the 3-class problem behave similarly to those for the 2-class problem (Fig. 3.12) with one notable exception. Both x_0 and y_0 have well-defined minima at L=5. Detailed analysis of the minima revealed that one of the 10 runs with L=5 gave no overfitting at all. As a result, x_0 and y_0 could be chosen arbitrarily

(see Appendix B.). In this case x_0 was very close to 0 and y_0 was negative. If we correct this artifact, the conclusion will be the same as for the 2-class problem: x_0 and y_0 are intrinsic characteristics of the data and do not depend on the parameters.

The a parameter for the 3-class problem is slightly higher, suggesting better correlation between results for the training and test sets, and less overfitting. Thus the 2-class problem seems more susceptible to overfitting, supporting our hypothesis stated earlier.

Analysis of the residual sums of squares for the 3-class problem (Fig. 3.13) also shows that fewer regions reduce the overfitting. Moreover, the BR-RSS/LR-RSS has a maximum at L=5, i.e., at one region less than the true number of features in the spectra.

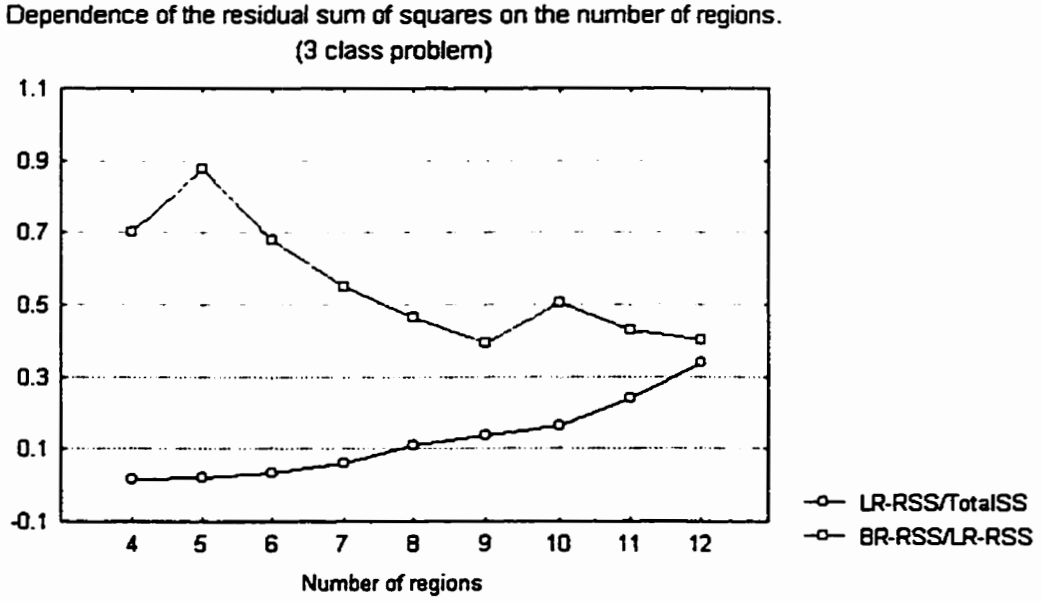


Fig. 3.13

Five out of nine values on the a curve are higher for the 3-class problem, the rest are higher for the 2-class problem, therefore this parameter cannot signal the tendency for overfitting.

We have demonstrated an increasing overfitting with increasing number of features for both datasets. To recommend an optimal value of L we need an additional criterion. Despite low variability, y_0 can fulfill this role. Indeed, y_0 is in essence a lower bound for the expectation of the MSE_{ts}. The lower this bound the better the results on a test set. For the 2-class problem the minimum was achieved at L=9, for the 3-class problem, ignoring the artifact at L=5, the true minimum was achieved at L=10. Therefore, although the overfitting increased when the number of sought features exceeded the actual number of features present, we may expect a slight improvement with a few extra features. This was probably achieved at the expense of increased variance in y_0, but we did not obtain enough data to support this statement.

Conclusions and future development

An evolutionary process of development of the methods of MR spectral analysis brought us from handpicked peaks, through PCA and/or DP-based methods of feature selection to a GA-based feature extraction method. Our contribution varies from method to method. In the case of the PCA, we simply coded the known algorithm. In the case of the Dynamic Programming (DP)-based feature selection, the published algorithm was modified and improved. The GA implementation is original for feature selection, as applied to the analysis of spectra. An original encoding method implemented in our algorithm allows combining of the power and flexibility of the GA while retaining the spectral identity of the new features, thus facilitating the interpretation of the results.

As we showed, the powerful methods of feature extraction/selection are susceptible to overfitting. This means that the results for the test set (MSE_{ts}) are weakly correlated with the results for the training test (MSE_{tr}). To demonstrate this we developed a simplified bilinear model for the dependence of MSE_{ts} on MSE_{tr}, for which we assumed that beyond a critical value MSE_{ts} is a constant (steady-state).

To test both the method and the bilinear model, artificial data sets were created to simulate typical problems of MR spectroscopy. The artificial data were used to demonstrate the superiority of the bilinear model over the linear. It was also shown that the x_0 and y_0 parameters of the bilinear model are intrinsic properties of the data and vary slightly, whereas the a value is very sensitive to the method's parameters. The bilinear model gave us reasonable means of selecting parameters for the feature extraction

methods. It would have been extremely dangerous to base our conclusions just on MSE_{tr}, since it loses its predictive power due to overfitting. In contrast, MSE_{ts} allowed us to choose the parameters that would produce a classifier delivering its promises.

The results also confirmed the superiority of the GA-based feature extraction algorithm over the DP. We have shown that the probability of error for noise-free data is close to the designed probabilities of error. As expected, noise deteriorates the results, but we don't have benchmarks to fully assess this case. A very peculiar result was produced for the 2-class problem when only white noise was present. For some runs of the methods the a parameter of the model became negative. This means that the assumed linear phase in our model is not present, but the steady-state phase surprisingly degenerated into a linear one with a negative slope. The problem was less severe for the 3-class data, and was negligible in the presence of complex noise.

The method gave good results for real-life IR data. Due to availability of many spectra, we have obtained a linear dependence of the MSE_{ts} on MSE_{tr} without the constant stage. However, reducing the size of the data set created the same overfitting problem as for artificial data. Thus we illustrated once again the importance of the size of the dataset.

We paid a lot of attention to the overfitting problem; unfortunately, it is not clear yet how to prevent it without dramatically decreasing the power of the method. The most obvious way is to increase the number of samples in both training and test sets. This will never fail, but may be unrealistic due to the high cost of the data collection.

If we have enough data, we can use the approach that is often used in PLS regression /31/. The data set is divided into three groups instead of two. The groups are

the training set, the test set, and the validation set. The terminology in the literature is not consistent; we'll use the above definition in the following.

First, the training set is used to guide the GA_ORS. Then we'll use the test set to choose the best solution out of the elite group. The validation set will help us to estimate the overall performance of such a two-stage process. The hope is that the small size of the elite group would not leave enough room for overfitting.

Another possible method development is a modification of the bilinear model, allowing an arbitrary (maybe only negative) slope instead of the constant stage. As we have seen, this modification has some merits. If this model had described the MSE_{ts}-MSE_{tr} dependence better, this would mean that overfitting could be costly, not just indifferent.

The algorithm itself can be improved in many ways. One possible development is a modification of the objective function. As we have seen in the literature review, what objective function is used is important for RDA. Since RDA requires a two-parameter optimization, it is also susceptible to overfitting, and we can expect that the choice of the objective function can also be important in our case.

One of the recent developments in GA theory, sharing functions, was not yet implemented in our program. The sharing function can prevent premature collapse of the population into a smaller set of chromosomes, and as a result, allows us to get better results, or to reach the same results faster. An obvious drawback of this approach is that one additional parameter has to be optimized.

Both GA- and DP- regions selections methods are implemented in software. The program has been used successfully for the analysis of both MR and IR data. The results were presented at several annual meetings of the Society of Magnetic Resonance as refereed contributions /24,25/. A paper, describing the methods is in print /49/.

Appendix A. Update of the statistics for the weighted LDA with leave-one-out cross validation

Let us consider the problem of a weighted discriminant analysis. This means that for each of the independent and identically distributed p-dimensional observations $\mathbf{x}_i, i = 1,...,n$, we are given weights w_i. Then the estimates of the mean vector $\overline{\mathbf{x}}$ and the covariance matrix S are calculated according to the following formulae:

$$W = \sum_{i=1}^{n} w_i \qquad \text{A.1}$$

$$\overline{\mathbf{x}} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \cdot w_i}{W} \qquad \text{A.2}$$

$$\overline{w} = \frac{\sum_{i=1}^{n} w_i \cdot w_i}{W} \qquad \text{A.3}$$

$$\mathbf{S} = \frac{1}{W - \overline{w}} \sum_{i=1}^{n} w_i \cdot (\mathbf{x}_i - \overline{\mathbf{x}}) \cdot (\mathbf{x}_i - \overline{\mathbf{x}})^T \qquad \text{A.4}$$

It will be convenient to introduce another matrix:

$$\mathbf{C} = (W - \overline{w}) \cdot \mathbf{S} \qquad \text{A.5}$$

These estimates are unbiased. This is obvious for the mean vector:

$$E(\overline{\mathbf{x}}) = \frac{\sum_{i=1}^{n} E(\mathbf{x}_i) \cdot w_i}{W} = \frac{E(\xi) \cdot \sum_{i=1}^{n} w_i \sum_{i=1}^{n} w_i}{W} = E(\xi)$$

where ξ is a random vector we have sampled. The proof is slightly longer for the estimate of the covariance matrix:

$$E(S)=\frac{1}{W-\overline{w}}\sum_{i=1}^{n}w_i E\left[(x_i-\overline{x})\cdot(x_i-\overline{x})^T\right]=$$

$$\frac{1}{W-\overline{w}}\sum_{i=1}^{n}w_i E\left[(x_i-E(\xi)+E(\xi)-\overline{x})\cdot(x_i-E(\xi)+E(\xi)-\overline{x})^T\right]=$$

$$\frac{1}{W-\overline{w}}\sum_{i=1}^{n}w_i\left[Cov(\xi)-2\cdot E\left((x_i-E(\xi))\cdot(\overline{x}-E(\xi))^T\right)+E\left((\overline{x}-E(\xi))\cdot(\overline{x}-E(\xi))^T\right)\right]$$

Since x_i are identically independently distributed,

$$E\left((x_i-E(\xi))\cdot(x_j-E(\xi))^T\right)=\begin{cases}0, \text{if } i\neq j\\ Cov(\xi), \text{if } i=j\end{cases}, \text{ therefore}$$

$$E\left((x_j-E(\xi))\cdot(\overline{x}-E(\xi))^T\right)=E\left((x_j-E(\xi))\cdot\left(\frac{\sum_{i=1}^{n}x_i\cdot w_i}{W}-E(\xi)\right)^T\right)$$

$$=\frac{\sum_{i=1}^{n}w_i\cdot E\left((x_j-E(\xi))\cdot(x_i-E(\xi))^T\right)}{W}=\frac{w_j\cdot Cov(\xi)}{W}$$

and

$$E\left((\overline{x}-E(\xi))\cdot(\overline{x}-E(\xi))^T\right)=E\left(\left(\frac{\sum_{j=1}^{n}x_j\cdot w_j}{W}-E(\xi)\right)\cdot\left(\frac{\sum_{i=1}^{n}x_i\cdot w_i}{W}-E(\xi)\right)^T\right)$$

$$=\frac{\sum_{i=1}^{n}w_i^2\cdot E\left((x_i-E(\xi))\cdot(x_i-E(\xi))^T\right)}{W^2}=\frac{\sum_{i=1}^{n}w_i^2\cdot Cov(\xi)}{W^2}$$

Finally,

$$E(S) = \frac{1}{W - \overline{w}}$$

$$\bullet \sum_{i=1}^{n} w_i \left[Cov(\xi) - 2 \cdot E\left((x_i - E(\xi)) \cdot \frac{w_i}{W} \cdot (x_i - E(\xi))^T \right) + E\left((\overline{x} - E(\xi)) \cdot (\overline{x} - E(\xi))^T \right) \right]$$

$$= \frac{W \cdot Cov(\xi) - 2 \cdot \sum_{i=1}^{n} \frac{w_i^2 \cdot Cov(\xi)}{W} + \sum_{i=1}^{n} w_i \left(\sum_{j=1}^{n} \frac{w_j^2 \cdot Cov(\xi)}{W^2} \right)}{W - \overline{w}}$$

$$= Cov(\xi) \cdot \left(\frac{W - 2 \cdot \sum_{i=1}^{n} \frac{w_i^2}{W} + W \left(\sum_{j=1}^{n} \frac{w_j^2}{W^2} \right)}{W - \overline{w}} \right) = Cov(\xi) \cdot \left(\frac{W - \sum_{i=1}^{n} \frac{w_i^2}{W}}{W - \overline{w}} \right)$$

$$= Cov(\xi) \cdot \left(\frac{W - \overline{w}}{W - \overline{w}} \right) = Cov(\xi)$$

Thus, we have shown that the above estimates are indeed unbiased.

To update the covariance matrix for the leave-one-out method, we will follow Lachenbruch /7/.

Suppose we have deleted the j-th observation $\mathbf{x}_j$; then we can calculate the difference between that observation and the mean vector:

$$\overline{\mathbf{u}} = \mathbf{x}_j - \overline{\mathbf{x}} \qquad \text{A.6}$$

Now we can update our estimates:

$$\tilde{W} = W - w_j \qquad \text{A.7}$$

$$\tilde{\overline{w}} = \frac{W \cdot \overline{w} - (w_j)^2}{\tilde{W}} \qquad \text{A.8}$$

$$\tilde{\overline{\mathbf{x}}} = \overline{\mathbf{x}} - \frac{w_j \cdot \mathbf{u}}{\tilde{W}} \qquad \text{A.9}$$

$$\tilde{\mathbf{C}} = \mathbf{C} - \frac{w_j \cdot W}{\tilde{W}} \cdot \mathbf{u} \cdot \mathbf{u}^T \qquad \text{A.10}$$

$$\tilde{\mathbf{S}} = \frac{1}{\tilde{W} - \tilde{\overline{w}}} \cdot \tilde{\mathbf{C}} = \frac{1}{\tilde{W} - \tilde{\overline{w}}} \cdot \left((W - \overline{w}) \cdot \mathbf{S} - \frac{w_j \cdot W}{\tilde{W}} \cdot \mathbf{u} \cdot \mathbf{u}^T \right)$$

$$= \frac{W - \overline{w}}{\tilde{W} - \tilde{\overline{w}}} \cdot \left(\mathbf{S} - \frac{w_j \cdot W}{(W - \overline{w})\tilde{W}} \cdot \mathbf{u} \cdot \mathbf{u}^T \right) \qquad \text{A.11}$$

To show this, one must simply substitute the new variables into the definitions of our estimates.

To update the probabilities in the leave-one-out method we also need an updated determinant of the covariance matrix. As we have seen, the updated covariance matrix differs from the original by a rank 1 matrix and a scaling factor. Let $A = \alpha \cdot \left(B + u \cdot v^T\right)$ and $\mathbf{B} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^T$ is the eigenvalue decomposition of $\mathbf{B}$ (i.e., $\mathbf{V} \cdot \mathbf{V}^{\mathbf{T}} = \mathbf{I}$ and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \cdots, \lambda_p)$), then

$$\det \mathbf{A} = \alpha^p \cdot \det \mathbf{B} \cdot \left(1 + \sum_{i=1}^{p} \frac{\tilde{u}_i \cdot \tilde{v}_i}{\lambda_i}\right), \text{ where } \tilde{\mathbf{u}} = \mathbf{V}^{\mathbf{T}} \cdot \mathbf{u} \text{ and } \tilde{\mathbf{v}} = \mathbf{V}^{\mathbf{T}} \cdot \mathbf{v}.$$

Indeed,

$$\det \mathbf{A} = \alpha^p \cdot \det\left(\mathbf{B} + \mathbf{u} \cdot \mathbf{v}^T\right) = \alpha^p \cdot \det\left(\mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^T + \mathbf{u} \cdot \mathbf{v}^T\right) =$$
$$= \alpha^p \cdot \det \mathbf{V} \cdot \det \mathbf{V}^T \cdot \det\left(\mathbf{\Lambda} + \tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}^T\right) = \alpha^p \cdot \det\left(\mathbf{\Lambda} + \tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}}^T\right)$$

but

$$\det\left(\Lambda+\tilde{\mathbf{u}}\cdot\tilde{\mathbf{v}}^{T}\right)=\det\begin{vmatrix}\lambda_1+\tilde{u}_1\cdot\tilde{v}_1 & \tilde{u}_1\cdot\tilde{v}_2 & \cdots & \cdots & \tilde{u}_1\cdot\tilde{v}_p\\ \tilde{u}_2\cdot\tilde{v}_1 & \lambda_2+\tilde{u}_2\cdot\tilde{v}_2 & & & \tilde{u}_2\cdot\tilde{v}_p\\ \vdots & & \ddots & & \vdots\\ \tilde{u}_{p-1}\cdot\tilde{v}_1 & & & \ddots & \vdots\\ \tilde{u}_p\cdot\tilde{v}_1 & \cdots & \cdots & \cdots & \lambda_p+\tilde{u}_p\cdot\tilde{v}_p\end{vmatrix}=$$

$$\det\begin{vmatrix}\lambda_1 & \tilde{u}_1\cdot\tilde{v}_2 & \cdots & \cdots & \tilde{u}_1\cdot\tilde{v}_p\\ 0 & \lambda_2+\tilde{u}_2\cdot\tilde{v}_2 & & & \tilde{u}_2\cdot\tilde{v}_p\\ \vdots & & \ddots & & \vdots\\ 0 & & & \ddots & \vdots\\ 0 & \cdots & \cdots & \cdots & \lambda_p+\tilde{u}_p\cdot\tilde{v}_p\end{vmatrix}+\det\begin{vmatrix}\tilde{u}_1\cdot\tilde{v}_1 & \tilde{u}_1\cdot\tilde{v}_2 & \cdots & \cdots & \tilde{u}_1\cdot\tilde{v}_p\\ \tilde{u}_2\cdot\tilde{v}_1 & \lambda_2+\tilde{u}_2\cdot\tilde{v}_2 & & & \tilde{u}_2\cdot\tilde{v}_p\\ \vdots & & \ddots & & \vdots\\ \tilde{u}_{p-1}\cdot\tilde{v}_1 & & & \ddots & \vdots\\ \tilde{u}_p\cdot\tilde{v}_1 & \cdots & \cdots & \cdots & \lambda_p+\tilde{u}_p\cdot\tilde{v}_p\end{vmatrix}=$$

$$\lambda_1\cdot\det\begin{vmatrix}\lambda_2+\tilde{u}_2\cdot\tilde{v}_2 & \cdots & \cdots & \tilde{u}_2\cdot\tilde{v}_p\\ \vdots & \ddots & & \vdots\\ \vdots & & \ddots & \vdots\\ \tilde{u}_p\cdot\tilde{v}_2 & \cdots & \cdots & \lambda_p+\tilde{u}_p\cdot\tilde{v}_p\end{vmatrix}+\tilde{v}_1\cdot\det\begin{vmatrix}\tilde{u}_1 & 0 & \cdots & \cdots & 0\\ \tilde{u}_2 & \lambda_2 & & & 0\\ \vdots & & \ddots & & \vdots\\ \tilde{u}_{p-1} & & & \ddots & \vdots\\ \tilde{u}_p & 0 & \cdots & \cdots & \lambda_p\end{vmatrix}=$$

$$\lambda_1\cdot\det\begin{vmatrix}\lambda_2+\tilde{u}_2\cdot\tilde{v}_2 & \cdots & \cdots & \tilde{u}_2\cdot\tilde{v}_p\\ \vdots & \ddots & & \vdots\\ \vdots & & \ddots & \vdots\\ \tilde{u}_p\cdot\tilde{v}_2 & \cdots & \cdots & \lambda_p+\tilde{u}_p\cdot\tilde{v}_p\end{vmatrix}+\frac{\tilde{u}_1\cdot\tilde{v}_1}{\lambda_1}\cdot\prod_{i=1}^{p}\lambda_i=$$

$$\cdots=\prod_{i=1}^{p}\lambda_i\cdot\left(1+\sum_{i=1}^{p}\frac{\tilde{u}_i\cdot\tilde{v}_i}{\lambda_i}\right)=\det\mathbf{B}\cdot\left(1+\sum_{i=1}^{p}\frac{\tilde{u}_i\cdot\tilde{v}_i}{\lambda_i}\right)$$

Thus, we know how to update the covariance matrix and its determinant. Now we can update the inverse of the covariance matrix. Following Lachenbruch, we'll use Bartlett's identity: if $\mathbf{B}=\mathbf{A}+\mathbf{u}\cdot\mathbf{v}^{\mathrm{T}}$, then $\mathbf{B}^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\cdot\mathbf{u}\cdot\mathbf{v}^{\mathrm{T}}\cdot\mathbf{A}^{-1}/\left(1+\mathbf{v}^{\mathrm{T}}\cdot\mathbf{A}^{-1}\cdot\mathbf{u}\right)$

In our case this means:

$$\tilde{\mathbf{S}}^{-1}=\frac{\tilde{\mathbf{W}}-\tilde{\overline{\mathbf{w}}}}{\mathbf{W}-\overline{\mathbf{w}}}\cdot\left(\mathbf{S}-\frac{\mathbf{w}_j\cdot\mathbf{W}}{(\mathbf{W}-\overline{\mathbf{w}})\tilde{\mathbf{W}}}\cdot\mathbf{u}\cdot\mathbf{u}^{T}\right)^{-1}$$

$$=\frac{\tilde{\mathbf{W}}-\tilde{\overline{\mathbf{w}}}}{\mathbf{W}-\overline{\mathbf{w}}}\cdot\left(\mathbf{S}^{-1}+\mathrm{c}\cdot\mathbf{S}^{-1}\cdot\mathbf{u}\cdot\mathbf{u}^{T}\cdot\mathbf{S}^{-1}/\left(1-\mathrm{c}\cdot\mathbf{u}^{T}\cdot\mathbf{S}^{-1}\cdot\mathbf{u}\right)\right)$$

A.12

where $\mathrm{c}=\dfrac{\mathbf{w}_j\cdot\mathbf{W}}{(\mathbf{W}-\overline{\mathbf{w}})\tilde{\mathbf{W}}}$.

This formula can be applied both to linear and quadratic discriminant analyses. The only difference consists in the calculation of $W - \overline{w}$ and its update. In the case of LDA we have to replace them by the sums of all individual terms for each class ($\sum_{i=1}^{p}(W_i - \overline{w}_i)$ and the corresponding update)

Finally :

$$\det \widetilde{S} = \left(\frac{W - \overline{w}}{\widetilde{W} - \widetilde{\overline{w}}}\right)^{p} \cdot \det S \cdot \left(1 - \frac{w_j \cdot W}{(W - \overline{w}) \cdot \widetilde{W}} \cdot \sum_{i=1}^{p} \frac{\widetilde{u}_i^2}{\lambda_i}\right) \qquad \text{A.13}$$

Appendix B. The solution for a nonlinear regression model

y(x)=y0+a•(x-x0)•u(x-x0)

Let us consider a nonlinear model $y(x) = y_0 + a \cdot (x - x_0) \cdot u(x - x_0)$, where $u(x)$ is the unit step function. This means that

$$y(x) = \begin{cases} y_0, & if\ x \le x_0 \\ y_0 + a \cdot (x - x_0), & if\ x > x_0 \end{cases} \qquad \text{B.1}$$

Let us also have a set of experimental points $(X_i, Y_i), i = 1,\ldots,N$ with the X values sorted in ascending order. The problem is to choose a, y_0, x_0 such that the mean square error (MSE) between experimental and theoretical data is minimal. If x_0 is given, we can partition all the experimental points into two groups, those which are less and those which are greater than x_0. Therefore we'll have an analytical expression for the MSE, which can be minimized provided that the point partition is unchanged. Thus our problem looks like a linear programming problem. We have a set of experimental X values which divides the set of real numbers into intervals. For each interval the MSE can be calculated analytically and minimized. The minimum can be achieved either inside or on the boundary of the interval.

After we have described the main idea of the algorithm, we'll solve the problem for each interval separately, starting from two half-open infinite intervals.

Before doing the actual minimization we'll introduce some useful definitions:

$$E(Z) = \frac{1}{N} \cdot \sum_{i=1}^{N} Z_i$$

$$E_k(Z) = \frac{1}{k} \cdot \sum_{i=1}^{k} Z_i$$

$$E^{N-k}(Z) = \frac{1}{N-k} \cdot \sum_{i=k+1}^{N} Z_i$$

where $Z = f(X,Y)$ is any function of two variables and $Z_i = f(X_i, Y_i)$.

The statistics are the standard sample estimate of the average and its truncated modifications. We'll need the following equations, which easily follow from the definitions:

$$E_N(Z) = E^N(Z) = E(Z) \qquad \text{B.2}$$

$$E_1(Z) = Z_1 \qquad \text{B.3}$$

$$E^1(Z) = Z_N \qquad \text{B.4}$$

$$E_{k+1} = \frac{k \cdot E_k(Z) + Z_{k+1}}{k+1} \qquad \text{B.5}$$

$$E^{N-k+1} = \frac{(N-k) \cdot E^{N-k}(Z) + Z_k}{N-k+1} \qquad \text{B.6}$$

$$E(Z) = \frac{k \cdot E_k(Z) + (N-k) \cdot E^{N-k}(Z)}{N} \qquad \text{B.7}$$

B.1 Two extreme cases of the problem.

These two cases are either $x_0 \in (-\infty, X_0]$ or $x_0 \in [X_N, \infty)$. We'll start with the latter.

In this case $y(x) = y_0$ and the solution is obvious:

$$\begin{aligned} & x_0 \in [X_N, \infty) \\ & a = 0 \\ & y_0 = E(Y) \\ & MSE = E(Y^2) - E(Y)^2 \end{aligned} \qquad \text{B.8}$$

When $x_0 \in (-\infty, X_0]$ we have got a standard linear regression model with a known solution again:

$$\begin{aligned} & x_0 \in (-\infty, X_0] \\ & a = \frac{E(XY) - E(X) \cdot E(Y)}{E(X^2) - E(X)^2} \\ & y_0 = E(Y) - a \cdot [E(X) - x_0] \\ & MSE = E(Y^2) - E(Y)^2 - \frac{[E(XY) - E(X) \cdot E(Y)]^2}{E(X^2) - E(X)^2} \end{aligned} \qquad \text{B.9}$$

As we can easily see, if there is correlation between the X and Y values ($E(XY) - E(X) \cdot E(Y) \neq 0$), MSE is less for the linear regression, otherwise those two solutions will coincide.

B.2 The minimum is achieved inside

We assume that $X_k \le x_0 < X_{k+1}$, then

$$N \cdot MSE = \sum_{i=1}^{N} (Y_i - y(X_i))^2 = \sum_{i=1}^{k} (Y_i - y_0)^2 + \sum_{i=k+1}^{N} [Y_i - y_0 - a \cdot (X_i - x_0)]^2 =$$
$$\sum_{i=1}^{N} (Y_i - y_0)^2 + a^2 \cdot \sum_{i=k+1}^{N} (X_i - x_0)^2 - 2 \cdot a \cdot \sum_{i=k+1}^{N} (X_i - x_0) \cdot (Y_i - y_0) = \qquad \text{B.10}$$
$$N \cdot [E(Y) - 2 \cdot y_0 \cdot E(Y) + y_0^2] + a^2 \cdot (N-k) \cdot [E^{N-k}(X^2) - 2 \cdot x_0 \cdot E^{N-k}(X) + x_0^2]$$
$$-2 \cdot a \cdot (N-k) \cdot [E^{N-k}(XY) - x_0 \cdot E^{N-k}(Y) - y_0 \cdot E^{N-k}(X) + x_0 y_0]$$

Differentiating with respect to a, x_0, and y_0, we'll get:

$$\partial N \cdot MSE / \partial a = -2 \cdot (N-k) \cdot \{E^{N-k}(XY) - E^{N-k}(X) \cdot y_0 - x_0 \cdot E^{N-k}(Y) + x_0 \cdot y_0 -$$
$$a \cdot [E^{N-k}(X^2) - 2 \cdot x_0 \cdot E^{N-k}(X) + x_0^2]\}$$
$$\partial N \cdot MSE / \partial y_0 = -2 \cdot N \cdot (E(Y) - y_0) + 2 \cdot a \cdot (N-k) \cdot (E^{N-k}(X) - x_0)$$
$$\partial N \cdot MSE / \partial x_0 = 2 \cdot a \cdot (N-k) \cdot (E^{N-k}(Y) - y_0 - a \cdot E^{N-k}(X) + a \cdot x_0)$$

At the minimum the partial derivatives are equal to zero and the Hessian matrix is positive definite. Therefore to get the minimum we must solve the following system:

$$\begin{cases} E^{N-k}(XY) - E^{N-k}(X) \cdot y_0 - x_0 \cdot E^{N-k}(Y) + x_0 \cdot y_0 = \\ \quad a \cdot [E^{N-k}(X^2) - 2 \cdot x_0 \cdot E^{N-k}(X) + x_0^2] \\ E(Y) - y_0 = a \cdot \dfrac{N-k}{N} \cdot [E^{N-k}(X) - x_0)] \\ E^{N-k}(Y) - y_0 = a \cdot [E^{N-k}(X) - x_0] \end{cases} \qquad \text{B.11}$$

To get the last equation we have ignored the case a=0, since the solution for this is given by B.8.

Multiplying the third equation by $\frac{N-k}{N}$ and subtracting it from the second one, we get:

$$E(Y)-y_0=\frac{N-k}{N}\cdot[E^{N-k}(Y)-y_0] \quad \text{or}$$
$$k\cdot y_0=N\cdot E(Y)-(N-k)\cdot E^{N-k}(Y) \underset{B.7}{=} \qquad \text{B.12}$$
$$k\cdot E_k(Y)+(N-k))\cdot E^{N-k}(Y)-(N-k)\cdot E^{N-k}(Y)=k\cdot E_k(Y), \text{ therefore}$$
$$y_0=E_k(Y)$$

Multiplying the third equation of B.11 by $E^{N-k}(X)-x_0$ and subtracting it from the first one we get:

$$E^{N-k}(XY)-E^{N-k}(X)\cdot E^{N-k}(Y)=a\cdot[E^{N-k}(X^2)-E^{N-k}(X)^2], \quad \text{or}$$
$$a=\frac{E^{N-k}(XY)-E^{N-k}(X)\cdot E^{N-k}(Y)}{E^{N-k}(X^2)-E^{N-k}(X)^2} \qquad \text{B.13}$$

Based on B.11-B.13 we find that:

$$x_0=E^{N-k}(X)-\frac{E^{N-k}(Y)-y_0}{a}=$$
$$E^{N-k}(X)-\frac{[E^{N-k}(Y)-E_k(Y)]\cdot[E^{N-k}(X^2)-E^{N-k}(X)^2]}{E^{N-k}(XY)-E^{N-k}(X)\cdot E^{N-k}(Y)} \qquad \text{B.14}$$

Close examination of B.12 and B.13 reveals that the overall solution is a combination of independent solutions of the regression model $y(x)=y_0$ for the first k points and the linear regression model $y(x)=a\cdot x+(y_0-a\cdot x_0)$ for the rest of data. Therefore

$$MSE = \frac{1}{N}\{k\cdot[E_k(Y^2) - E_k(Y)^2] +$$

$$(N-k)\cdot[E^{N-k}(Y^2) - E^{N-k}(Y)^2 - \frac{[E^{N-k}(XY) - E^{N-k}(X)\cdot E^{N-k}(Y)]^2}{E^{N-k}(X^2) - E^{N-k}(X)^2}]\} \underset{B.7}{=}$$

$$E(Y^2) - E(Y)^2 - \frac{N-k}{N}\cdot\frac{[E^{N-k}(XY) - E^{N-k}(X)\cdot E^{N-k}(Y)]^2}{E^{N-k}(X^2) - E^{N-k}(X)^2} \qquad \text{B.15}$$

The Hessian is positive definite iff all three leading principal minors are positive /26/. That is:

$$\frac{\partial N\cdot MSE}{\partial x_o^2} > 0, \begin{Vmatrix} \frac{\partial N\cdot MSE}{\partial x_o^2} & \frac{\partial N\cdot MSE}{\partial x_0\partial y_0} \\ \frac{\partial N\cdot MSE}{\partial x_0\partial y_0} & \frac{\partial N\cdot MSE}{\partial y_o^2} \end{Vmatrix} > 0, \text{ and } \begin{Vmatrix} \frac{\partial N\cdot MSE}{\partial x_o^2} & \frac{\partial N\cdot MSE}{\partial x_0\partial y_0} & \frac{\partial N\cdot MSE}{\partial x_0\partial a} \\ \frac{\partial N\cdot MSE}{\partial x_0\partial y_0} & \frac{\partial N\cdot MSE}{\partial y_o^2} & \frac{\partial N\cdot MSE}{\partial y_0\partial a} \\ \frac{\partial N\cdot MSE}{\partial x_0\partial a} & \frac{\partial N\cdot MSE}{\partial y_0\partial a} & \frac{\partial N\cdot MSE}{\partial a^2} \end{Vmatrix} > 0$$

To verify this we first calculate the partial second derivatives at the optimum point:

$$\frac{\partial N\cdot MSE}{\partial x_0^2} = 2\cdot a^2\cdot(N-k)$$

$$\frac{\partial N\cdot MSE}{\partial x_0\partial y_0} = -2\cdot a\cdot(N-k)$$

$$\frac{\partial N\cdot MSE}{\partial x_0\partial a} = 2\cdot(N-k)\cdot[E^{N-k}(Y) - y_0 - 2\cdot a\cdot(E^{N-k}(X) - x_0)] \underset{B.11}{=}$$

$$-2\cdot a\cdot(N-k)\cdot(E^{N-k}(X) - x_0)$$

$$\frac{\partial N\cdot MSE}{\partial y_0^2} = 2\cdot N$$

$$\frac{\partial N\cdot MSE}{\partial y_0\partial a} = 2\cdot(N-k)\cdot(E^{N-k}(X) - x_0)$$

$$\frac{\partial N \cdot MSE}{\partial a^2} = 2\cdot(N-k)\cdot[E^{N-k}(X^2) - 2\cdot E^{N-k}(X)\cdot x_0 + x_0^2]$$
$$= 2\cdot(N-k)\cdot[E^{N-k}(X^2) - E^{N-k}(X)^2 + (E^{N-k}(X) - x_0)^2]$$

The first principal minor $2\cdot a^2\cdot(N-k)$ is obviously positive, unless k=N or a=0.

The second is equal to $\begin{vmatrix} 2\cdot a^2\cdot(N-k) & -2\cdot a\cdot(N-k) \\ -2\cdot a\cdot(N-k) & 2\cdot N \end{vmatrix} = 4\cdot a^2\cdot k\cdot(N-k)$ and is positive under the same conditions, plus k>0. Finally, the determinant of the Hessian itself is equal to

$$\begin{vmatrix} 2\cdot a^2\cdot(N-k) & -2\cdot a\cdot(N-k) & -2\cdot a\cdot(N-k)\cdot(E^{N-k}(X)-x_0) \\ -2\cdot a\cdot(N-k) & 2\cdot N & 2\cdot(N-k)\cdot(E^{N-k}(X)-x_0) \\ -2a(N-k)(E^{N-k}(X)-x_0) & 2(N-k)(E^{N-k}(X)-x_0) & 2(N-k)[E^{N-k}(X^2)-E^{N-k}(X)^2+(E^{N-k}(X)-x_0)^2] \end{vmatrix} =$$
$$\begin{vmatrix} 0 & 2\cdot a\cdot k & 0 \\ -2\cdot a\cdot(N-k) & 2\cdot N & 2\cdot(N-k)\cdot(E^{N-k}(X)-x_0) \\ -2a(N-k)(E^{N-k}(X)-x_0) & 2(N-k)(E^{N-k}(X)-x_0) & 2(N-k)[E^{N-k}(X^2)-E^{N-k}(X)^2+(E^{N-k}(X)-x_0)^2] \end{vmatrix} =$$
$$-2\cdot a\cdot k\cdot\begin{vmatrix} -2\cdot a\cdot(N-k) & 2\cdot(N-k)\cdot(E^{N-k}(X)-x_0) \\ -2\cdot a\cdot(N-k)\cdot(E^{N-k}(X)-x_0) & 2\cdot(N-k)\cdot[E^{N-k}(X^2)-E^{N-k}(X)^2+(E^{N-k}(X)-x_0)^2 \end{vmatrix} =$$
$$8\cdot a^2\cdot k\cdot(N-k)^2\cdot[E^{N-k}(X^2)-E^{N-k}(X)^2]$$

It is positive under the same conditions, plus the condition of positive variance of X, not satisfied only if X is a constant.

Therefore the optimum obtained is indeed the minimum. If the basic assumption ($X_k \le x_0 < X_{k+1}$) is also met, then the minimum is achieved inside, otherwise the minimum of MSE for the given interval is at $x_0 = X_k$ or at $x_0 = X_{k+1}$.

B.3. The minimum is achieved at a sampling point.

Now we assume that $x_0 = X_k$. The minimization involves only two variables and B.11 reduces to

$$\begin{cases} E^{N-k}(XY) - E^{N-k}(X)\cdot y_0 - X_k \cdot E^{N-k}(Y) + X_k \cdot y_0 = \\ \quad a\cdot[E^{N-k}(X^2) - 2\cdot X_k \cdot E^{N-k}(X) + X_k^2] \\ E(Y) - y_0 = a\cdot\dfrac{N-k}{N}\cdot[E^{N-k}(X) - X_k)] \end{cases} \qquad \text{B.16}$$

The solution to B.16 and the problem itself is:

$$x_0 = X_k$$

$$a = \frac{E^{N-k}(XY) - E^{N-k}(X)\cdot E^{N-k}(Y) + \dfrac{k}{N}\cdot[E^{N-k}(X) - X_k]\cdot[E^{N-k}(Y) - E_k(Y)]}{E^{N-k}(X^2) - E^{N-k}(X)^2 + \dfrac{k}{N}\cdot[E^{N-k}(X) - X_k]^2} \qquad \text{B.17}$$

$$y_0 = E(Y) - a\cdot\frac{N-k}{N}\cdot[E^{N-k}(X) - X_k]$$

MSE can be calculated according to B.10.

Finally, we must verify that the Hessian is positive definite. Indeed

$$\frac{\partial N\cdot MSE}{\partial y_0^2} = 2\cdot N > 0 \text{ and}$$

$$\begin{Vmatrix} \dfrac{\partial N\cdot MSE}{\partial y_0^2} & \dfrac{\partial N\cdot MSE}{\partial y_0 \partial a} \\ \dfrac{\partial N\cdot MSE}{\partial y_0 \partial a} & \dfrac{\partial N\cdot MSE}{\partial a^2} \end{Vmatrix} =$$

$$= 4\cdot(N-k)\cdot\{N\cdot[E^{N-k}(X^2) - E^{N-k}(X)^2] + k\cdot[E^{N-k}(X) - X_k]^2\} > 0$$

B.4 The Algorithm

Now we can formulate the algorithm giving the solution to our problem.

1. Calculate $E(X), E(Y), E(XY), E(X^2), E(Y^2)$ and initialize the current minimal MSE according to B.9; set k=0
2. k=k+1; if k=N backtrack the values of x_0, y_0, a to where the current minimal MSE was achieved and exit, otherwise update $E^{N-k}(X)$, $E^{N-k}(Y)$, $E^{N-k}(XY)$, $E^{N-k}(X^2)$, $E^{N-k}(Y^2)$, $E_k(Y)$, $E_k(Y^2)$ according to B.5-B.7
3. Find x_0 according to B.14; if $x_0 < X_k$ or $x_0 > X_{k+1}$ go to the next step, otherwise find MSE according to B.15 and compare it with the current minimal MSE; if there is no improvement go to step 2, otherwise update the current minimal MSE, calculate a and y_0 according to B.13 and B.12 and save all three parameters for future use. Go to step 2.
4. Assign $x_0 = X_k$; calculate a and y_0 according to B.17, calculate MSE according to B.10 and compare it with the current minimal MSE; if there is no improvement go to step 2, otherwise update the current minimal MSE and save all three parameters for future use. Go to step 2.

Note that if the minimum was achieved at a boundary point, we check only one them, the left one. The right point will be checked, if necessary, on the next step.

It proved to be more reliable to calculate MSE always according to B.10, since using B.9 or B.15 can give negative values for the sum of squares. This happens due to

the inprecise nature of computer calculations. These expressions can be calculated in a robust way, i.e., sort the numbers in the ascending order and start summation from the lowest ones. Unfortunately, we are loosing the benefits of faster computation of $E^{N-k}(X)$, $E^{N-k}(Y)$, $E^{N-k}(XY)$, $E^{N-k}(X^2)$, $E^{N-k}(Y^2)$, $E_k(Y)$, $E_k(Y^2)$, since they must be recalculated at each step. Therefore this procedure does not give any computational advantages.

References

1. Flury, B. (1988) Common Principal Components and Related Multivariate Models. Wiley & Sons, New York.
2. Jackson, D.A. (1993) Stopping Rules in Principal Components Analysis: a Comparison of heuristical and statistical approach. Ecology 74(8), pp 2204-2214
3. Dey, D.K., Srinivasan, C., (1985), Estimation of a covariance matrix under Stein's loss. The Annals of Statistics. 13(4), pp 1581-1591.
4. Golub, G.H., Van Loan, C.F. (1983), Matrix Computations, Baltimore: John Hopkins University Press.
5. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. (1992). Numerical Recipes in C. Cambridge University Press.
6. McLachlan, G.J. (1992) Discriminant analysis and statistical pattern recognition, Wiley & Sons, New York.
7. Lachenbruch, P.A.(1967) An almost unbiased method of obtaining confidence intervals for the probability of misclassification in discriminant analysis. Biometrics, 23, pp 639-645.
8. IMSL User's Manual. (1987) pp 695-713.
9. Friedman J.H., (1989) Regularized Discriminant Analysis, J. Amer. Stat. Association. 84(405), pp 165-175.
10. Aeberhard, S., Coomans, D., De Vel, O. (1993) Improvements to the classification performance of RDA., J of Chemometrics, 7, pp 99-115.
11. Rayens, W., Greene, T., (1991). Covariance pooling and stabilization for classification. Computational Statistics & Data Analysis, 11, pp 17-42.
12. Holland, J.H. (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

13. Davis, L. (1991) Handbook of genetic algorithms. Van Nostrand Reinhold, New York.

14. Goldberg, D.E., Richardson, J. (1987) Genetic algorithms with sharing for multimodal function optimization. Proc. of the 2nd International Conference on Genetic Algorithms, Cambridge, MA.

15. Lin, C.-Y., Hajela, P. (1994) Design optimization with advanced genetic search strategies. Advances in Engineering Software 21, pp 179-189.

16. Smith, R.E., Smuda, E. (1995) Adaptively Resizing Populations: Algorithm, Analysis, and First Results. Complex systems 9 pp 47-72.

17. Naitoh, K. (1995) Four-Group Equation of Genetic Algorithm. JSME International Journal, Series C, 38, pp 240-248.

18. Siedlecki,W., Sklansky, J. (1989) A note on genetic algorithms for large-scale feature selection. Pattern Recognition Letters 10, pp 335-347.

19. Prakash, M., Narasmha Murty, N. (1995) A genetic approach for selection of (near-) optimal subsets of principal components for discrimination. Pattern Recognition Letters 16, pp 781-787.

20. Strarcuk, Z. Jr., Bartusek, K., Starcuk, Z. First-Data-Point Problem and the Baseline Distortion in Fourier-Transform NMR Spectroscopy with Simultaneous Sampling. J. Magn. Res. Series A 108, pp 177-188.

21. Gesmar, H., Led, J.J., Abildgaard, F. (1990) Improved Methods for Quantitative Spectral Analysis of NMR Data. Progress in NMR Spectroscopy, 22, pp 255-288.

22. Hoult, D.I., Chen, C.N., Eden, H. (1983) Elimination of Baseline Artifcts in Spectra and Their Integrals. J. Mag. Resonance 51, pp 110-117.

23. Proakis, J.G., Dimitris, G. (1992) Digital signal processing: principles, algorithms, and applications. Macmillan, New York.

24. Somorjai,RL; Pizzi,N; Nikulin,A; Jackson,R; Mounyford,CE; Russel,P; Lean,CL; Delbridge,L and Smith,ICP (1993): Thyroid Neoplasms: Classification by Means of

Consensus Multivariate Analysis of ^{1}H MR Spectra. Proceedings of the Society of Magnetic Resonance in Medicine, Twelfth Meeting, p72.

25. Nikulin,A; Brière,KM; Friesen,L; Smith,ICP and Somorjai,RL (1995): Genetic Algorithm-Guided Optimal Attribute Selection: A Novel Preprocessor for Classifying MR Spectra. Proceedings of the Society of Magnetic Resonance, Third Meeting, p1940.
26. Horn,R.A., Johnson,C.R. (1985) Matrix analysis. Cambridge University Press, Cambridge.
27. Lucasius, C.B., Kateman, G. (1991) Genetic algorithms for large-scale optimization in chemometrics: an application. Trends in analytical chemistry, 10, pp 254-261.
28. Kuncheva, L. (1993) Genetic algorithm for feature selection for parallel classifiers. Information Processing Letters 46, pp 163-168.
29. Kohavi, R. (1995) Feature Subset Selection Using the Wrapper Method: Overfitting and Dynamic Search Space Technology.
30. Bangalore, A.S., Shaffer, R.E., Small, G.W (1996) Genetic Algorithm-Based Method for Selecting Wavelengths and Model Size for Use with Partial Least-Squares Regression: Application to Near-Infrared Spectroscopy. Anal. Chem. 68, pp 4200-4212.
31. Miller, A.J. (1990) Subset Selection in Regression. Chapman and Hall. Great Britain.
32. Goldberg, D.E. (1989) Genetic Algorithm in Search, Optimization, and machine Learning. Addison-Wesley, Reading, MA, USA.
33. Goldberg, D.E. (1990) A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing. Complex Systems 4, pp 445-460.
34. Lin, F.-T., Kao, C.-Y., Hsu, C.-C. (1993) Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Problems. IEEE Transactions on Systems, Man, and Cybernetics 23, pp 1752-1767.

35. François, O. (1996) Convergence in Simulated Evolution Algorithms. Complex Systems 10, pp 311-319.

36. Davis, T.E., Principe, J.C. (1993) A Markov Chain Framework for the Simple Genetic Algorithm. Evolutinary Computation 1, pp 269-288.

37. Miller, J.A., Potter, W.D., Gandham, R.V., and Lapena, C.N. (1993) An Evolution of Local Improvement Operators for Genetic Algorithms. IEEE Transactions on Systems, Man, and Cybernetics 23, pp 1340-1351.

38. Lucasius, C.B., Kateman, G. (1993) Understanding and using genetic algorithms. Part 1. Concepts, properties and context. Chemometrics and Intelligent Laboratory Systems, 16, pp 1-33.

39. Van den Boogaart, A., Ala-Korpela, M., Jokisaari, J., Griffiths, J.R. (1994) Time and Frequency Domain Analysis of NMR Data Compared : An Application to 1D ^{1}H Spectra of Lipoproteins. Magn. Reson. Med. 31, pp 347-358.

40. Pijnappel, W.W.F., van den Boogaart, A., de Beer, R., van Ormondt, D. (1992) SVD-based quantification of magnetic resonance signals. J. Magn. Reson. 97, pp 122-134.

41. Van der Veen, J.W.C., de Beer, R., Luyten, P.R., van Ormondt, D. (1988) Accurate quantification of *in vivo* ^{31}P NMR signals using the variable projection method and prior knowledge. Magn. Res. Med. 6, pp 92-98.

42. Hiltunen, Y., Ala-Korpela, M., Jokisaari, J., Eskelinen, S., Kiviniitty, K., Savolainen, M., Kesäniemi, Y.A. (1991) A lineshape fitting model for ^{1}H NMR spectra of human blood plasma. Magn. Res. Med. 21, pp 222-232.

43. Koehl, P., Ling, C., Lefèvre, J.F. (1994) Statistics and Limits of Linear Predicting Quantification of Magnetic Resonance Spectral Parameters. J. Magn. Res. Series A 109, pp 32-40.

44. Hastie, T., Tibshirani, R. (1996) Discriminant Analysis by Gaussian Mixtures. J. R. Statist. Soc. B 58, pp 155-176.

45. Fukunaga, K., Short, R.D. (1980) A Class of Feature Extraction Criteria and Its Relation to the Bayes Risk Estimate. IEEE Transactions on Information Theory IT-26, pp 59-65.

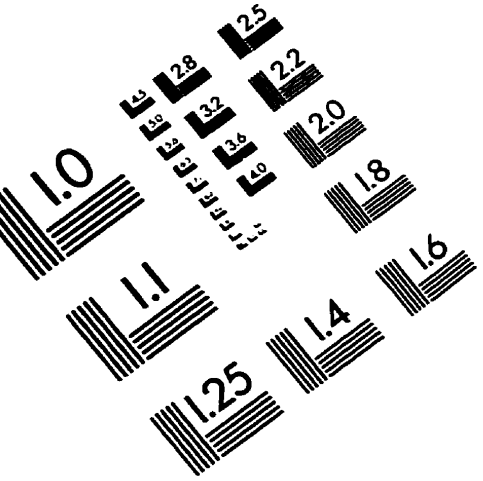
46. Krishnaiah, P.R., Kanal, L.N. (1982) Handbook of Statistics. Vol. 2. North-Holland Publishing Company. Amsterdam.

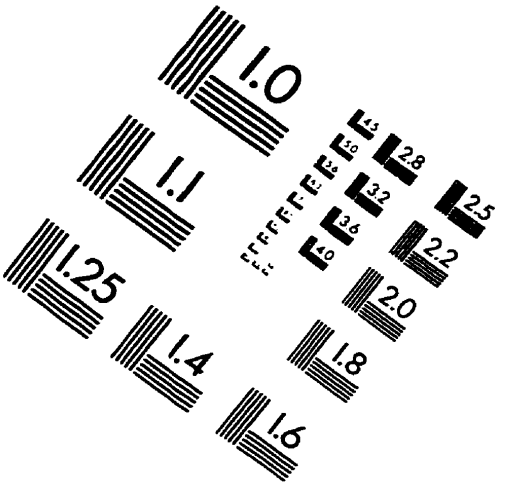
47. Young, T.Y., Fu, K.-S. (1986) Handbook of Pattern Recognition and Image Processing. Academi Press, Inc. Orlando Florida.

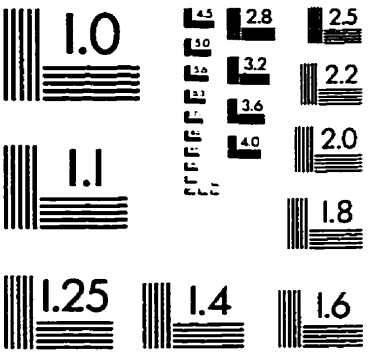
48. CSS : Statistica. Volume I (1991) StatSoft, Inc.

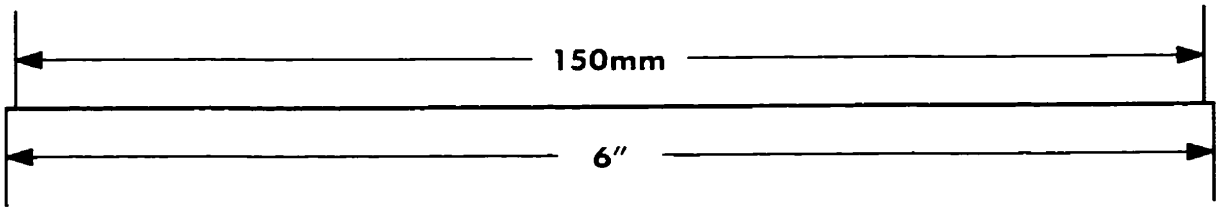
49. Nikulin, A.E., Dolenko, B., Bezabeh, T., Somorjai, R.L. (1998) Near-optimal Region Selection for Feature Space Reduction: Novel Preprocessing Methods for Classifuing MR Spectra. NMR in Biomedicine, 11 (1-8).

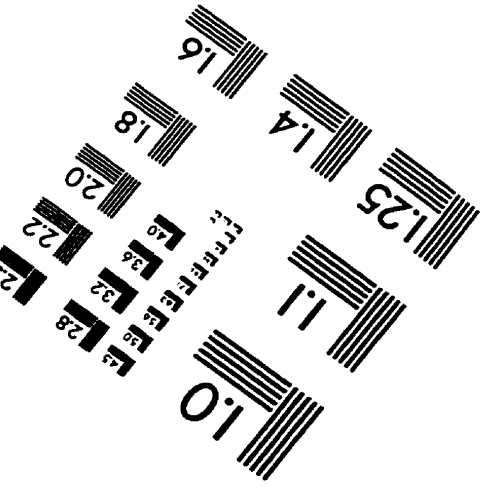
IMAGE EVALUATION
TEST TARGET (QA-3)

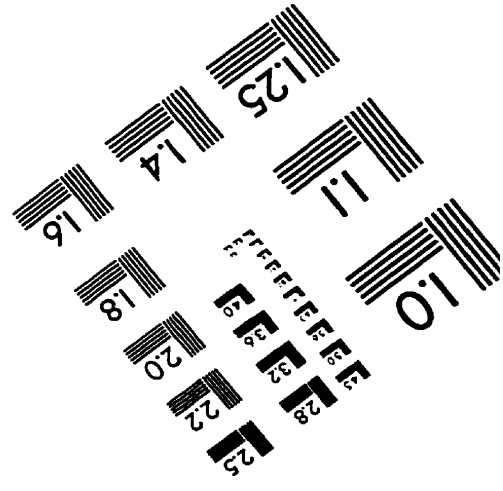












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989