Application of genetic algorithms to the analysis of microarray gene expression data

Magnus Alm Rosenblad

June 25, 2001

Abstract

New techniques of measuring the activity of thousands of genes simultaneously are revolutionizing research in biology and medicine. The commonly used analytical methods for grouping genes to distinguish between different kinds of samples are not, however, informative for finding direct relationships or simple rules for prediction. After performing a survey of the methods used in research so far, a different approach was tried in this thesis. This approach employs a method for finding explicit relationships between genes, that can be used directly for predicting phenotypes or as a classification tool. The rules may also be used as a pool of possible relationships to test further.

The method, based on genetic algorithms, is non-deterministic and states the result as rules consisting of a set of conditions based on the expression averages of some genes. The search is conducted with a population of rules that are subjected to an artificial evolutionary process until the conditions can be met and the samples are correctly classified. The expression values used as input to the search algorithm are trinerized, as the significance of the exact level of the absolute expression values and ratios is unclear and much debated.

Contents

Chapter 1

Introduction

The ability to simultaneously measure the activity of thousands of genes poses completely new challenges to researchers because of the wealth of information. An experiment may already today include 10,000 genes and generate more than a million datapoints. The number of possible interactions is so large that new means of discovering the important ones must be found.

The purpose of this master's thesis is to find and test a new method extracting relevant genes, with the aim of formulating easily interpretable rules for both sample classification and possible gene interactions. The classifier is formulated as a boolean function where a small set of input genes determines the output value, for instance "1" for a tumor sample and "0" if normal. But even if the expression values of the genes are idealized to being just 0 or 1, there exists 2^{2^n} possible functions for n input genes, making an exhaustive search intractable already with $n = 6$ genes as input.

A special problem with this kind of data is the high noise level, making some algorithms hard to implement (for instance Fourier analysis with discrete values), so what is needed is a search algorithm that can handle the noise and search vast spaces. Because of the uncertainty in how to interpret the data from microarray experiments, a way of filtering the data to limit the possible expression values is also used.

The method used in this thesis is now being further developed at Chalmers (at the Mechatronics division), in collaboration with cancer researchers.

The following two chapters explain how DNA microarrays work, the statistical problems that are raised when interpreting the gene expression data, and present an overview of research in which microarrays have been used. For a summary, please see Table 3.1 (page 12), Table 3.2 (page 13), Table 2.4.3 (page 8), Table 3.3 (page 13).

Clustering is often used to reduce the dimension of the feature space. However, such methods are not altogether optimal for our purposes. As an example, consider a case of 5,000 genes clustered into 5 clusters: each cluster may contain of order 1,000 genes, making it difficult to identify the interesting ones.

Genes that are causative or, in any case, strongly related to the condition under study, should show more consistent *misregulation* than genes that are only triggered as a result of the condition. For example, (small) variations in the environment may cause loosely related genes to take different expression values in different individuals, whereas this is less likely in the case of genes that are strongly coupled to the condition.

Acknowledgement I would like to thank my supervisor, assistant prof. Mattias Wahde, who provided the enthusiasm needed for the completion of the thesis. Dr. Zoltan Szallasi (USUHS, Bethesda, USA) has been helpful in the discussions about what data sets to use, how to discretize the data and in the biological evaluation of results.

Contact information: If you have any questions about this work, please contact
Magnus Alm Rosenblad[1]
(magnus.alm@lundberg.gu.se),
Mattias Wahde
(mwahde@me.chalmers.se).

[1] This thesis project was carried out during the fall and winter 2000–2001. It completes my master of science degree in Computer Science and Engineering, at Chalmers University of Technology, Sweden.
– *Magnus Alm Rosenblad*, Göteborg, June 2001.

Chapter 2

Gene Expression and Microarrays

2.1 Genes, RNA and Proteins

The living cell produces RNA which is transcribed from the genes active in the DNA, the RNA may be used directly or as a template (after splicing in eukaryotic cells) for production of proteins (translation) which are composed of amino acids. (Each triplet of nuclein acids is equivalent to a specific amino acid, nearly all of them may be coded by several different triplets. There are 20 amino acids and 3 stop codons, but $4^3 = 64$.)[1]

The production of the different RNAs depends on several factors: the species, the cell type (muscle, neuron ...), the evolutionary phase of the organism, the cell cycle state. The majority of the genes are active in all cell types of which there are approximately 50 in humans, and just a single one in bacteria.

If the production of RNA could be extensively monitored or tested, it would be possible to do many things, and DNA microarrays are a promising new biotechnology which in fact allows this monitoring of thousands of genes simultaneously. It is being applied increasingly in biological and medical research to address a wide range of problems, such as the classification of tumors, the study of host responses to bacterial infections, or even reverse engineering of genetic networks.

2.2 Measuring Expression

An active gene is said to be *expressed*, and the result, i.e. the amount of RNA produced, can be measured by techniques such as *microarrays*. An RNA molecule will bind (*hybridize*) to a complementary molecule, so with many of these complementary molecules it is thus possible to measure the amount of RNA produced, which is a measure of the gene expression. The method used for measuring consists of (1) labeling the sample molecules with a fluorescent dye, (2) applying the sample to the microarray and then, after getting rid of the sample molecules that do not hybridize, (3) scanning (photographing) the array now containing light-emitting hybridized molecules, to create a digital image in which the intensities can be measured.[2] In *cDNA* arrays the reference sample gets green dye (Cy3) and the target sample gets red dye (Cy5), whereas in the *oligonucleotide*[3] arrays there is just one dye (see below).

The number of genes that can be put on a chip is rapidly increasing. As of early 2001 the chips contain approx. 13,000 genes, the publicly available data sets ranging between 6,000 to 8,000. The current calculations of the number of genes in the human genome is <30,000 (down from earlier approximations of >100,000), so very soon there will be arrays that contain the whole human genome.

2.2.1 The Genes on the Array

The arrays now manufactured[4] have different sets of genes (human, rat etc.), and researchers that make their own chips (cDNA, see below) can of

[1] For details on the biology and biochemistry please see a standard textbook like *Molecular biology of the cell* by Alberts and others.

[2] Most papers on microarrays explain the technology, see for instance Chen et al. 1997 (cDNA microarrays).

[3] cDNA stands for *complementary* DNA and is a reverse transcription of *mature* RNA. In organisms other than bacteria, this mature RNA is a stripped down (*spliced*) version without all the non-protein-coding segments of the original RNA transcribed from DNA. An oligonucleotide is simply a shorter segment of RNA.

[4] See last section in Bibliography for suppliers. Affymetrix chips currently sold: Human: Human Genome U95 Array, HuGeneFL Array; Rat: Rat Genome U34 Set, Rat Toxicology U34 Array, Rat Neurobiology U34 Array; Mouse: Murine Genome U74 Set; Arabidopsis: Arabidopsis Genome Array; Drosophila: Drosophila Genome Array; Yeast: Yeast Genome S98 Array; E. coli: E. coli Genome Array; Cancer: P53 Assay; Neurobiology: Rat Neurobiology U34 Array; Toxicology: Rat Toxicology U34 Array; Genotyping: HuSNPTM Mapping Assay, GenFlexTM Tag Array, CYP450 Assay.

course make a unique selection that fits the problem at hand. Genes known to be related to some human diseases have been put on chips (e.g. the *lymphochip* used by Alizadeh et al.) and this process will surely continue since, with this approach, fewer genes (simpler chips) are needed and replication of genes on chips can be used to get more reliable results (see section on Problems).

2.2.2 Different Types of Arrays

There are several kinds of microarrays (even for proteins[5]), the two main ones being cDNA and oligonucleotide arrays. These differ in several ways, the first being that molecules are directly syntesized on the surface in the oligonucleotide arrays[6] (not put there as in cDNA) and in the way the sample molecules are hybridized to the array targets: in the cDNA arrays the *probes* (spots) consist of molecules that are more or less complete complementaries, in the oligonucleotide arrays there are about 20 probes (oligonucleotides) that have been chosen to uniquely identify the gene. How this is done in detail is however not revealed by the company that markets these arrays (Affymetrix), but the size of the oligonucleotides is 25 nucleotides (nt) and they are taken from the first 600 nucleotides in the gene. To avoid mistakes there are also an equal number of *mismatch* probes, where the nucleotide in position 13 is altered (see Table 2.1, page 7).

Another difference is that the cDNA arrays always measure two samples at the same time, that is the samples are both applied to the same array, while in the oligonucleotide arrays one just measures one sample at a time. Therefore one has to specify a "baseline" array when calculating ratios (ratios are called *fold change*), which can be of great help if you do not know from the start which samples are going to be compared. In the cDNA arrays the green intensity and the red intensity are measured and the logarithm (base 2) of the ratio (R/G) is taken to give positive values for increase and negative for decrease.

The calculation of how a gene is considered "present" is also different. In the cDNA arrays the intensity in each channel, green (G) for reference and red (R) for target samples, gives the value and the problem is how to estimate the background intensity and setting a threshold value.

In the Affymetrix arrays, both a *match* and a *mismatch* intensity for each of the 20 probe pairs is measured. The difference between match and mismatch probes decides if there really is a match. Thus, strong "match" signals do not by themselves give the amount of RNA in the sample. Also one has to take into account how many probe pairs that signal a match. The Affymetrix software uses several parameters to calculate "present", for instance the number of positive and negative probe pairs. A probe pair is considered positive if the difference between the matching intensity and mismatching intensity is greater than a user specified threshold, and (after background subtraction) the ratio of matching and mismatching intensities are greater than a user specified threshold. More parameters like these are weighted together, but unfortunately Affymetrix does not clearly state how this is done.

2.2.3 How to Make the Molecules – Amplification

Large libraries of genes and parts of genes (ESTs) now exist, and these serve as references when detecting RNA in samples. Since each RNA molecule just hybridizes to one complementary, lots of molecules are needed to bind all the RNAs in a sample. For each gene there are about 10,000,000 molecules per probe (spot) on the cDNA array, in the oligonucleotide arrays the probes consist of 10,000,000 molecules each. To produce all these copies *polymerase chain reaction* (PCR) is used, a revolutionary technology introduced in 1987 that avoided the previous need to transfer DNA into living cells for replication. From a double stranded sequence that is divided into single strands, the complementaries to each strand are formed by polymerase chain extension. The two double stranded molecules are then broken up and the process repeated, thus doubling the number of molecules for each cycle. After 25 cycles there are more than 10 million copies.

When analyzing a sample one might find an insufficient amount[7] of some RNAs (as little as a single molecule per cell is not rare, so many cells are

[5] MacBeath and Schreiber, *Science*, Vol. 289, Sept. 2000.

[6] The oligonucleotides used for gene identification are called probes. Probe arrays are manufactured by Affymetrix's proprietary, light-directed chemical synthesis process, which combines solid-phase chemical synthesis with photolithographic fabrication techniques employed in the semiconductor industry. Using a series of photolithographic masks to define chip exposure sites, followed by specific chemical synthesis steps, the process constructs high-density arrays of oligonucleotides, with each probe in a predefined position in the array. The probe positions used to be according to gene, but were changed in 2000 to a more random positioning, to reduce the effects of variable local hybridization.

[7] Affymetrix has introduced special test arrays, so researchers can identify degraded samples containing insufficient target that would result in poor expression array results.

needed), and to get more one uses *amplification* methods operating in a similar manner to PCR. But the RNAs in the sample respond nonlinearly to the amplification process, and thus one cannot compare arrays prepared from amplified and unamplified samples. This means that one has to double the runs in order to get an amplified and an unamplified line of arrays if both are being used.

2.2.4 Fluorescence into Numbers

When the spots are scanned the resulting image is made out of pixels with varying intensity, the number of pixels that make up each spot is a consequence of the resolution of which the chip is scanned. As can be seen in the enlarged (and exaggerated) picture (Fig. 2.1 on page 7), each spot does not contain that many pixels at the currently used resolutions. The calculations are basically done by first removing the background intensity (the intensity of pixels outside the spots), then choosing a shape and calculating the average intensity in that area.

2.2.5 Reliability of the Measured Levels

The first tests with microarrays dealt with the basic question whether the expression values at all could be said to correlate with known functions of the genes measured, for instance the activity of a known gene in a normal cell and a tumor cell (see Chu et al. 1998, DeRisi et al. 1997). This was found to be the case, hence the excitement over the new technique. But the purpose of experiments now is not to confirm known properties, but rather acquire information about unknown genes and functions. It is becoming clear that a comparison of exact values (or ratios) can not be easily used in the analysis. Genes with low expression values are particularly troublesome as they often get high fold differences and the effects of noise on measurements are worse at low expression levels.

Although DNA microarrays have been used for more than four years, a major issue is still how to define a differentially expressed gene, i.e. a gene that has a significantly changed expression in different samples. Some problems are discussed in the section below and a steadily increasing number of papers deal with this issue (see Table 2.4.3, on page 8 for an overview).

2.3 Microarray Problems

Obviously, the process of preparing the samples is of great importance. The type of cell or cells that are sampled as well as the current state of the cell regarding the cell cycle and the evolution of the organism, influence the amounts of RNA. For instance, as can be seen in the paper by Alon et al. (1999), if the "abnormal" cells more often come from a specific type of tissue, the abnormal expression profile will be biased so that genes highly expressed in that tissue become good separators of normal and abnormal tissue. (To level out the effects samples are sometimes *pooled*, that is mixed together.) Below the statistical issues of measurements are discussed.

2.3.1 Statistics and Problems

DNA microarrays (cDNA or oligonucleotide) also raise numerous statistical questions in fields as diverse as image analysis, experimental design, multiple testing, discriminant analysis and cluster analysis etc. The key question is how to identify differentially expressed genes, i.e. genes whose expression levels are associated with a response or covariate of interest (treatment/control status, cell type, drug type, dose of a drug, survival times or other clinical outcomes).

An overview of the problems could, for a cDNA experiment, be:

Biological questions:
Differential expression, Sample class prediction.
$\Downarrow$
Experimental design
(of the array, of the experiments, replication)
$\Downarrow$
Microarray experiment
($\Rightarrow$ 16 bit image)
$\Downarrow$
Image analysis
($\Rightarrow$ Values for [Rfg, Rbg] [Gfg, Gbg])
$\Downarrow$
Normalization
($\Rightarrow$ values for each spot on each array)
$\Downarrow$
Data analyis:
Discrimination + Clustering + ...etc.
$\Downarrow$
Biological verification and interpretation
$\Longrightarrow$ REPEAT

The details of image analysis and normalization for cDNA arrays:

1. Image analysis

   - Segmentation: Classify each pixel as foreground or background. Spots have varying sizes and shapes. (Manual work.)
   - Background correction: For each spot remove background. Few pixels lead to incorrect estimation of pixel identity.
   - Intensity extraction: Foreground, background: A good way is to use a non-linear filter that generates an image of the estimated background intensity. More important than shape and size of spots.
   - Spot quality measures: Weights on quality of spots.

2. Normalization

   - Identify and remove systematic sources of variation: Different labeling (Cy3 and Cy5 give different result); Different amounts of Cy3 and Cy5; Different scanners; Print tip group (a cDNA array is divided into at least four groups each printed by different print tip).
   - MA-plots: M = $log_2 R/G$; A = $log_2\sqrt{RG}$, gives non-linear pattern.
     Within-slide location normalized (2x2 grid), each tip has got its own line.
   - Location and scale normalization.
   - Normalization (i) within slide, (ii) for paired slides in dye-swap experiment: self-normalization, (iii) between slides.
   - Which spots to use: all spots; constantly expressed genes ("house keeping genes"); controls: spike controls, titration series.

Discussion: Preliminary data processing steps (image analysis, normalization ...) can obviously have a large impact on final expression values. Proper experimental design is very important. Since there is no easy way to tell differentially expressed genes from a single slide, *replication* is necessary. Microarray experiments means an extreme multiple testing situation: adjustment needed to control false positive rates.

Genes that are known to behave in a special way and therefore can be used for calibrating the other genes' expression values are called "house keeping genes". However, nobody seems to have found a reliable set of house keeping genes yet.

2.3.2 Replication Is Necessary

At a seminar series (Chalmers/GU, 2001) the problem of the reliability of the expression values from DNA microarrays was discussed. An example given was a test in which Affymetrix arrays were used (at AstraZeneca, Gbg) and the target and reference samples were the same, e.g. one sample was divided into two. The target and reference arrays should have given the same expression values, but this was not the case:

- 13,000 genes on chip
- Remove all NC ("Not Changed" by Affymetrix standards): 115 genes left
- Remove all A ("Absent" by Affymetrix standards): 65 genes left (should be 0)

Genes that have a very low expression may of course have a big relative change due to noise in the measurements, so remove genes with low absolute difference in expression.

Keep genes with ABS Avg diff ("Absolute average difference") change greater than a certain value: still 18 genes left (false positives).

The "fold change" was then raised from the minimum value 2.0 (regarded by Affymetrix as a minimum for stating that a change has occurred):

- Keep those with ABS Fold change $> 2 = 18$ genes as above
- Fold change threshold $> 3 = 3$ genes left
- Fold change threshold $> 3.9 = 0$ genes left (Now we have no false positives.)

This test revealed that even though samples are identical, the values differ substantially. This means replication must be used to avoid false positives.

Lee et al. (2000) and others also stress replication as essential. Kerr and Churchill advocate both multiple spotting of genes and two arrays, and more arrays if not multiple spotting is possible, something that is supported by Dudoit (at seminar at AstraZeneca, Gbg, April 2001). But how much? No clear answers exist at the moment.

2.3.3 Statisticians–Biologists

When working with gene expression values, people tend to have different views on the data. In the words of statistician Rob Tibshirani at Stanford: "The biologists do not want to miss anything (low type II error), the statisticians have to teach them

an appreciation of type I error (false positives)." Assembling groups of people from different departments are therefore a good practice when making studies in these areas.

2.4 Differential Expression

Assuming that the expression values have been adjusted in some way so that they can be used for calculating the difference between two samples, what is the minimum change required, and how shall we value the fold change magnitude?

2.4.1 Fold Change Factors

When calculating fold changes, ratios for genes in two samples, one cannot directly divide the expression values as this tends to give genes with low values rather high ratios. The absolute change difference should also be taken into account and a minimum threshold value be specified, thus eliminating genes that get high ratios despite the change in values being just noise, that is within the expected error for the measurement.

Next, the fold change threshold must be set: Affymetrix states factor 2.0 as being the minimum value, but the relevance in differing 3 or 4 can be questioned. This is an argument for discretizing (binning) the values, something often done in reverse engineering. Several papers on fold change have been published, see Table 2.4.3 for examples.

Kerr and Churchill (2001) stress that large changes in gene expression is not the only thing to look for as important genes may have small, but reproducible, changes in expression. In order to detect such genes, scientists need statistically designed experiments and data analysis that together with estimates of relative expression also produces error bars for those estimates. Error bars provide a basis to decide which features in the data likely represent interesting biology and which are likely to have arisen by chance, according to Kerr. Measures of confidence should also be incorporated.

2.4.2 Trinerization

In reverse engineering of genetic networks the gene values are often discretized and most often to a binary ON or OFF (1 or 0). In microarray experiments the values are continuous, but since the meaning of those values may be discussed there might be a way of using the values in a simpler way. One way of extending the notion of a gene being ON or OFF (1 or 0), is to use trinerized data that equals NO CHANGE, DOWN or UP (0, –1, +1). This preserves the information about the up- or downregulation of a gene while discarding the high ratios that are much above the selected threshold. Not many examples of this exist in the literature however.

2.4.3 The Criteria Used in this Work

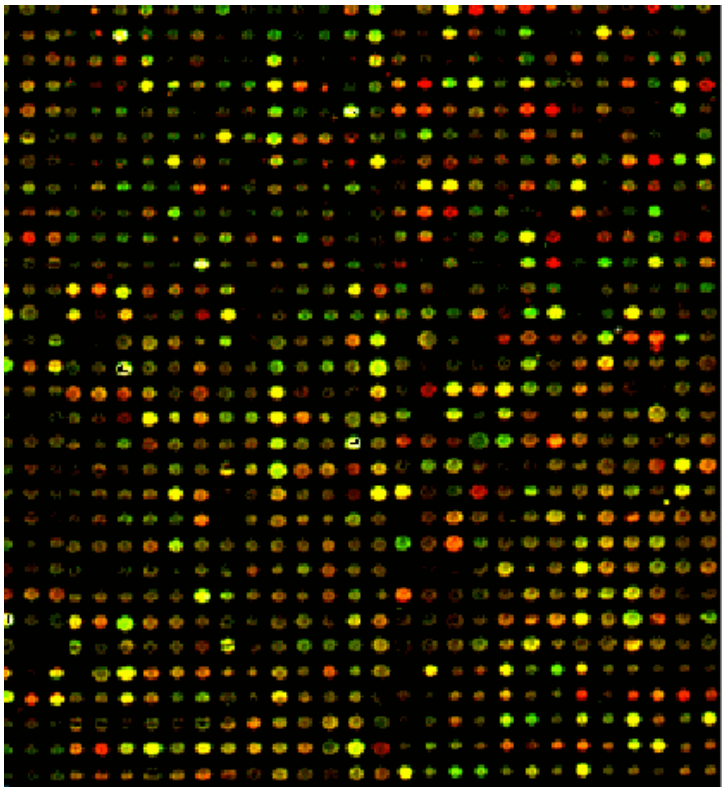
As most of the papers on how to calculate significant fold differences were not published when this project started, a far simpler method has been used. If the values are Affymetrix type "pseudo-absolute", that is they are not ratios but absolute values to be compared to reference values of another array, the gene's average value in the *normal samples* is calculated. This value is then used to determine the fold difference for the abnormal samples (as well as for the normal samples).

The "average" can be the arithmetic mean, but since this is influenced by large and small outliers in the data one may of course trim the data in some way. It is not done in this study, but different ways of obtaining an "average" or baseline value to compare with should naturally be tried in a more detailed study.

The next step is deciding on a factor used in calculating if the fold difference is above or below a threshold for defining a differentially expressed gene. This factor should be at least 2.0 (Affymetrix' minimum value), and both different levels or a slightly higher value (after consultation with an experienced user) have been used.

Since there does not exist a standard way of calculating this threshold and how to interpret fold differences, a simplified (discretized) data set is used where all fold differences over the threshold are considered upregulated (+1) and below downregulated (–1). Others are not changed (0).

This method is also applied to the cDNA data, since the values in the two abnormal sample sets are ratios compared with normal samples (Cutaneous melanoma data set, Bittner et al.). One of the abnormal sets then is considered normal and the values can be used as absolute values as in the first example. (For details on normalization etc., see the section "Data sets".)



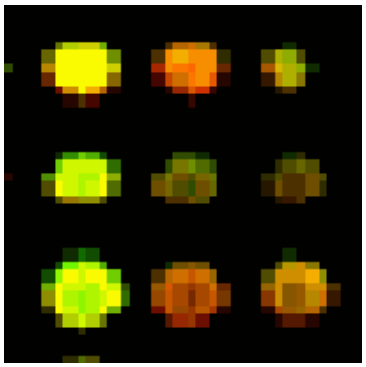


Figure 2.1: *This is an enlargement (and exaggeration) of a cDNA array image. Each spot is made up of quite few pixels, making it quite hard to choose the right values. Both the shape and size of the spots matters, but most important is the correct estimation of the background intensity over the whole image* [Dudoit].

Figure 2.2: *This is a cDNA array image. Intensity values are calculated for both the green and red channel for each spot. If the spot is red, then the gene's target sample contained more mRNA and thus the gene's expression has increased compared with the reference sample, if green then it has decreased. If a spot is yellow, intensities of green and red being fairly equal, the gene's expression has not changed. These cDNA arrays are divided into 4 or more areas, with a different printing tip used for each of them.*

Table 2.1: *Comparison of different microarrays*

	cDNA microarrays	**Affymetrix oligonucleotide**
Molecules on array	Put on array by printing tip	Manufactured directly on array
Kind of molecules	Complete complementaries	20 probe pairs: PM–MM, 25 nt oligonucl. per probe
Samples per array	Reference (green) + target (red)	Target (reference sample run on separate array)
"Gene present"	Intensity, threshold	Comparison of PM and MM
What genes on array?	Decided when manufacturing	Different kinds sold †: Mouse, Human . . . different sets of genes

†See footnote p. 3.

Table 2.2: *Papers on how to define differentially expressed genes.*

Author	**Title (*Comment*)**
Chen et al. (Oct, 1997)	Ratio-based decisions and the quantitative analysis of cDNA microarray images (*Signal calibration.*)
Newton et al. (1999)	On differential variability of expression ratios: Improving statistical inference about gene expression changes ...
Sapir and Churchill	Estimating the posteriori probability of differential gene expression from microarray data
Kerr et al. (July, 2000)	Analysis of variance for gene expression microarray data (*Anova can be used, framework for general analysis.*)
Dudoit et al. (Aug, 2000)	Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments
Lee et al. (Aug, 2000)	Importance of replication in microarray gene expression studies
Kepler (fall, 2000)	Normalization and analysis of DNA microarray data by self-consistency and local regression
Efron et al. (Oct, 2000)	Microarrays and their use in a comparative experiment
Butte et al. (Jan, 2001)	Determining significant fold differences in gene expression analysis
Yang et al. (Jan, 2001)	Normalization for cDNA microarray data
Kerr et al. (2001)	Analysis of a designed microarray experiment + Statistical ...replication (*+ 1 more paper*)
Goss et al. (April, 2001)	Significance analysis of microarrays applied to the ionizing radiation response (*Stanford, fall 2000.*)

Chapter 3

Research with microarrays

So far the biggest data sets consist of 50–100 samples with expression values (absolute values or ratios) for approximately 5,000–10,000 genes. Some data sets are publicly available and there are often several studies using them, each paper examining a different subject. (See Tables 3.1 and 3.2.) For cDNA arrays even the image files are possible to download from the researchers' websites, making a completely new analysis of values possible.

The wealth of gene expression data now available poses several questions ranging from the analysis of the images produced by microarray experiments and the study of the variability of measured gene expression levels (Chen el al. 1997, Newton et al. 1999), to the elucidation of biochemical pathways and reverse engineering.

3.1 Reverse Engineering

Several papers deal with how to *reverse engineer* the interaction of genes by using large-scale gene expression values. In general these papers use time-series, for instance the expression values during a transition of the cell between different states or stages in the cell cycle. A multitude of approaches have been tried[1], including boolean networks, weight matrices and differential equations.

Limiting factors so far have been the low number and the limited length of the available time series, one of the longest used even being created by combination from two data sets (Wen et al. 1998).

3.1.1 Models

The holy grail of analysis (of gene expressions and other variables) is the complete reverse engineering of the complex networks that make up the different cell types.

Some key questions regarding reverse engineering are:

- How many samples, time points, are needed to be able to sketch a wiring diagram?
- What are the variables (expressed genes, protein levels ...) and how many of them are needed?
- What can be assumed about the connectivity between the variables/genes?
- Are there special kinds of boolean functions that are preferred in the wiring?
- Is the network deterministic or stochastic and in what ways?
- Should variables be continuous or discrete?

Examples of models include boolean networks (Liang), weight matrices (Weaver), hybrid petri nets (Matsuno), differential equations (Wahde and Hertz) and many more.

So far the high number of variables and lack of long time series have made reverse engineering difficult, but attempts have been made. The number of samples needed has been investigated[2] and found being proportional to at least $\log n$, where n is the number of variables (genes). With a maximum of 50,000 genes in the human genome and maybe a tenfold increase if the environment in the cell is to be included, n is approaching 500,000, a big number indeed.

The binning (discretizing) of the expression values affects the number of entropy values possible, as explained in Fuhrman et al. 2000.

[1] For a review of genetic network reconstruction work, see Dutilh 1999. For papers, PSB-Online is a good source (Pacific Symposium on Biocomputing, http://psb.stanford.edu/).

[2] Many authors have tackled this, for instance D'haeseleer, Hertz, Akutsu.

3.1.2 Complexity in Genetic Networks

The control of gene activity is very complex as the genes interact with each other (and other substances in the cell).[3] Assuming that a gene may be either ON or OFF, and that a gene takes N inputs, there are

$$2^{2^N}$$

different possible functions of how those N genes turn this gene ON and OFF. This number grows so fast that it quickly becomes computationally intractable to try each one of them. The search space is vast and a special kind of search algorithm must be employed to explore it.[4]

Traditional statistical methods, as correlation etc., do not explicitly state the interaction of several genes, so something different is needed.

If a method for finding simple rules for the interaction could be found, it would greatly facilitate the analysis of the experiments.

3.2 Looking for a Pattern: Predictors and Learning

The data from microarray experiments form a matrix, where the rows are different genes and the columns are different samples. In some experiments the samples are cells from different organisms, or cells from different tumors etc. In other experiments the samples constitute a time series of measurements during different phases of cell development or response to a drug or otherwise.[5]

With more than 5,000 genes and several (maybe 50) arrays or samples, the amount of data to be analyzed is large. The goal is to find patterns in the data. This whole area of research is called *machine learning* [6] and the methods used and the problems they solve can be separated into

- *unsupervised* ("class discovery") and
- *supervised* ("class prediction").

When dealing with classification of tumors, for instance, there are three main types of statistical problems (Dudoit 2000):

1. the identification of new/unknown tumor classes using gene expression profiles – cluster analysis/unsupervised learning,
2. the classification of malignancies into known classes – discriminant analysis/supervised learning,
3. the identification of "marker" genes that characterize the different tumor classes – variable selection.

So far, most published studies done with microarrays deal with tumors. The last section in this chapter gives examples of studies in cancer research.

3.2.1 Unsupervised Learning: Clustering

To find structure in the gathered data many techniques are being used, for instance *cluster analysis*. Clustering is a kind of classification (grouping), in which there are no category labels that tag objects with prior identifiers. The absence of category labels distinguishes cluster analysis from discriminant analysis (and pattern recognition and decision analysis). Clustering is therefore an unsupervised learning technique. So it is used when one knows very little about the data, as opposed to when there is a set of positive and negative examples that are employed to produce a classifier that can be used to classify examples never before seen.

To be able to group the objects one has to have be some kind of metric by which it is possible to decide which objects are closer to each other. A

[3] An example of how genes interact in detail is presented in *A Genetic Switch* by M. Ptashne.

[4] Some good arguments (by Kauffman, 1993) exist for low average connectivity (k). In simulations, values of $k > 2$ generate fitness landscapes where it seems difficult for evolution to reach global maxima. This may be helped, however, by using "canalyzing" boolean functions, that is functions where one variable determines one or more output values (True or False, 1 or 0). For instance, the *OR* function just needs one input variable to be *TRUE* to get *TRUE* as output, the other variables do not matter. The *AND* function is also a canalyzing function as just one input needs to be *FALSE* to get *FALSE* as output, regardless of what the other input variables are. A function that seems to be used in known networks is the *IF NOT* function, where an inhibitor just needs to be present (*TRUE*) to get *FALSE* as output. Some new research on complex networks, see Jeong et al., *Nature*, 5 Oct. 2000, have added insights on error tolerance and stability.

[5] The methods mentioned here all rely on many samples. But often research is being done with only a few arrays, however. An example: experiment at Sahlgrenska Endocrinology lab with two groups of women, with and without diabetes, all of them overweight, and biopsis were taken at baseline (day 0), 8 weeks and 18 weeks, with two chips at every time interval (labeled 0A, 0B, 8A, 8B, 18A, 18B). A total of 2x3x2=12 arrays. A previous run (with males) was done with only a single array at every time interval. This small number of samples makes clustering techniques almost useless, but if similar experiments are done later on, the combined data sets may be analyzed.

[6] See *Machine learning* by T. Mitchell for an overview.

proximity matrix is calculated where all pair-wise distances are included. There are, however, different ways of measuring distance, the most common being euclidean distance (between points i and j in multi-dimensional space),

$$d = \left((x_i - x_j)^2 + (y_i - y_j)^2 + \ldots\right)^{1/2} \qquad (3.1)$$

and Hamming distance (number of differing bits or features) for binary metric (else it is called Manhattan distance).

Clustering is an often used technique in gene expression analysis and there exists many different kinds (see Jain and Dubes), of which hierarchical clustering was among the first to be used in gene expression analysis. Since the data is in a matrix form, clustering may be applied to both rows and columns, which is called *two-sided* clustering. In Table 3.1 on page 12 different clustering techniques are listed, and studies with comparisons of different methods are listed in Table 3.3 on page 13. Other unsupervised methods of course exist (see Table 3.1).

Something to bear in mind when using clustering methods is that they *always* produce a group of clusters, so the mere existence of clusters do not by itself contain information.

Jain and Dubes conclude that different problems have different best clustering algorithms, and there is no single best way of measuring how good a clustering is. Yeung et al. makes an effort on creating a measure for this ("FOM").

3.2.2 Supervised Learning: Classification

Supervised techniques are used when there is a set of examples in which each have a label with the class it belongs to. If we have enough examples we can construct a classifier that can decide to which class an unknown sample belongs to. It is the number of examples available for learning that is the biggest problem in research that uses microarrays. Also, the quality of the samples is crucial, as incorrectly classified samples used in training produce a classifier that classifies something different than what is intended [7].

The aim of the classification (class prediction) is to identify a minimal set of features which provides a correct and robust classification of the data. The generalizing ability of the classifier is of great importance, as the new samples may be somewhat different than those in the training set. If the classifier is good at classifying the examples in the training set but not others, it is *overfitted*. To avoid this, separate test sets should be used, or parts of the training set be withheld for this purpose ("leave-one-out", cross-validation).

Examples of some supervised machine learning methods are: k-nearest neighbour, decision trees, inductive logic programming, artificial neural networks, support vector machines, Bayesian belief networks, genetic programming. (See Table 3.2 and 3.3.)

[7] Alon et al. have an example of this.

Table 3.1: *Some* unsupervised *methods used in microarray data analysis.*

Method	**Used by**	**Data set**
Hierarchical clustering	Eisen et al. (1998)[a] Perou et al. (2000)[b] Alizadeh et al. (2000)[c]	[a] Yeast (79 x 2,467) and human (12 x 8,600) time-series. [b] 65 x 8,102 genes, human breast tumours. [c] 96 x 4026, lymphoma.
k-means	Tavazoie et al. (1999) Mjolsness et al. (1999)	Yeast
Self Organizing Maps (SOM)	Tamayo et al. (1999)[a] Golub et al. (1999)[b] (+ neighbourhood analysis)	[a] 17 x 6,000 human genes, hemapoietic differentiation, time-series. [b] 38 x 6,817 human, leukemia.
Principal Component Analysis (PCA)	Raychaudhuri (1998)	Yeast sporulation time-series, 7 x 6,200 (Chu 1998).
Singular Value Decomposition[a]*, Gene Shaving ("iterative SVD")*†[b]	Alter et al. (2000) ‡[a] Hastie et al. (2000)[b]	[a] Spellman et al. (1998) yeast time-series, Mol. Biol. Cell **9**. [b] Lymphoma (Alizadeh 2000).
Cluster Affinity Search (CAST)	Ben-Dor and Yakhini (1999), Bittner (2000)[a]	[a] 38 x 8,067 human, cutaneous melanoma
*Plaid models**	Lazzeroni and Owen (2000)	Yeast (Eisen 1998).
Shannon entropy, Mutual information	Fuhrman et al. (2000)[a] Butte, Kohane (2000)[b]	[a] Rat CNS development: 9 x 112 genes (Wen 1998). [b] Yeast (Eisen 1998).

†May also be supervised.

‡"Eigen-genes" and "eigen-arrays".

* "Overlapping two-sided clustering."

Table 3.2: *Examples of* supervised *methods in microarray data analysis.*

Algorithm	**Used by**	**Data set**
Support Vector Machine	Brown et al. (2000)	Yeast 79 x 2,467 (Eisen 1998).
	Cai et al. (2000)	Lymphoma 96 x 4026 (Alizadeh 2000).
Decision Trees (C4.5)	Brown et al. (2000)	Yeast 79 x 2,467
	Cai et al. (2000)	Lymphoma 96 x 4026
Linear Discriminant	Brown et al. (2000)	Yeast 79 x 2,467
Supervised harvesting of expression trees†	Hastie et al. (2000)	Lymphoma 96 x 4026; Cancer 61 x 6,830 (Ross 1999).

†Starts with hierarchical clustering.

* Implementation by Quinlan.

Table 3.3: *Papers* comparing different methods *in microarray data analysis.*

Author	**Methods**	**Comment**
Brown et al. (1999)	SVM, Parzen windows, Decision trees, Fisher's linear discriminant.	Radial basis SVM best at classifying genes trained on gene expression data.
Cai et al. (2000)	SVM, Decision trees.	SVM superior.
Tibshirani et al. (1999)	Two-way hierarch., two-way k-means, two-way TSVQ†, block clustering.	Gene shaving introduced.
Dudoit et al. (June, 2000)	Fischer Linear, Maximum likelihood, Nearest neighbour, Classification Trees, Aggregating classifiers (bagging, boosting).	Data: lymphoma, leukemia, Ross cancer.
Yeung et al. (Aug, 2000)	Validating hierarchical clustering, k-means (random init.), CAST.	Introduces "Figure of Merit", an estimate of the predictive power of method
Ben-Dor, Friedman and Yakhini (2001)	Evaluating putative classification.	Also introduces own approach to clustering and scoring.
Shamir and Sharan (2001)	CLICK, SOM, k-means, hierarchical clust..	Their own CLICK alg. based on graph-theory get best results.

†Tree-Structured Vector Quantization, uses k-means clustering.

3.3 Applications in Research

3.3.1 Lymphochip, Alizadeh et al.

This paper studies Diffuse large B-cell lymphoma (DLBCL, the most common subtype of non-Hodgkin's lymphoma) which is clinically heterogeneous: 40% of patients respond to therapy, the others do not. Diversity in gene expression among the tumors of DLBCL patients "apparently" reflects the variation in tumor proliferation etc., according to the authors, and they identify two molecularly distinct forms of DLBCL. Patients with one of these had a significantly better overall survival than those with the other kind. The molecular classification of tumors based on gene expression can thus identify previously undetected and clinically significant subtypes of cancer.

The classification does not appear, however, when all genes are included (Ben-Dor et al. 2001). The authors hand-pick a small subset and cluster tissues with respect to this data set.

Microarrays: A specialized cDNA microarray, the 'Lymphochip', was prepared from selected genes that are preferentially expressed in lymphoid cells and genes with known or suspected roles in processes important in immunology or cancer. About a quarter of the genes were represented by two or more different cDNA clones (the array has a total 17,856 cDNA clones), providing internal controls for the reproducibility of gene expression quantitation. A total of 96 samples (72 cancer and 24 non-cancer samples) were used.

Data analysis: Fluorescence ratios for each array were scaled so that the median ratio of well measured spots was 1.0. The ratios that were more than 1.4 times the local background intensity in each channel were considered well measured. The ratios were log-transformed. For samples that were measured on several arrays, the values for the genes were averaged. Genes that were not measured well on at least 80% of the 96 mRNA samples were excluded. Data for the remaining genes were centered by subtracting (in log space) the median observed value, to remove any effect of the amount of RNA in the reference pool. This data set contained 4,026 rows (genes).

Hierarchical clustering was applied to both genes and samples using the weighted pair-group method with centroid average (as implemented in the program Cluster by Eisen, available from Stanford Genomic Resources). The result was analyzed with TreeView (Eisen).

3.3.2 Alizadeh Follow-up: Cai et al.

The Alizadeh data set was used by Cai et al. to examine two supervised machine learning algorithms, Support Vector Machines and C4.5 (an implementation of decision tree learning, that works with information gain in using a feature for classification, based on Shannon entropy calculations)[8] for their abilities to classify cancerous or non-cancerous tissue samples. The results show that SVM outperformed C4.5. In the experiments, the results obtained from leave-one-out and 10-fold cross-validation methods were consistent. With both SVM and C4.5, after excluding 218 genes that are known to be important for lymphoma, Cai et al. obtained the same performance in classification or the same decision tree compared to the ones where all genes were used. This result strongly suggests that other genes may play crucial role in distinguishing cancer from non-cancer tissues.

The difference between SVM and C4.5 however, is that the latter is no "black box" as separating conditions are explicity stated in the output.

SVMs have proven to be good in other related areas as well, Brown et al. used it in "Knowledge-based analysis of microarray gene expression data by using support vector machines" and it is being used at Stockholm Bioinformatics center by A. Elofsson in his work on protein fold prediction.

3.3.3 Other Examples

In the chapter on Data sets used in this study, summaries of some other papers are presented. (Most papers mentioned in this study are publicly available via WWW, please see Bibliography for URLs.)

[8] See *Neural Networks* by Haykin for information about Support Vector Machines, and *Machine Learning* by T. Mitchell for information about Decision Trees. C4.5 and its successor C5.0 are implementations by Quinlan (http://www.rulequest.com/).

Chapter 4

A New Method for Extracting Relevant Genes and Gene Interaction

This study started with the formulation of a problem that large-scale gene expression analysis faces: How to extract relevant gene interactions among the vast amount of possible interactions. As mentioned earlier, the number of possible functions with N genes with binary values is 2^{2^N}, which is greater than 10^{19} already for $N = 6$. If 10^{10} functions could be evaluated every second, an exhaustive search would still take 30 years to complete.

A much more clever search algorithm must be employed, and before deciding on an approach a survey was done on the research papers published before October 2000. Biologically inspired methods were considered interesting as these are suited for exploring large search spaces. None of the 50 papers employed genetic algorithms (for these objectives[1]).

Genetic algorithms start with a population of hypotheses (individuals, often called *chromosomes*), that are evaluated and mutated repeatedly until a solution has been found (see Appendix for details).

4.1 Design of the Method

The method is based on classifier rules made of conditions based on the average expression value of some genes. This average is compared to a threshold. If the conditions are met the sample is classified as "abnormal". The gene expression values are trinerized (but floating point values would naturally also work).

[1] In reverse engineering genetic algorithms and genetic programming have been used, see Wahde and Hertz (1999), Lanza et al. (2000) and Koza et al. (2000).

4.1.1 Encoding of the Chromosome

The rules can be very flexible. Conditions are made up of a different number of genes and have possibly different operators. The thresholds may be different in the conditions and the conditions may be linked differently (AND, OR, XOR, IF NOT ...).

Definition of a rule:

$$COND_1 \bowtie COND_2 \bowtie \ldots \bowtie COND_N$$
$$\text{where } COND_i \text{ is } \quad g_1 \ldots g_k \diamond a$$
$$\text{where } \diamond \in \{>, <, =, \neq\} \text{ and } -1.0 < a < +1.0$$

The conditions are linked by $\bowtie \in \{$AND, OR, XOR, IF NOT ...$\}$, the average value of genes $g_1, \ldots, g_k$ in a sample is compared to a according to the operator $\diamond$, and $k, N \leq some - max - number$.

Maximum number of genes and conditions: Some maximum values must be used in order to avoid that the complexity of the rules grow without limits. Arbitrarily a maximum of 10 genes in each of a maximum of 10 conditions were chosen ($k, N \leq 10$), which should be more than enough. In the runs made in this study this maximum was never reached. (Also, the purpose of this study is to, hopefully, find simple rules for prediction. A big number would make the rules resemble the clustering made so far in the research.)

Limiting the possibilities ($\bowtie$, $\diamond$): Here, just an AND linkage between the different conditions is used. As floating point average levels are used, the $=$ and $\neq$ do not have any practical meaning, so just $>$ and $<$ are used as operators.

The simple AND linkage between conditions makes hierarchical conditions unnecessary[2], also

[2] Example: If parentheses are moved in (a OR b) AND (c OR (d AND e)), we get a completely different function.

the different functions would have needed different evaluation functions, which is excluded in this version.

Computer code for chromosome: The chromosome may be coded like this in Pascal:

```
classification_chromosome = record
  // No. of conditions:
  ngroups : integer;
  // No. of genes in each:
  number_of_members : array[1..NGmax] of integer;
  // Genes in conditions:
  members : array[1..NGmax, 1..NMmax] of integer;
  // Operators in conditions:
  operator : array[1..NGmax] of char;
  // Levels in conditions:
  level : array[1..NGmax] of single;
  // Conditions linked by:
  comparison : array[1..NGmax] of char;
end;
```

Example of a rule: A rule may be `23 651 > 0.872 AND 4 < 0.3`, which means that the genes 23 and 651 must both be +1 (since we work with trinerized expression values) as the average otherwise cannot be >0.5, and gene 4 must be either 0 or –1.

4.1.2 Preprocessing of Input Data and Relevance Measures

This GA system has the possibility to incorporate information to guide the picking of genes that are used in the rules, the implemented way is based on a relevance measure.

When the data matrix is imported to the system the number of lines (genes), columns (samples) and identification symbol for abnormal samples are stated. The system then performs an relevance calculation according to this formula[3] (which gives genes with a *consistent misregulation* high value): $R(g) = D(g) * C(g)$, where D is a distance measure

$$D = |N(1,A)/N(A) - N(1,B)/N(B)| + \\ +|N(0,A)/N(A) - N(0,B)/N(B)| + \\ +|N(-1,A)/N(A) - N(-1,B)/N(B)|$$

where N(1,A) is the number of 1s in the normals, N(A) is the number of normal samples etc. C is defined as $C = \max(N(-1,B), N(1,B))$.

This relevance measure is only used for (a) Comparison with how much a gene is used, and (b) To select genes proportionally to its "relevance", and choosing another measure would probably not change the results that much. (See for instance the results on choosing genes with equal probability or proportionally in the Cutaneous Melanoma data set results.) For a specific data set, one could of course develop a special relevance measure built on some kind of capability for a gene to define clusters in the data etc. But biasing the relevance measure, and thus the selection, towards a specific capability also limits the search, making some seemingly uninteresting, but possibly valuable genes, much less probable in the selection. Whatever relevance measure that is used for proportional selection of genes in the evolution of rules, one should also make a reference line of runs with equal probability selection.

[3]The relevance measure was formulated by M. Wahde.

4.1.3 Population

Default population size is 100, and the *tournament size* is (as usual in GAs) 2, but a possibility to change this is put into the graphical user interface (although it is not implemented). Tournament size is the number of individuals that are randomly picked and compared, the winner replacing the less fit individual with its offspring.

The initialization of the population is made with individuals of 1 gene and 1 or 2 conditions.

Selection of two individuals before comparing them is random. The least fit individual is *always* replaced with offspring from the fittest. (Another way would be to sometimes let the least fittest produce offspring.)

4.1.4 Mutations

An extended form of mutation has been used instead of having crossover (see Appendix: GA Decisions). Mutation types are: mutation of the number of genes in a condition, the identities of the genes, the level in the condition, the number of conditions and how the conditions are linked, but in this version the last possibility is not implemented as there is just one kind of linkage.

There are mutation rates that can be set for all these (0.05 is used as default) and a new individual may thus have (a) many changed genes in a condition as every gene may be changed, (b) a new number of genes in that same condition, (c) a new level to be compared with, (d) more or less conditions. (A new condition only has one gene in it.)

The condition for mutation is:
If random number $r < P_{mutation}$ *then* mutate.

```
Mutate(ind)
begin
  r:=random;
  if (r < ngroups_mutation_prob) then
    {change number of groups in ind +1 or --1 with equal prob}

  for j=1 to ind.ngroups do
    r:=random
    if (r < nmem_mutation_prob) then
      {mutate the number of group members (add or subtract one)}
```

```
    for k=1 to number_of_members_in_group do
      r:=random
      if (r < member_mutation_prob) then
        {choose a new gene and replace the old one}
      r:=random
      if (r < level_mutation_prob) then
       {new random level}
end mutate
```

4.1.5 Fitness

The primary fitness (f) is the number of correctly classified samples, but this is decreased by multiplying it with a punishment factor p for the length of the individual giving an adjusted fitness $f* = f \cdot p^n$, where n is the number of genes in the chromosome. (This was however changed to $f* = (f-1) + p^{n-1}$ after testing, so that a shorter individual never can replace a more correct individual.)

4.1.6 Termination Criteria

A run is continued for a number of evaluations (which is specified by the user), but may be terminated earlier if a solution has been found. Two ways of stopping were tried: (1) terminate as soon as a 100% correct solution has been found, (2) keep on evaluating a specified number even after a 100% correct solution has been found. The first was used on the Alon data set, the second on Bittner et al. and the artificial Alon data set. In the program these extra individuals are called "razor".

The fitness of the best individual is monitored, but for a development of the software that may be extensively used for batch processing, the monitoring of other measures should be helpful (see Appendix for an overview). A user specified maximum number of evaluated individuals is now included, as a means to cut off unsuccessful runs and restart the run with a new initial population.

4.1.7 Validation of Rules

To validate the rule some samples are needed that are not used in training: (1) Specification of a separate validation set, supplied by the user, (2) Cross-validation (random choice of training set and validation set from the set of examples), (3) Leave-one-out.

Most data sets are small and therefore cross-validation or leave-one-out are good choices since they do not require a separate set of examples.

A test set is not available yet, but is needed if the prediction formula is to be evaluated.

Leave-one-out is an option for a run (not used on all of the data). Cross-validation is not implemented in this version.

4.1.8 Expression Values

There are two kinds of microarray data: ratios from cDNA microarrays and absolute values from Affymetrix oligonucleotide arrays. To express whether a gene is up- or downregulated (or not changed) in some samples (the "abnormal" ones), the average expression of a gene in the normal samples (using all the normal samples) is calculated and used as a reference, calculating ratios by dividing all that gene's values with its normal average. This enables a gene to have different values (all positive, a value of >1 showing a greater than average value, <1 less than average) in the normal samples as well as in the abnormal samples.

A threshold is then applied for what is to be considered as "change" to give us a trinerized set (0, +1, –1) of values for all of the samples.[4]

So long as the set of samples is divided into normal and abnormal, there shouldn't be any difference between "absolute" values or ratios in the original data when calculating the new values as described above.

4.1.9 Output

In the *graphical user interface* the evolution may be monitored: the number of evaluated individuals, number of samples to classify, the best individual, number of correct classified samples and fitness for the best individual, and if a run is terminated or unsuccessful (as well as the run number of the total) are all displayed.

A *logfile* is also printed with the run ID, the left out sample (0 if all samples are used in run), the last best individual (even though many may be equally good), the number of correctly classified samples and the "time" (measured as number of evaluated individuals), and last in the logfile the sums of all the different genes, how many runs that gave a 100% correct individual and how many runs that were unsuccessful (and thus repeated).

[4] When this project started the idea was to use the raw data from microarray experiments with just a variable cutoff value for the ratios (fold changes) to create genes values of not changed genes (0), downregulated (–1) and upregulated (+1). It became clear however that this was a too simplified view, and a lot of papers published on the subject were studied. Because of the different papers not being quite in agreement with each other, a very simple version was chosen that could later be changed and new runs made with new data sets calculated. Some new ideas were input by Zoltan Szallasi who provided us with the data sets (Alon and Bittner). In this work, however, all data sets result from this first preprocessing and then applying a threshold ratio value for change.

An extension could be to somehow sum the rules and group them for a easier analysis, but this has not been done in this version.

4.2 Properties of the Method

Before discussing the details of how the method works, let us repeat how the algorithm is supposed to learn: Supervised learning relies on a training set of normal and abnormal samples to evolve a predictor for the classification of unknown samples as being normal or abnormal. The aim is not to discover classes but to predict them.

4.2.1 How to Avoid a Black Box

Artificial Neural Networks (ANN)[5] have been proved to give good results in areas where other techniques have problems, but there is no easy way of calculating which topology ("wiring" of the network) that should be used and – if the network works fine – how to interpret the network connections, weights and thresholds to get an equivalent program that can be analyzed, which is preferable. The classifier thus becomes a "black box" that hides the knowledge.

Other techniques, like Genetic Programming with machine code[6] or Support Vector Machines, also have the black box problem.

Genetic Algorithms (GA)[7] was chosen to evolve rules that classify sets of samples of (discrete) gene expression levels, but avoiding the "black box" character by simply limiting the way the genes can influence the outcome. The result is a classifier made of a set of conditions in which the average value of the included genes is compared to a threshold level. The conditions are grouped with a boolean AND (or another boolean function).[8].

When working with genetic algorithms (and as always in machine learning), the most central factor for success is the *encoding* of the candidate solutions – Can the real solution even be expressed using the chosen encoding?

The method does not limit itself to classifying examples that are linearly separable (XOR is an example of a function that is not linearly separable, it evaluates to TRUE only if exactly one of the variables is TRUE). For instance, a classifier that contains two genes that may not be +1 at the same time may be constructed as (g_1 $g_2 > 0.0$) AND (g_1 $g_2 < 1.0$) – the genes' values are averaged before comparing with the level, so if both are +1, then $g_1 = g_2 = 1$.

Other functions: IF NOT (with a = threshold value for inhibitor) may be represented like $g_1 < a$ AND g_2 $g_3 > b$; OR functions may be encoded, for instance $g_1 < 0.5$ OR $g_2 > 0.5$, but if the possibility to have OR between the conditions is left out and just AND is used, these OR functions can get more complicated. With binary values OR is simply g_1 $g_2 > 0$, with trinary values it is g_1 $g_2 > -0.5$ but that also makes $g_1 = g_2 = 0$ TRUE. $g_1 \neq 0$ AND $g_1 \neq 0$ have to be added.

Can a chromosome with AND and OR between conditions represent all possible functions (not just boolean)? No, not all possible ones since then we would have to have continuous weighting of all genes (0.876*g_1 + 0.543*g_2 ...). But a gene may be given more weight by using it several times (g_1 g_1 g_2) and at least get a "less discrete" function.

Can a chromosome with just AND between the conditions (averages with thresholds) and trinerized values represent all *boolean* functions? No not all (there are problems with some OR).

As data sets of this kind always contain some rules, the exclusion of some functions may be accepted as this limitation is just one more in the aim to limit the number of output rules. Also, a more complete set of functions may be tried later on.

4.2.2 GA Theoretical Arguments

Although there is no rigorous answer to the question when (or if) to use genetic algorithms on a problem, many researchers, according to Melanie Mitchell, share the intuitions that *if*

- the space to be searched is large and
- is known *not* to be perfectly smooth and unimodal (i.e. consist of a single smooth "hill"), or

[5] ANNs are described in detail in *Neural Networks – a comprehensive foundation* by Simon Haykin, 1999.

[6] Genetic programming come in many flavors, for an overview see *Genetic Programming an introduction* by Banzhaf, Nordin, Keller and Francone, 1999.

[7] See Appendix and *An Introduction to Genetic Algorithms* by M. Mitchell, 1996. For an overview of different machine learning approaches, see *Machine Learning* by T. Mitchell, 1997.

[8] For those who want to compare our rules to an ANN, they can be formulated as an ANN with a hidden layer. The first layer of genes are connected via weights of $1/N$ to a node (N is the number of genes connected to that node), and the weight is 0 for the genes that are not included in the condition. Each node in the second layer is equal to a condition in the classifyer. The third (output) layer take the "1" or "0" from the nodes in the second layer and have a threshold of $N - 0.5$ (all weights are 1), which equals that all inputs must be "1", if the conditions are grouped with a boolean "AND", and 0.5 if "OR".

- is not well understood, or if
- the fitness function is noisy, and if
- the task does not require a global optimum to be found (quickly finding a sufficiently good solution is enough),

then a GA will have a good chance of surpassing other "weak" methods (methods that do not use domain-specific knowledge in their search procedure).

This fits our problem nicely, but still several crucial decisions have to be made.

4.2.3 Running

The way the population moves towards a solution (evolution) is also very important and is the consequence of many separate decisions taken together. A usual problem is that the GA converges too quickly and gets stuck at a sub-optimal solution. There are several ways to avoid this. In this implementation a *maximum number of evaluated individuals* makes the GA restart if no solution has been found. By making many runs (each new run getting a completely different initial population) the problem with a sub-optimal solution in a single run is small. Thousands of runs have been made with every data set.

4.2.4 Complexity of Rules

When calculating the fitness measure longer rules were penalized, but no distinction was made between rules consisting of fewer but longer conditions compared to shorter but more conditions, as it is difficult to assess the importance of such a distinction. It can be argued that simple conditions are more desirable than a small number of conditions, as the number of possible values of genes in a condition increases rapidly with the number of genes. A possibility would be to have different kinds of penalties and do parallel runs (maybe with the same initial population).

Initially, a penalty that from a user specified value lowered the fitness according to the length of the rule was used. This "punishment factor" may be set high to give shorter but less correct rules an advantage. This was tried ($f* = f \cdot p^n$, where p is the punishment factor specified and n is the number of genes in the rule)[9], but it was changed in order not to let shorter rules outperform longer but more correct rules. The fitness adjustment was changed to $f* = (f - 1) + p^{n-1}$.

This change always gives the more correct rules a higher fitness, since the penalty may be 0 but always less than 1. The penalty thus only distinguishes between equally correct rules but with different number of genes in them.

Tests were performed with the Alon and Bittner data sets without and then with a "razor" that continues evolving new rules even though a 100% correct solution has been found. The razor was able to cut the length of the rules in the Bittner set by 1–2 genes, but not in the Alon set. As the Alon set is considered to have more noise, this was not surprising.

Reducing the complexity of the rules is important, since the aim is to avoid long rules which would be similar to other predictors that employ hundreds of genes as input.

4.2.5 How to Use the Rules

Even though the method is formulated as searching for predictor rules, the discovered rules can be used in several ways:

1. Each rule is used as a classifier. (Development of new test methods for diseases.)
2. Combination of several rules can be used to create different sets of rules where a gene may be substituted for by others ($g_1 g_X > a$, where X can be 2 or 3 or ...).
3. The number of times a gene is used in rules is a measure of how important it is.

When a rule is found (or rather, many similar rules) a biological analysis should be done if possible to determine whether the result is due to chance or represents a possible biological relationship.

Since there are so many possible interactions, the pool of rules is a much smaller search space, so limiting the biological analysis in this way makes it much easier for the researchers.

4.2.6 Number of Rules Found

It must be stressed that the real data sets we are working with have a very small number of columns (samples) compared to the number of rows (genes). Therefore it is not hard to find rules that separate the samples. With more data (samples), it gets harder.

[9] The software had "Max fitness decrement" as an option to maximize this advantage, but it was disabled when the fitness adjustment was changed.

4.3 Implementation

For the system we have used Delphi which is an object-oriented Pascal development environment for Microsoft Windows. The system is called *GANNclassifier*.

Running the system consists of five parts:

1. Preparing of the input file.
2. Starting GANNClassifier and loading the prepared input file.
3. Selecting parameter values.
4. Run the GA.
5. Evaluate (the logfile).

This can be done iteratively as done in two cases, discarding the unused genes in the next iteration. The system is explained in detail below:

4.3.1 Data Input and Output

The system for the input to the genetic algorithm and writing output had small additional subroutines for conversion of files, but the raw data was supplied as Microsoft Excel spreadsheet files and the initial preprocessing (converting expression values to discretized data, limiting the values to –1, 0 and +1) was done with simple macros in MS Excel.

Later, the frequencies of used genes were extracted from the output files (text files) and imported to Excel for inspection of results. Spotfire Pro was used a few times for plotting results.

4.3.2 User Interface

The user prepares a text file for input in the following format (space separated files):

```
<data set name>
<no. of lines> <no. of columns>
<normal samples classification symbol>
<>   <>    <class> <class> <class> <class> ...
<ID1><ID2> <val>   <val>   <val>   <val>   ...
...
```

The file is loaded after starting GANNClassifier.

4.3.3 The Main GANNClassifier Window

If GANNClassifier is started it displays the main window (Fig. 4.1) where the user load files and chooses parameter settings for the runs. This window is not closed during the run (which can be monitored using the Run window, Fig. 4.2).

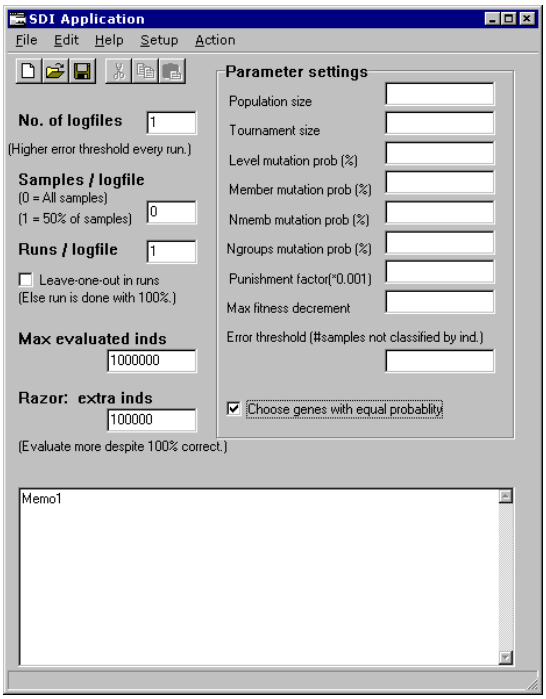


Figure 4.1: *This is the main window for the application that is used for selecting input file, parameters and starting a run.*

Parameters

The system has several parameters that can be easily manipulated before each run:

- Run and Logfile parameters
  - No. of logfiles: For step up change of error threshold
  - How many samples to use in run (all samples or 50% randomly selected)
  - Runs per logfile: Since runs are non-deterministic, many runs must be made
  - Leave-one-out: Randomly exclude one sample in a run
  - Max evaluated individuals (run is terminated at this value even though correct rule is not found)
  - Razor: continue to run for this many individuals after finding 100% correct rule
- Parameters:
  - Population size (default: 100)
  - Tournament size (not used, constant value: 2)
  - Level mutation probability (default: 5%)

- Member mutation probability (default: 5%)
- No. of members in each condition mutation probability ["Nmemb"] (default: 5%)
- No. of conditions mutation probability ["Ngroups"] (default: 5%)
- Punishment factor (default: 0.990)
- Max fitness decrement (not used with the current implementation of fitness calculation)
- Error threshold (max no. of incorrectly classified samples in successful run)
- Choose genes with equal probability: In evolution, if not checked a relevance-proportionate probability will be used.

These settings may be saved in a text file (named `<file>.prefs`) by choosing "File; Save as". Default values are loaded when input file is specified. In the box at the bottom the name of the set and some values are presented.

4.3.4 The Run Window

When a run or set of runs is started the application displays the Run window (see Figure 4.2 on page 23). It first displays the relevance values for all genes (even if they are not used in selection of genes in the evolution). The initial population's use of operators etc. is also displayed in the right box, but this is mainly for development reasons. Then the run starts and new values are displayed as the run progresses. When all runs are made the user gets a message in the rule monitoring box, where it is also displayed if a run was unsuccessful and the number of unsuccessful runs so far.

4.3.5 Logfiles

Logfiles are written on this format:

```
START -- New error_threshold = 0
Run: 1    Left out sample: 0
          Ind: 1963  > -0.3585 &&  1993  1995  < -0.2014 &&
               1926  > -0.370
          [correct=62] Evals: 1010141
Run: 2    Left out sample: 0
          Ind: 913  > -0.9449 && 1464  1423  > -0.7834 &&
               1926  > -0.0233 && 1635  < 0.4729 &&
               813  756  < 0.2809
          [correct=62] Evals: 1156363
Run: 3    Left out sample: 0
          Ind: 908 36 1397 913 507 94 1522 1212  > -0.0933
               && 785  > -0.9503
          [correct=62] Evals: 1018611
...
Run: 1000 Left out sample: 0
          Ind: 1926  > -0.8356 && 1870  > -0.187 &&
               1211  > -0.2746 && 567  > -0.9833 &&
               756  813  1442  < 0.1581
          [correct=62] Evals: 1013219
Used genes:
RelID   ID1     RELval  RELnorm #used   %of used  USEDnorm
1       243     16.36   1       25      0         0.06
2       1414    16.32   0.99    260     3         0.65
3       1992    14.54   0.88    42      0         0.1
...
```

RelID is the ranking in the relevance list, ID1 is taken from the input file, RELval is the relevance value, RELnorm is that value normalized, #used is how many times the gene is used in the runs, all appearances counted, %used is simply the percentage of all appearances and the USEDnorm is the normalized value of #used (so that the most used gene gets 1.0).

4.3.6 Testing

With many parameters and different implementations of some functions, testing could take too long a time. This has been a problem, and therefore (after testing) the same parameters are used, but the way genes are picked for inclusion in an individual is changed to make dual rounds in all experiments.

The mutation rates were changed during testing, but no significant change in the results could be seen.

A program was made to generate data sets (Mathematica was also used), and we started with 10 simple AND-functions made out from 2–5 genes being upregulated (+1) with high probability (equal probability of becoming the other two possible values otherwise) in a 100x100 size matrix (100 genes, 100 samples: 50 normal, 50 abnormal), the other genes had equal probability for all values (0, +1, –1).

AND test function: These sets were generated in several versions. Genes with a >90% probability were readily included in the rules, but the randomly created sets often contained genes that fit together and created good separators even though they had low upregulation probability or were completely random. Typically a group of "good" genes used often had an added condition with randomly created genes, but this added condition came in many variations with low frequencies, thus showing the low robustness of those exact rules. When adding the results from the different sets generated with the same probabilities, these random genes were naturally outperformed.

The genes that were not randomly created always got the best scores in the used relevance measure, and the values were consistent with the probability that they were up- (or down-)regulated.

XOR test function Next binary data matrices (100x100) were generated with hidden XOR func-

tions, and the GA did not have any problem extracting those rules if the other randomly created genes did not contain some that were +1 in more than 70 of the samples. The point in using XOR is that the ad hoc relevance value is 0 for those genes as they have an equal number of 0 and +1 in the normals and abnormals, making a combination of the genes necessary. A clean XOR in a random environment was no problem to find, but when "hidden" by generating genes with more than 70% upregulation, then the XOR could not be found in all runs – typically half of the runs found it. This result was quite good considering single planted genes of 95% upregulation was used to hide XOR.

Time and Runs: Trials in these tests were made with 5–300 runs, depending on how hard it was to find the rules. Often an error threshold higher than 0 had to be employed because of time constraints. An easy set could produce a correct rule in a second, a hard one could need hours on a ordinary Pentium 500 MHz PC. The real data sets would typically use a maximum of 24 hours for a run of a couple of milion evaluated individuals.

Conclusion: The GA finds the separators, but if noise is present (as always in microarray experiments), a lot of random rules may be found. The use of many runs, adding the results, make real rules outperform those random rules.

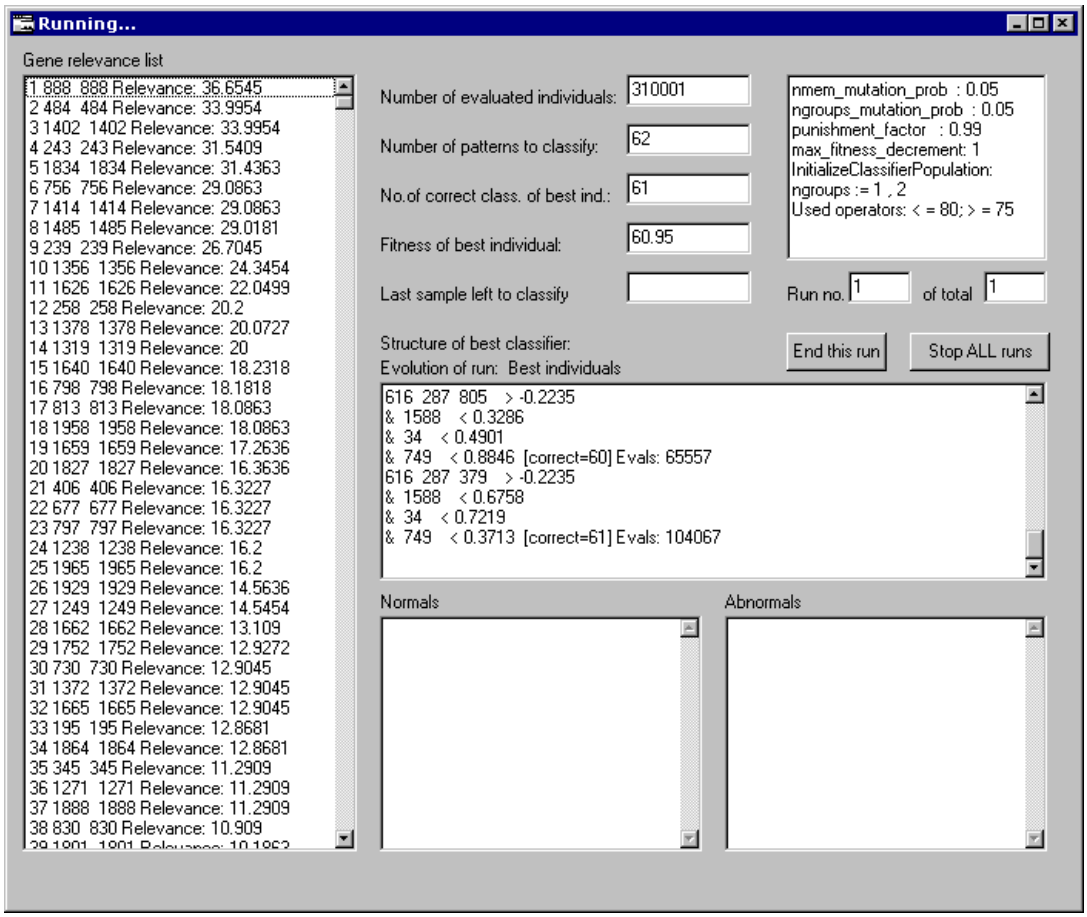


Figure 4.2: *The window for on-line inspection of the runs. At the left the internal relevance list is displayed which may be used for proportional selection of genes (for inclusion in individuals) in the evolution, if not the option "Choose genes with equal probability" was checked in the main window. The number of evaluated individuls is displayed every 10,000 individuals, and for the best individual so far the fitness is displayed. In some cases the last sample to be predicted is interesting. The buttons may jump to the next run or stop all runs of a logfile (this button has to be pressed once for every logfile if several logfiles are being run). For development reasons two boxes, one for normal and one for abnormal samples, are included where the individuals evaluation values for each condition on each sample is presented. The ampersand in the rules is the boolean AND-linkage between the conditions making up the rule:* 616 287 379 > -0.2235 AND ... AND 749 < 0.3713 [correct=61] Evals: 104067. *The number of the current run is listed as well as the total number of runs that shall be completed for this logfile. If the maximum number of individuals is evaluated without the number of correct classifications being above the allowed threshold, the run is simply restarted with a new initial population. (For every new data set the user has to use the main window to load it and manually start a new run. No batch processing is implemented in this version.)*

Chapter 5

Data sets

5.1 Real data sets

This study originally was to use a completely new data set without published results. This had to be changed and we turned to studies already published that had available data sets.

5.1.1 Alon Data set

The data set consisted of 62 samples (22 normal and 40 tumor) with 22 paired samples (we used all samples, not just the paired ones).

The expression values were obtained by using Affymetrix oligonucleotide arrays. The raw data output (intensities obtained with the Affymetrix equipment) was preprocessed in the following way:

1. To compensate for possible variations between arrays, the intensity of each gene (EST) on an array was divided by the mean intensity of all genes (ESTs) on that array and multiplied by 50. [Alon]

2. Then the columns were divided by the average of the columns i.e. by the average intensity of signals on a given array. (This is called "inter-chip wise" normalization.) [Szallasi]

The preprocessing done in this project was based on the gene average normal intensity, that is the average expression value for the gene in all the normal samples. This was then used to calculate the trinerization, where the intensities were divided by the normal average and then trinerized with five different thresholds (2, 2.5, 3, 3.5, 4). This resulted in five data matrices with values –1, 0, +1.

Comparing the gene expression to the average of the 22 normal samples, using a factor 3 as threshold for change, there is one gene that is misregulated in 20 normal samples (out of 22), 1 gene misregulated in 12 normal samples, 1 in 11, 2 in 9, 6 in 7, 8 in 6, 9 in 5, 34 in 4, 95 in 3, 189 misregulated in 2 normal samples, 489 genes misregulated in just 1 normal sample and 1134 genes did not change in any of the tumor samples. Thus the noise level was rather high.

The matrix can also be said to be very sparse, with few +1, –1 values, when the trinerization factor is raised above 2.0.

The probabilities for the values (0, –1, +1) in the Alon matrix when trinerized with a factor 3 (as in the example above) were: Normal samples:
$-1 \Leftrightarrow 0.037$; $+1 \Leftrightarrow 0.0071$; $0 \Leftrightarrow 0.9559$
Abnormal samples:
$-1 \Leftrightarrow 0.038$; $+1 \Leftrightarrow 0.0059$; $0 \Leftrightarrow 0.9561$

5.1.2 Bittner et al. Data set: Cutaneous melanoma

The 38 samples in the original set were from 31 melanomas and 7 controls. The separation was not between the normals (controls) and the melanomas, but between the melanoma samples themselves. A major cluster with 19 of the melanoma samples was regarded, by Bittner et al., as a highly reproducible grouping, and the question is what makes the other 12 samples "abnormal" compared to the 19. The data set used by us contained 8,067 cDNAs representing 6,971 unique genes.

Background (Bittner): The most common human cancers are malignant neoplasms of the skin, and cutaneous melanoma is getting more and more common. Unfortunately, minimal progress is made in non-surgical treatment of advanced disease. Despite efforts to identify independent predictors of melanoma outcome, no accepted [1] marker defines subsets of melanoma. Accordingly, though melanoma is thought to be present with different 'taxonomic' forms, these are considered part of a continuous spectrum rather than discrete entities. Bittner et al. and others (as DeRisi, Khan, Perou, Golub, Alizadeh) have proposed that a discrete and

[1] Bittner refers to histopathological, molecular or immunohistochemical markers.

previously unrecognizable cancer taxonomy can be identified by viewing the systematic data from gene expression experiments.[2]

The samples were obtained from melanoma biopsies (from patients) and from tumor cell cultures (from laboratory). The microarray contained probes for 8,150 cDNAs, representing 6,971 unique genes.

Clustering: Using a matrix of Pearson correlation coefficients from the complete pair-wise comparison of all experiments, the 31 melanoma samples were displayed as a hierarchical clustering dendogram and as a three-dimensional multidimensional scaling (MDS) plot, in which the distance between samples reflects their approximate degree of correlation. Bittner et al. also employed a non-hierarchical clustering algorithm (CAST, cluster affinity search technique), that identified the same major 19 sample cluster.

Two independent approaches were used by Bittner et al. to test the validity of the cluster prediction. The first examines the power of individual genes to discriminate the major cluster of 19 from the remaining samples by examining the frequency of strong classifier genes compared to the expected frequency of such genes if expression is randomly variable, and to the frequency of strong classifiers in random partitions of the same samples into new groupings of 19 and 12. The non-randomness of the cluster results was considered evident. The second approach was based on evaluating cluster membership after introducing random perturbations to the data set. Hierarchical clustering was then performed and a comparison made between the original and perturbed tree. The results strongly supported the view that the major cluster with 19 of the melanoma samples is a highly reproducible grouping.

A weighted list (of 276 genes) was made with the genes with the most power to define the major melanoma cluster of 19 samples. The analysis ranked genes according to their impact on minimizing cluster volume and maximizing centre-to-centre inter-cluster distance. The variance of change of the genes across all experiments correctly defines the boundary of a given sample cluster. This weighted list was compiled from a list of all genes in the data set, also those with a "0" in the Strongly detected column. For this reason, these genes were also included in our runs.

[2] A problem, according to Bittner, is: "... for melanoma, inherent or technically induced variation could obscure such a classification as its appearance is very similar between patient samples and, in contrast to haematologic cancers, it has few known recurring genetic changes."

Genes 1–3613 (our ID numbering) are "strongly detected" according to the original data set if "spots have an average mean intensity above background of the least intense signal [green and red channels are measured independently in cDNA array experiments] (Cy3 or Cy5) across all experiments >2000 arbitrary units, and an average spot size across all experiments of >30 pixels."

5.1.3 Values in Trinerized Data set

Expression values in the Bittner data set after trinerization with factors 2 and 3 have the following probabilities:

```
Frequencies (as probabilities):
Value:        +1      --1       0
Factor 2
  normals: 0.095   0.071   0.834
  abnorms: 0.082   0.047   0.870
Factor 3
  normals: 0.040   0.031   0.929
  abnorms: 0.029   0.020   0.951
```

A clear distinction between normal and abnormal samples, although this is decreased when the trinerization factor is raised to 3. This higher factor also more than halves the number of non-zeros, clearly showing the impact of the threshold value. (The Bittner data set in our iterations have been trinerized with 2.5 as threshold. But we have also tried with 2.0, 2.5 and 3.0 and compared the results.)

5.2 Artificial Data sets

Apart from the test data sets for the genetic algorithm mentioned earlier an artificial data set based on the Alon data set was also created. As a test of how these noisy cancer data sets can hide rules the frequencies of the different trinerized values (–1, 0, +1) in the Alon data matrix were calculated and an artificial version (1991 genes and 62 samples, just as in the original) were generated and four new genes were added. These new genes (1992–95) had values that together could form a separator with 100% prediction. (The sum of their values always being ≥ 2 for tumors.)

The above mentioned overall probabilities for the values (–1, +1, 0) in the Alon matrix (factor 3 for trinerization) were used.

Other ways of generating artificial sets can be tried, for instance having probabilities for values for each gene, making the set more like the original set. In future work several schemes should be tried.

Chapter 6

Results

Even though the main purpose of the method is to produce reliable predictors, running the GA on the data sets results in *both* a set of rules (predictors) and a frequency of used genes in these predictors. These can be used in several ways:

1. The rules are analyzed by themselves:
   - Which rule best separates the samples?
   - Which rules are most common in the runs?
   - Can rules be but together to form rules like g1 [g2 g3] > 0 (g1 with g2 or g3).
2. The rules may be used as a pool of possible relations between genes.
3. The genes with high frequencies can be investigated further regarding their functions.
4. Both the gene frequencies and the rules that they appear in are analyzed. Which genes are cooperating and how?

This work mainly looked at the frequencies for used genes, and no processing of the discovered rules has been done, e.g. ranking of the rules, putting several rules together or trying to formulate gene interaction hypotheses. Results were compared with the papers published by Alon et al. and Bittner et al. and discussed with researchers familiar with the original studies.

Unfortunately, no biological analysis of the found rules has been done in time for the completion of this thesis, but this analysis may be used in the further work on this method.

Results: In short, the GA found rules in both artficial data sets and the real data sets it was applied to, and the genes found in the Bittner data set were interesting as shown by comparison with the ranked gene list.

The problem of a thorough validation of the results for the real data sets depends unfortunately on getting new data, and this is not available at the moment.

6.1 Does the GA Work? Artificial Data

The runs of the artificial Alon set (with 1991 generated + 4 planted genes making up a separator) were iterated and the genes that were not used in the rules were discarded before starting the next run. The GA did not have any trouble finding the hidden genes, although the exact separator only turned up in some of the runs.

Due to the randomness of the generated matrix it contained genes that could be combined with our inserted genes, so that the separating rules did not solely have the four inserted genes as members. Only one generated data set was tried, but it is obvious that randomly generated matrices with these low probabilities still contain a lot of separating rules. This problem, however, is always present in analysis of microarray data, and must be adressed both theoretically and practically. The problem is clearly not to find separators, but to distinguish the real ones from those randomly created. (See Wahde and Szallasi, 2000, for a discussion.)

6.2 Trinerization Effects

The factor used as threshold when trinerizing the values from continuous ratios to +1, –1, 0, does determine what genes are being used in the rules. This is shown in the Fig. 6.1 where results for the top 25 genes are displayed.

6.3 Iteration Effects

As mentioned earlier, the main problem is not to find rules, but to limit the number of interesting

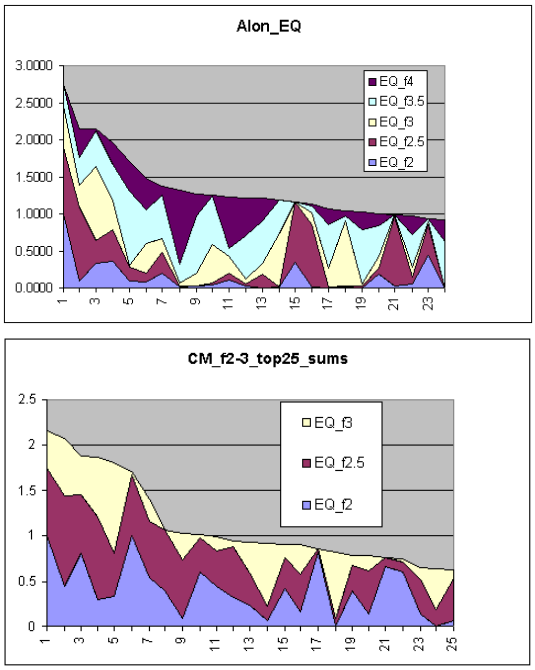


Figure 6.1: *The 25 overall most used genes in different trinerized versions of the Alon data set and "Cutaneous Melanoma". How much a gene is used differs with the trinerization factor. A higher factor makes more genes have a "0" value. When a gene is no longer considered changed (–1, +1), it cannot be used as much in a rule, and therefore other genes are used. Genes that are used a lot at a low trinerization factor are less used when this is increased etc. Parallel rounds are therefore needed. (All values are normalized.)*

genes to be used in predictors and to be explored further. How can the number of genes used in all the rules be limited in several thousands of runs?

A possible route is to use iteration. Such a procedure was used on the Bittner data set (and on the artificial set) with the new input having just the genes used in the last run. Several results have been obtained:

1. Clearer *distinction* of most used genes: An example of how the frequencies of the most used genes were affected by five iterations is shown in Figure 6.2.

2. *Total number* of used genes: Total number of used genes was cut down, making analysis more focused. In Bittner from 8,067 (down to 1161 or 2784 after the first round) down to 125 genes.

Table 6.1: *Iterations with Cutaneous melanoma, EQUAL probability when choosing genes. "Cut" = after runs, to yield new set. Value is number of times a gene was used. "Evals" are how many million individuals that are evaluated before termination (razor inds after +), "Error" is threshold for successful run, "Samples" are how many of the input data set that are classified and "Runs" the number of successful runs.*

Genes	Cut	Evals	Error	Samples	Runs
8067	0	3+2	0	ALL	1000
2784	0	3+2	0	ALL	1000
863	1	3+2	0	ALL	1000
315	1	3+1	0	ALL	1000
214	1	3+1	0	ALL	1000
168	1	3+1	0	ALL	1000
131	1		0	ALL	1000
123		3+1	0	ALL	1000

Table 6.2: *Average number of genes per rule in Cutaneous melanoma runs.*

Iteration	1st	5th	8th
Genes in data set	8067	214	123
Total #used (1000 runs)	4025	2401	2258
Average length	4.0	2.4	2.2

3. The *length of the rules* decreased, making usage and interpretation much easier. In Bittner the average length went down from 4.0 to 2.3 genes per rule. (If the *best* genes were removed in several iterations, the length of the rules increased as would be expected. This test was done at the fifth iteration step.)

6.4 Alon and Bittner Data sets

The Alon data set was quite sparse (not many non-zeros) after trinerization, as was shown in the section of the Alon data set, and this is also true for the Bittner set, even though it did contain more non-zeros.

6.4.1 Alon Data

Alon et al. used a two-way clustering of the genes and tissues (samples) based on deterministic annealing that separates a set of objects into two groups, then separates these into two subgroups

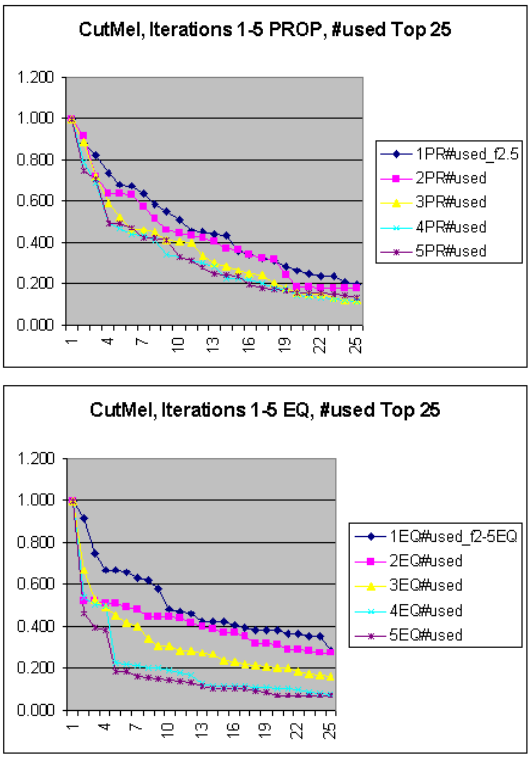


Figure 6.2: *Five iterations with data set "Cutaneous Melanoma". As can be seen the iteration sharpens the difference in frequencies between the most used genes when iterating with a smaller and smaller subset of the original genes. Also, it is clear that the initial frequencies are less different when the genes are selected with equal probability, but that difference is very amplified with iterations, making the curve much more pronounced. (All values are normalized.)*

etc., until all objects are arranged in a binary tree. (2,000 high-intensity genes of the original data set were used for the clustering.) The rows and columns were rearranged so that correlated genes and tissues are displayed together. Gene clusters found by Alon et al.: Ribosomal protein genes and ESTs homologuous to genes that appear to be related to cellular metabolism such as ATP-synthase component cluster together, the former being in agreement with previous observations.

In the paper by Alon the tissue clusters are commented: "the clustering algorithm separated tumor and normal tissues into two distinct clusters [see Figs. 3 and 4 in article], probably primarily because of tissue composition. It is expected that the normal tissue samples include a mixture of tissue types, while the tumor samples are biased to epithelial tissue of carcinoma. For example, among the 20 genes with the most statistically significant difference between tumors and normal tissues (by t test), were five muscle genes."

The average of the expression levels of 17 ESTs in the array that are homologous to smooth muscle genes, was used as a"muscle index". Normal tissues had high index value, while tumors had low muscle index value. The five tumor samples that clustered with the normal samples had the highest muscle index, "perhaps representing tumor samples with a high content of nonepithelial tissues." Similarily, "the three normal tissues in the tumor cluster appear to have relatively low smooth-muscle content."

To test whether the clustering depended only on a few genes (e.g. muscle-specific genes), Alon et al. removed the 1,500 genes that individually best separate tumor and normal tissues (using a 500-gene set). The clustering algorithm still "effectively" separated the tumor from normal tissues. Genes were sorted according to statistical significance (t-test)[1] of the difference in normal and tumors. A 500-gene window was used.

Alon et al. concludes that for many genes there is a subtle, systematic difference between tumor and normal samples. When they inserted cell lines derived from colon carcinoma, the clustering algorithm separated these into a cluster of their own, distinct from but closer to the tumors than the normal tissue.

Comment: An interesting thing to see is that the useage of the overall most used gene 1414 decreases as the trinerization factor increases, even though the relevance value still stays high (p 29). As the relevance value is a simple measure of how consistently a gene is behaving in normal and abnormal samples, it would be expected that it would stay high. But it is not used alone in a condition, so it is the genes that were paired with it that have disappeared at higher trinerization factors.

Many runs were made with the Alon set allowing for a lower number of correct classifications at the maximum time for the run to being considered successful, and this shows the difference with the Bittner set, where few runs stopped at a lower number of correct classifications. This indicates more noise in the Alon set, and this was confirmed in information about the quality of the Alon data not included

[1] t-test?: Although not described in the paper, here it is assumed that Alon et al. calculated the significance level for each gene for the assumption that mean(normals)<>mean(tumors), as can be done with a two sample t-test, also called Student test (sample 1 is for normals, sample 2 tumors). (t-test is a significance test for the mean value of a *normal* distribution.)

Table 6.3: *Our result for the frequencies for the four genes that were included in the Alon "muscle index", all values normalized. As can be seen only one of the genes were heavily used in our predictor rules, but it (1414) was at the very top in the sets with the lowest trinerization factors (2 and 2.5). It is interesting to see that the useage of gene 1414 decreases as the trinerization factor increases (even though the relevance value still stays high), something noticed in other results and something that is hardly surprising, revealing the possibility of making composite predictors that uses several data sets as input. REL stands for our own relevance measure, EQ for equal probability in choosing genes for inclusion in rules and PROP for a relevance-proportionate probability.*

		Factor 2			Factor 3			Factor 4		
Clone	ID	REL	EQ	PR	REL	EQ	PR	REL	EQ	PR
H20709	14	0.04	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
T60155	406	0.44	0.02	0.01	0.49	0.00	0.00	0.74	0.00	0.01
X12369	730	0.35	0.03	0.01	0.09	0.02	0.00	0.03	0.00	0.00
J02854	1414	0.79	1.00	0.86	1.00	0.55	0.45	0.87	0.00	0.03

in the original study.

6.4.2 Bittner: Comments

Several of the genes included in the "weighted list" by Bittner were found by the GA, clearly demonstration the algorithm's ability to find interesting genes, as has been verified by researchers with knowledge about the original study. The runs added genes that were not at all in the list, making further analysis of these interactions interesting. Unfortunately, it has not been possible to include such an analysis in this report.

The iterations did not add more of the genes from the list to the top of the most used genes, but iterations were primarily tried for discarding the uninteresting genes, so this was not considered a failure.

An interesting study by Cai et al. on the Alizadeh data set (Lymphoma) showed that it was possible to remove all genes known to be part of the lymphoma process, and still get an equally good predictor.

This demonstrates the differences in gene expression in a large number of genes in certain states (as a disease etc.). Thus, it is not surprising that the most used genes in the runs on the Bittner set are not genes that are obviously related to the cutaneous melanoma process (but no thorough analysis of this has been made in this study).

6.5 Future Work

In this project, we have used several ideas that by themselves may be both questioned and explored further:

1. "Normalization" the gene expression values by just dividing it with the average of the normal samples. As this is an area in which a lot of research is being done, future work will have to use those techniques that are developed for this kind of analysis. This is clearly outside the scope of looking for new methods of finding separators.

2. The idea of filtering expression value ratios by not only applying a threshold value for what is considered significant (which is part of the normalization etc. that was mentioned above), but also to *trinerize* them. This is not a necessary prerequisite for the method to work, and parallell tests can be done trying out continuous as well as trinerized data.

3. How the gene expression values are used in the rules: calculating the *average expression of several genes.*

4. Hiearchical linking between conditions is not used in the rules at present, the AND-conditions making it unnecessary. With more linking functions the different possibilities of linking them (as in a structured tree) should be an area where different possibilities can be tried.

5. The frequency of a gene used in rules is not really established as a way of judging the importance of that gene, is it just a way of analyzing the runs made or is it by itself something that is worth exploring.

Further potential research topics are:

Possible Boolean functions: The system was tried with an incomplete set of operators and possible boolean functions, with the idea that the same

Table 6.4: *Cutaneous Melanoma with equal probability to choose genes, result after fifth iteration (214 genes out of 8067 left), each iteration being 1000 runs. Rank is the Bittner rank in power to define the nice 19 sample cluster, SD means "strongly detected". In the runs made with a relevance-proportionate selection of genes, the number of genes from the Bittner list was even higher: 8 from Bittner top 25, 8 from Bittner 25-112, 9 outside Bittner list.*

ID	#used	Rank	CloneID	UniGene	UniGene Cluster Title	SD
7935	319	62	288733	Hs.69547	myelin basic protein	0
4577	147	3	266361	Hs.154069	melan-A	0
2243	126		132381	Hs.83790	KIAA0305 gene product	1
6528	122		126681	Hs.190339	ESTs	0
1524	60		810512	Hs.87409	thrombospondin 1	1
3383	59	86	471631	Hs.232068	transcription factor 8 (represses interleukin 2 expression)	1
249	53	2	324901	Hs.152213	wingless-type MMTV integration site family, member 5A	1
2900	50	12	23185	Hs.204133	hexabrachion (tenascin C, cytotactin)	1
3114	48		365647	Hs.17483	ESTs	1
1761	46	94	770935	Hs.9291	ESTs	1
73	45	52	50043	Hs.69547	myelin basic protein	1
1156	43	32	813533	Hs.8180	syndecan binding protein (syntenin)	1
692	37	1	357278	Hs.47343	ESTs	1
1898	35	66	51814	Hs.695	cystatin B (stefin B)	1
3067	34		40026	Hs.2043	adenine nucleotide translocator 1 (skeletal muscle)	1
3275	34	72	110503	Hs.4245	ESTs, Weakly similar to similar to S. cerevisiae hypothetical protein YKL166 [C.elegans]	1
3786	34		502527	Hs.15970	ESTs	0
5223	31		246543	Hs.246234	ESTs, Weakly similar to reverse transcriptase related protein [H.sapiens]	0
141	29		666425	Hs.317	phospholipase C, gamma 1 (formerly subtype 148)	1
1275	24	78	244307	Hs.82085	plasminogen activator inhibitor, type I	1
2246	24		84820	Hs.153954	KIAA0057 gene product	1
50	23	4	234237	Hs.38842	pirin	1
788	23		827156	Hs.82575	small nuclear ribonucleoprotein polypeptide B"	1
927	23		51916	Hs.245990	EST	1
3371	23		417357	Hs.171695	dual specificity phosphatase 1	1

5 from Bittner Top 25, 8 from 25–100, 12 outside 1–100, 5 not strongly detected

runs could be made later with an extended set, making more complex (but still easy to interpret) rules possible. This would be a natural continuation of the project, provided that it may be defined how the results can be used in a biological analysis.

Easier formulation of rules: As the possible averages in the conditions may just have some possible values due to the trinerization of gene expression values, the rules could be processed to yield truth tables for the combined set of conditions.

Separating power of rules: An issue that has not been included in this project is how well the generated rules may separate different sets of samples, creating some kind of extra ranking of generated rules.

Putting together rules: With so many variables (genes), a number that is growing fast as microarrays gets bigger, a way of automatically combining good rules is needed to facilitate the analysis of biological interactions.

How many samples are needed?: It is not a problem, however, to find rules that separate samples (for most publicly available data sets) as the limited data makes a lot of rules possible.

It would be a good idea in future work to try to establish the amount of data needed to yield a certain amount of rules. It is easy to find predictors. However, choosing the right ones is more complicated, and an estimate of how the rules change according to the number of samples would be of great help. This should be done with artificial data sets, maybe using frequencies of values derived from real data sets of high quality.

Extending data: If not enough data is available (to limit the number of rules found, making analysis more tedious), are there ways of extending the number of samples by adding errors in some way?

Guiding the GA: The point of using a genetic algorithm is to be able to search a vast space of possible solutions, but ways of improving this search by limiting the number of genes (by biological analysis) or providing supplementary data should be tried. As shown in the work by Per Jonsson (2001) (at conference Bioinformatics 2001, Skövde), results may be improved substantially by just adding some knowledge to the system, in that case a clustering algorithm.

As new ways of linking information stored in databases are being developed, this kind of search could be guided through information of known interactions and functions to have a kind of automated biological preprocessing analysis that could filter out unlikely rules from the analysis.

Chapter 7

Appendix

7.1 Computation Methods Inspired by Biology

Since nature has been so good at producing complex organisms out of simple atoms and molecules, the human brain being the best example, it is no surprise that nature also has been used in computer science as an inspiration. In the 1950s and 1960s several computer scientists independently studied evolutionary systems with the idea that evolution could be used as an optimization tool for engineering problems, and already in the 1940s the pioneers McCulloch and Pitts united neurophysiology and mathematical logic and showed that a network could compute any computable function. All these biologically inspired methods can be divided into three categories:

- Neurons, brains ⇒ *Artificial Neural Networks*,
- Evolution ⇒ *Evolutionary Algorithms*,
- Thought ⇒ *Fuzzy Logic*.

7.2 Evolutionary Algorithms

The term Evolutionary Algorithms is a general term used to denote several computation methods inspired by natural evolution. There are four major types of such methods, namely Evolutionary Strategies, Evolutionary Programming, Genetic Programming (GP) and Genetic Algorithms (GA). These differ mostly in the coding of the problem and the areas in which they are used. GP and GA are clearly the most used of the four kinds. They all use mutations (and maybe also sexual recombination, where a new individual is created with parts from the ancestors) and evolution of a population. The solution to the problem is found by random changes in the individual programs that are evaluated using a *fitness measure*, the best ones are kept and the worst ones "killed" by replacing them with the best individuals' offspring.

Some biologically inspired techniques may be combined. Genetic algorithms may be used for evolution of different topologies for neural networks: the algorithm works with a population of networks, the best ones "survive".

7.3 Genetic Algorithms

Genetic algorithms were introduced by John Holland in the 1960. GA provide an approach to learning that is based on evolution. The search for an appropriate hypothesis begins with a population of hypotheses, the members of which are referred to as individuals. Usually, the initial population is generated in a random fashion. Members of the current population give rise to the next generation by means of mutations and cross-over.

GAs have been applied successfully to a variety of learning tasks and to other optimization problems. For example, they have been used to learn collection of rules for robot control and to optimize the topology and learning parameters of artificial neural networks.

GAs are easily parallelized and can take advantage of the decreasing costs of powerful computer hardware.

The problem addressed by GAs is to search a space of candidate hypotheses to identify the best hypothesis. This is defined as the one that optimizes a predefined measure called the *fitness*. If the task is to learn a strategy for playing a specific game, fitness could be the number of games won by the individual when playing against the others in the population. If the task is to classify unknown samples of some kind, the fitness could be the number of correct classifications in the set of training samples.

7.3.1 GA decisions

These are the most important decisions to be made:

1. **Encoding:** Most GA applications use fixed-length binary strings, but there are multi-character encodings with good results and in genetic programming tree encoding schemes without limits have been used. There are no rigorous guidelines for predicting which encoding that will work best, so how is one to decide on an encoding?

   Lawrence Davis, a researcher with much experience applying GA to real world problems, strongly advocates using "whatever encoding is the most natural for your problem" and then devising a GA that can use that encoding.[1]

2. **Fitness Measure:** In any GA application, a fitness measure is needed in order to compare the quality of different individuals. In our case, the fitness measure is simply the number of samples that the individual classifies correctly as the fitness, but as short rules are preferable (Occam's razor), a penalty proportional to the length of the individual is added so that a short but equally correct individual always gets a higher fitness value. The length of an individual is the number of genes, regardless of how many conditions the individual consists of.

3. **Selection and Replacement:** There exists many ways of selecting individuals in the population that will create offspring for the next generation. The purpose of selection is of course to emphasize the fitter individuals in hopes that their offspring will have even higher fitness. Selection has to be balanced so that sub-optimal individuals do not take over the whole population and that the evolution does not become too slow.

   The biggest difference in selection methods[2] is whether the whole population should be evaluated (for fitness) before choosing which individuals are to be kept and maybe produce offspring, thus creating a new generation, or whether some individuals are picked at random and compared, thus changing just a couple of individuals at a time. This last method is a version of *tournament selection*. The tournament size (how many individuals that are compared at a time) is usually two in GAs. Tournament selection is, often, computationally more efficient than direct fitness-proportionate selection, and is used in this study.

   One has to distinguish between selection and replacement: replacement is the way in which one chooses the individuals that will be replaced. In the GA in this study, the less fit of the two picked individuals will always be replaced by offspring of the kept more fit individual. In a GA that evaluates the whole population before selecting the individuals that will be part of the new generation, the individuals that are mutated could be replacing themselves.

4. **Genetic Operators:** The population evolves through changes in the individuals, and these may be of two kinds[3]: *mutations* that changes one or more details in the individual, or *crossover* in which two individuals exchanges parts (just as in humans) so that the offspring is a mix of the parents. As the encoding permits a wide range of different kinds of indivuals (several genes in several conditions of several kinds), this first version just work with mutations, but to make the mutations work on different parts of the individual (changing a gene, or removing/adding genes and/or conditions etc.).

5. **Parameters:** Let us start with something that is special to this GA: the way genes are picked for inclusion in the individual (rule). As some genes may be more important, a relevance list of the genes have been included so that the user can either pick genes with equal probability or a probability proportional to this relevance measure. (Several relevance measures may be used, letting the GA work with information from previous calculations/tests/studies etc.)

   Then there is *population size, mutation rate and crossover rate* and a termination criterion.

[1] *Handbook of Genetic Algorithms*, ed. Davis 1991, quoted by M. Mitchell. Mitchell also states that most research is currently done by guessing at an appropriate encoding and she also says that choosing an encoding ahead of time presents a paradox: for any problem that is hard enough that one would want to use a GA, one does not know enough about the problem ahead of time to come up with the best encoding. That's one reason for us to not start with encoding the most complex boolean functions.

[2] Please see M. Mitchell and other authors for details on the different fitness-proportionate selection methods ("roulette-wheel", sigma scaling, elitism, Boltzmann and rank selection etc.).

[3] Actually, there are other kinds of genetic operators as well, please see M. Mitchell for examples of these.

The first two typically interact with each other non-linearly, so they cannot be optimized one at a time. Unfortunately there are no "best" parameter settings, so researchers often use what has worked well before. (A population size of about 100 is often sufficient, in GP the populations are much bigger.) Crossover rates are typically 0.95 and mutations 0.01, but as crossover is not used a somewhat higher mutation rate is used as default (0.05), also different kinds of mutations are applied. (An idea is of course to use a GA for self-adaption of parameter values!) Termination criteria is added to be able to make several runs on the same data without having to restart the GA.

7.3.2 Monitoring the Evolution

To monitor the evolution of individuals certain measures can be used[4]:

- Fitness
  1. Fitness of best individual.
  2. Average fitness of entire population.
  3. Variance of the fitness, fitness histogram, entropy of fitness.

  But there is also a structural difference among individuals with the same fitness, so the "diversity" may also be a measure of the state of the population.

- Diversity
  1. Structural differences (*genotype*) – number of genes and conditions in rule
  2. Behavioural differences (*phenotype*) – in this GA the same as fitness.

What is the purpose of observing diversity, either in the structure or behaviour? The reason one would want information about the status of diversity of the population is simply that it helps estimate the chances that continuing the run would have some prospect of discovering a solution to the problem (high diversity is better).

[4]See *Genetic Programming – an introduction* (pp. 221), by Banzhaf, Francone, Keller, Nordin, 1999.

Bibliography

[1] Alizadeh, A. et al.: "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling", Nature 403:503-11, 2000. http://llmp.nih.gov/lymphoma/

[2] Alter, O., Brown, P.O., Botstein, D.: "Singular value decomposition for genome-wide expression data processing and modeling", PNAS Vol.97, No. 18, pp. 10101–10106, August 29, 2000. http://www.pnas.org/

[3] Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., Levine, A.J.: "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays", PNAS Vol.96, pp. 6745-6750, June 1999. http://www.pnas.org/

[4] Ben-Dor, A., Yakhini, Z.: "Clustering gene expression patterns", REKOMB99: Proceedings of the Third Annual Int. Conf. on Computational Mol. Biology, Lyon, France, 1999.

[5] Ben-Dor, A., Friedman, N., Yakhini, Z.: "Class Discovery in Gene Expression Data" Proceedings of Fifth International Conference on Computational Molecular Biology, (RECOMB 2001), April 2001. http://www.cs.technion.ac.il/Labs/cbl/ /publications/abstracts.html.

[6] Bittner, M., Meltzer, P., Chen, Y., Jiang, Y., Seftor, E., Hendrix, M., Radmacher, M., Simon, R., Yakhini, Z., Ben-Dor, A., Sampas, N., Dougherty, E., Wang, E., Marincola, F., Gooden, C., Lueders, J., Glatfelter, A., Pollock, P., Carpten, J., Gillanders, E., Leja, D., Dietrich, K., Beaudry, C., Berens, M., Alberts, D., Sondak, V., Hayward, N., Trent, J.: "Molecular classification of cutaneous melanoma by gene expression profiling", Nature, vol.406, 3 August 2000.

[7] Brown, M.P.S., Grundy, W.N., Lin, D., Cristiani, N., Sugnet, C.W., Furey, T.S., Ares, M., Haussler, D.: "Knowledge-based analysis of microarray gene expression data by using support vector machines", PNAS, Vol. 97, pp. 262-267, January 2000. http://www.pnas.org (Tech report UCSC-CRL-99-09)

[8] Butte, A.J., Ye, J., Niederfellner, G., Rett, K., Hring, H.U., White, M.F., Kohane, I.S.: "Determining significant fold differences in gene expression analysis", PSB2001, http://psb.stanford.edu/psb-online/

[9] Butte, A.J., Kohane, I.S.: "Mutual Information Relevance Networks: Functional Genomic Clustering Using Pairwise Entropy Measurements", Pacific Symposium on Biocomputing 5:415-426 (2000). http://psb.stanford.edu/psb-online/

[10] Cai, J., Dayanik, A., Hasan, N., Terauchi, T., Yu, H.: "Supervised machine learning algorithms for classification of cancer tissue types using microarray gene expression data", (2000). http://www.cpmc.columbia.edu/homepages/ jic7001/cs4995/project1.htm.

[11] Chen, Y., Dougherty, E., Bittner, M.: "Ratio-based decisions and the quantitative analysis of cDNA microarray images", Journal of Biomedical Optics 2(4), 364-374, October 1997.

[12] Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P.O., Herskowitz, I.: "The transciptional program of sporulation in budding yeast", Science 282, 699–705, 1998.

[13] Cho, R.J., Campbell, M.J., Winzeler, E.A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T.G., Gabrelian, A.E., Landsman, D., Lockhart, D.J., Davis, R.W.: "A genome-wide transcriptional analysis of the mitotic cell cycle", Molecular Cell, **2**, 65–73, 1998.

[14] DeRisi, J.L., Iyer, V.R., Brown, P.O.: "Exploring the metabolic genetic control of gene expression on a genomic scale", Science, Vol. 278, pp. 680-686, 24 Oct. 1997. http://www.sciencemag.org/

[15] D'Haeseleer, P., Wen, X., Fuhrman, S., Somogyi, R.: "Linear modeling of mRNA expression levels during CNS development and injury", PSB 4:41–52, 1999. http://psb.stanford.edu/psb-online/

[16] Dudoit, S., Yang, Y.H., Callow, M.J., Speed, T.P.: "Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments", UC Berkeley, Technical report 578, August 2000. http://stat.berkeley.edu/tech-reports/

[17] Dudoit, S., Fridlyand, J., Speed, T.: "Comparison of Discrimination methods for the classification of tumors using gene expression data", UC Berkeley, Technical report 576, June 2000. http://stat.berkeley.edu/tech-reports/

[18] Dutilh, B.: "Analysis of data from microarray experiments, the state of the art in gene network

reconstruction", Literature thesis, Theorethical biology and Bioinformatics, Utrecht University, October 1999.
http://www-binf.bio.uu.nl/~dutilh/gene-networks/

[19] Efron, B., Tibshirani, R., Goss, V., Chu, G.: "Microarrays and their use in a comparative experiment", Stanford Univ., Tech report, October 23, 2000.
http://www-stat.stanford.edu/~tibs/research.html

[20] Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D.: "Cluster analysis and display of genome-wide expression patterns", PNAS, Vol. 95, pp. 14863–14868, December 1998. http://www.pnas.org/

[21] Fraser, A.G., Kamath, R.S., Zipperlen, P., Martinez-Campos, M., Sohrmann, M., Ahringer, J.: "Functional genomic analysis of C. elegans chromosome I by systematic RNA interference", Nature, Vol. 408, 16 November 2000.
http://www.nature.com [Assigns function to 13% of analysed genes, increasing known genes with phenotypes from 70 to 378.]

[22] Fuhrman S., Cunningham, M.J., Wen, X., Zweiger, G., Seilhamer, J., Somogyi, R.: "The application of Shannon entropy in the identification of putative drug targets", BioSystems, Vol.55, pp. 5–14, February 2000. http://www.elsevier.com/locate/biosystems

[23] Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D., Lander, E.S.: "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring", Science, Vol. 286, 15 October 1999. http://www.sciencemag.org

[24] Goss Tusher, V., Tibshirani, R., Chu, C.: "Significance analysis of microarrays applied to the ionizing radiation response", Stanford, 2000, PNAS 2001, Vol.98, pp. 5116–5121, April 24, 2001.

[25] Hastie, T., Tibshirani, R., Botstein, D., Brown, P.: "Supervised harvesting of expression trees", Tech report, Stanford Univ., August 23, 2000. http://www-stat.stanford.edu/~tibs/research.html

[26] Hastie, T., Tibshirani, R., Eisen, M., Brown, P., Ross, D., Scherf, U., Weinstein, J., Alizadeh, A., Staudt, L., Botstein, D.: "Gene shaving: a new class of clustering methods for expression arrays", Tech. report, Stanford Univ., 14 Jan. 2000. http://www-genome.stanford.edu/nci60/
http://www-stat.stanford.edu/~tibs/research.html

[27] Jonsson, P.: "Using functional annotation to improve clustering of gene expression patterns", Bioinformatics 2001, Skövde (Sweden) March 29, Univ. of Skövde 2001.
http://www.ida.his.se/ida/bioinformatics2001/
/abstracts_collection.html

[28] Kaminski, N., Allard, J.D., Pittet, J.F., Zuo, F., Griffiths, M.J.D., Morris, D., Huang, X., Sheppard, D., Heller, R.A.: "Global analysis of gene expression in pulmonary fibrosis reveals distinct programs regulating lung inflammation and fibrosis", PNAS, Vol.97, pp. 1778–1783, February 2000. http://www.pnas.org/

[29] Kepler, T.: "Normalization and analysis of DNA microarray data by self-consistency and local regression", Functional genomics, IPAM, UCLA, fall 2000.
http://www.ipam.ucla.edu/programs/fg2000/
/abstracts/fgsn_tkepler.html [Ref by Yang]

[30] Kerr, M.K., Martin, M., Churchill, G.A.: "Analysis of variance for gene expression microarray data", July 2000, manuscript. http://www.jax.org/research/churchill/

[31] Kerr, K., Churchill, G.: "Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments", The Jackson Laboratory, Bar Harbor, ME, September(?) 2000. http://www.jax.org/research/churchill/

[32] Kerr, K., Churchill, G.: "Experimental design for gene expression microarrays", The Jackson Laboratory, Bar Harbor, ME, revised August 2000.
http://www.jax.org/research/churchill/

[33] Kerr, K., Churchill, G.: "Statistical design and the analysis of gene expression microarray data", The Jackson Laboratory, 2001.

[34] Kerr, K., Martin, M., Churchill, G.: "Analysis of variance for gene expression microarray data", The Jackson Laboratory, Bar Harbor, ME, revised July 2000. http://www.jax.org/research/churchill/

[35] Kerr, K., Leiter, E.H., Churchill, G.: "Analysis of designed microarray experiment", The Jackson Laboratory, 2001.

[36] Kerr, K., Afshari, C.A:, Bennett, L., Bushel, P., Martinez, J., Walker, N., Churchill, G.: "Statistical analysis of a gene expression microarray experiment with replication", The Jackson Laboratory, 2001.

[37] Koza, J.R., Mydlowec, W., Lanza, G., Yu, J., Keane, M.A.: "Reverse engineering of metabolic pathways from observed data using genetic programming", abstract, ICSB 2000, Tokyo, November 2000.
http://www.systems-biology.org/,
http://mitpress.mit.edu/ICSB2000/.

[38] Lanza, G., Mydlowec, W., Koza, J.R.:"Automatic creation of a genetic network for the LAC operon from observed data by means of genetic programming", abstract, ICSB 2000, Tokyo, November 2000. http://www.systems-biology.org/,

http://mitpress.mit.edu/ICSB2000/. [Uses time-domain expr. levels of two genes and concentrations of two substances.]

[39] Lazzeroni, L., Owen, A.: "Plaid models for gene expression data", preprint, 2000. http://www-stat.stanford.edu/~owen/reports/

[40] Lee, M., Kuo, F., Whitmore, G., Sklar, J.: "Importance of replication in microarray gene expression studies: statistical methods and evidence from repetitive cDNA hybridizations", PNAS Vol.97, Aug 29, 2000.

[41] Lönnstedt, I., Speed, T.: "Replicated microarray data", submitted, June 1, 2001. [Compares four different forms of statistics. Presents an empirical Bayes method which combine data from all the genes in a replicate set of experiments into estimates of parameters of a prior distribution. These are then combined at the gene level with means and standard deviations to form a statistic B which can be used to decide whether differential expression has occurred.]

[42] Lee, C-K., Klopp, R.G., Weindruch, R., Prolla, T.A.: "Gene expression profile of aging and its retardation by caloric restriction", Science, Vol. 285, pp. 1390–1393, 27 Aug. 1999. http://www.sciencemag.org/

[43] MacBeath, G., Schreiber, S.L.: "Printing proteins as microarrays for high-throughput function determination", Science, Vol. 289, 8 September 2000. http://www.sciencemag.org

[44] Michaels, G.S., Carr, D.B., Askenazi, M., Fuhrman, S., Wen, X., Somogyi, R.: "Cluster analysis and data visualization of large-scale gene expression data", PSB 3:42–53 1998. http://psb.stanford.edu/psb-online/

[45] Mjolsness, E., Castaño, R., Mann, T., Roden, J., Gray, A., Wold, B.: "Clustering Methods for the Analysis of C. elegans Gene Expression Array Data", JPL-NASA, Caltech, 1999. http://www-aig.jpl.nasa.gov/public/ /mls/mls_papers.html

[46] Morgan, B.J.T., Ray, A.P.G., Appl. Stat. 44:117-134, 1995. [Critique of hierarchical clustering. Lack of robustness, nonuniqueness, have inversion problems that complicate interpretation of hierarchical. Trees can lock in accidental features. Ref. by Tamayo.]

[47] Morohashi, M. et al.: "Identifying gene regulatory networks from time series expression data by in silico sampling and screening", Proc. the Fifth European Conference on Artificial Life, pp. 477-486 (1999).

[48] Newton, M.A., Kendziorski, C.M., Richmond, C.S., Blattner, F.R., Tsui, K.W.: "On differential variability of expression ratios: Improving statistical inference about gene expression changes from microarray data", Journal of Computational Biology, 6/2000, revision of Technical report 139, Dept. of Biostatistics and Medical informatics, Univ. of Wisconsin Madison 1999/2000. http://www.stat.wisc.edu/~newton/papers/ /abstracts/btr139a.html

[49] Perou, C.M. et al. (17 persons): "Molecular portraits of human breast tumours", Nature, Vol. 406, 17 August, 2000. http://www.nature.com

[50] Raychaudhuri, S., Stuart, J.M., Altman, R.B.: "Principal components analysis to summarize microarray experiments: application to sporulation time series", PSB 5:452–463, 2000. http://psb.stanford.edu/psb-online/

[51] Roberts, J.R., Nelson, B., Marton, M.J., Stoughton, R., Meyer, M.R., Bennett, H.A., He, Y.D., Dai, H., Walker, W.L., Hughes, T.R., Tyers, M., Boone, C., Friend, S.H.: "Signaling and circuitry of multiple MAPK pathways revealed by a matrix of global gene expression profiles", Science, Vol. 287, pp. 873-880, 4 Feb. 2000. http://www.sciencemag.org/ [Uses fold change plots and 2-dim hierarchical clustering to reveal new genes, analyzing similarities between experiments and studying internal feedback loops. Good fig. to illustrate red-green-clustering. "Genome-wide transcriptional profiling may trace the signaling mech. and circuits that underlie complex biological responses."]

[52] Ross et al.: "Systematic variation in gene expression patterns in human cancer cell lines", Nature Genetics , March 2000, 24(3): 227–234. http://www-genome.stanford.edu/nci60/ http://www.nature.com/

[53] Sapir, M., Churchill, G.: "Estimating the posterior probability of differential gene expression from microarray data", The Jackson Laboratory, Bar Harbor, ME, 2000(?). http://www.jax.org/research/churchill/

[54] Scherf, U. et al. (+ 16 persons): "A gene expression database for the molecular pharmacology of cancer", Nature Genetics, Vol. 24, March 2000. http://genetics.nature.com/ http://discover.nci.nhi.gov/nature2000/ http://www-genome.stanford.edu/nci60/ [Gene expression profiles from 60 human cancer cell lines clustered very differently from those clustered visavi response to drugs. First study to integrate large databases on gene expression and molecular pharmacology. Data: t-matrix. Average-link hierarchical cluster alg. with a correlation metric. Stat anal. in S-plus where scripts generated html-docs that could be looked at in a browser.]

[55] Shamir, R., Sharan, R.: "CLICK: A clustering algorithm with applications to gene expression analysis", Proc. 8th Int. Conference on Intelligent Systems for Molecular Biology (ISMB'00), pp.260–

268, AAAI Press, Menlo Park, 2000.
http://www.math.tau.ac.il/~roded/click.html

[56] St. Croix, B., Rago, C., Velculescu, V., Traverso, G., Romans, K.E., Montgomery, E., Lal, A., Riggins, G.J., Lengauer, C., Vogelstein, B., Kinzler, K.W.: "Genes expressed in human tumor endothelium", Science, Vol. 289, pp. 1197–1202, 18 Aug. 2000. http://www.sciencemag.org/ [SAGE, simple analysis of ESTs, comparing normal and tumor cells. No plots etc.]

[57] Szallasi, Zoltan: "Gene expression patterns and cancer", Nature Biotechnology, Vol. 16, December 1998. http://www.nature.com

[58] Tavazoie, S., Hughes, J., Campbell, M., Cho, R., Church, G.: "Systematic determination of genetic network architecture", Nature Genetics Vol. 22, pp.281–285, 1999.

[59] Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E.S., Golub, T.R.: "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation", PNAS Vol.96, pp. 2907–2912, March 1999. http://www.pnas.org/

[60] Tibshirani, R., Walther, G., Hastie, T.: "Estimating the number of clusters in a dataset via the Gap statistic", Stanford Univ. Tech Report, March 29, 2000. http://www-stat.stanford.edu/~tibs/research.html

[61] Tibshirani, R., Hastie, T., Eisen, M., Ross, D., Botstein, D., Brown, P.: "Clustering methods for the analysis of DNA microarray data", Stanford Univ. 15 Oct. 1999. http://www-stat.stanford.edu/~tibs/research.html

[62] Ueda, N., Taisuke, S.: "Finding original regulatory networks with weight matrices", abstract, ICSB 2000, Tokyo, November 2000. http://www.systems-biology.org/, http://mitpress.mit.edu/ICSB2000/.

[63] Wahde, M., Hertz, J.: "Coarse-grained reverse engineering of genetic regulatory networks", Biosystems 2000, IPCAT99, Preprint submitted to Elsevier Preprint, November 1999.

[64] Wahde, M., Szallasi, Z.: "Generative Model Based Analysis of Cancer Associated Gene Expression Matrices", Proceedings of the ICSB 2000.

[65] Velculescu, V.E.: "Tantalizing transcriptomes – SAGE and its use in global gene expression analysis", Science, Vol. 286, pp. 1491–1492, 19 Nov. 1999.
http://www.sciencemag.org/

[66] Wen, X., Fuhrman, S., Michaels, G.S., Carr, D.B., Smith, S., Barker, J.L., Somogyi, R.: "Large-scale temporal gene expression mapping of CNS development", PNAS, 95:334-339, 1998. http://www.pnas.org/

[67] Yang, Y.H., Buckley, M.J., Dudoit, S., Speed, T.: "Comparison of methods for image analysis on cDNA microarray data", November 2000.
http://www.stat.berkeley.edu/users/terry/
/zarray/Html/papersindex.html

[68] Yang, Y., Dudoit, S., Luu, P., Speed, T.: "Normalization for cDNA microarray data", SPIE BiOS 2001, San Jose, California, January 2001.
http://www.stat.berkeley.edu/users/terry/
/zarray/Html/papersindex.html

[69] Yeung, K.Y., Haynor, D.R., Ruzzo, W.L.: "Validating clustering for gene expression data", Bioinformatics 2001, volume 17, number 4, pp. 309–318.

Books:

[70] Alberts, Bray, Lewis, Raff, Roberts and Watson: "Molecular biology of the cell", 3rd ed., Garland Publ., 1994.

[71] Ptashne, M.: "A Genetic Switch, gene control and phage lamda", Cell Press and Blackwell Scientific publ., 1 ed., 1986.

[72] Kauffman, S.: "The Origins of Order, self-organization and selection in evolution", Oxford University Press, 1993.

[73] Jain, A.K., Dubes, R.C.: "Algorithms for clustering data", Prentice Hall, 1988.

[74] Baldi, P., Brunak, S.: "Bioinformatics, the machine learning approach", MIT Press, 1998.

[75] Mitchell, M., "An introduction to genetic algorithms", MIT Press, 1996.

[76] Mitchell, T., "Machine Learning", McGraw-Hill, 1997.

[77] Haykin, S., "Neural Networks, a comprehensive foundation", Prentice-Hall, 1999.

[78] Banzhaf, W., Francone, F.D., Keller, R.E., Nordin, P., "Genetic Programming, an introduction", Morgan Kaufman Publ., 1998.

URLs:

- Genomics and bioinformatics group, National Cancer Institute, http://discover.nic.nih.gov/.
- Stanford Genomic Resources, http://www-genome.stanford.edu/
- Central Nervous System group (Somogyi, Fuhrman, Wen, Chang, Smith) http://rsb.info.nih.gov/mol-physiol/
- Microarray suppliers:
  http://www.incyte.com/
  http://www.mergen-ltd.com/
  http://www.affymetrix.com/