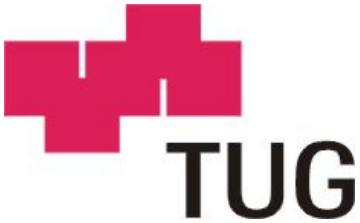
Alexander Sturn

Cluster Analysis for Large Scale Gene Expression Studies

master thesis

Conducted at the [1]Institute for Biomedical Engineering,
Graz University of Technology, Inffeldgasse 18, 8010 Graz, Austria and
[2]The Institute for Genomic Research, 9712 Medical Center Drive,
Rockville, Maryland 20850, United States of America



Head of Institute: Univ.-Prof. Dipl.-Ing. Dr. techn. Gert Pfurtscheller[1]
Supervisor: ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Zlatko Trajanoski[1]
Advisor: John Quackenbush, Ph.D., Associate Investigator[2]

Rockville, Maryland, USA, December 20th, 2000

for my parents

Cluster Analysis for Large Scale Gene Expression Studies

High throughput gene expression analysis is becoming more and more important in many areas of biomedical research. cDNA microarray technology is one very promising approach for high throughput analysis and provides the opportunity to study gene expression patterns on a genomic scale. Thousands or even tens of thousands of genes can be spotted on a microscope slide and relative expression levels of each gene can be determined by measuring the fluorescence intensity of labeled mRNA hybridized to the arrays. Beyond simple discrimination of differentially expressed genes, functional annotation (guilt-by-association) or diagnostic classification requires the clustering of genes from multiple experiments into groups with similar expression patterns. A platform independent Java package of tools has been developed to simultaneously visualize and analyze a whole set of gene expression experiments. After reading the data from flat files several graphical representations of hybridizations can be generated, showing a matrix of experiments and genes, where multiple experiments and genes can be easily compared with each other. Fluorescence ratios can be normalized in several ways to gain a best possible representation of the data for further statistical analysis. Hierarchical and non hierarchical algorithms have been implemented to identify similar expressed genes and expression patterns, including: 1) hierarchical clustering, 2) k-means, 3) self organizing maps, 4) principal component analysis, and 5) support vector machines. More than 10 different kinds of similarity distance measurements have been facilitated, ranging from simple Pearson correlation to more sophisticated approaches like mutual information. Moreover, it is possible to map gene expression data onto chromosomal sequences. The flexibility, variety of analysis tools and data visualizations makes this software suite a valuable tool in future functional genomic studies.

Keywords: microarray, cluster analysis, genomics, bioinformatics, java

Clusteranalyse von umfangreichen Genexpressionsdaten

Die Hochdurchsatz-Genexpressionsanalyse wird immer wichtiger in vielen Bereichen der biomedizinischen Forschung. Die cDNA Micorarray-Technologie ist ein sehr vielversprechender Ansatz für Hochdurchsatzanalyse und bietet die Möglichkeit, Genexpressionsmuster auf Genomebene zu studieren. Tausende oder sogar zehntausende von Genen können auf ein Mikroskopplättchen gedruckt werden, und die relative Expression von jedem Gen kann durch die Messung der Fluoreszenzintensität der zu den Arrays hybridisierten und markierten RNA gemessen werden. Geht man über die simple Unterscheidung von unterschiedlich exprimierten Genen hinaus, benötigen die funktionelle Annotation oder diagnostische Klassifikation das Clustern von Genen von multiplen Experimenten in Gruppen von ähnlich exprimierten Genen. Es wurde ein plattform-unabhängiges Java-Paket von Werkzeugen für die simultane Visualisierung und Analyse von ganzen Genexpressionsexperimentensets entwickelt. Nachdem die Daten eingelesen worden sind, können verschiedene graphische Repräsentationen der Hybridisierungen erstellt werden, die eine Matrix von Experimenten und Genen zeigt, mit der mehrere Experimente und Gene leicht miteinander verglichen werden können. Die Fluoreszenzverhältnisse können auf mehrere Arten normalisiert werden, um eine bestmögliche Repräsentation der Daten für die weitere statistische Auswertung erlangen zu können. Es wurden hierarchische und nicht hierarchische Clusterverfahren für die Identifikation von ähnlich exprimierten Genen und Expressionsmustern implementiert, einschließlich: 1.) Hierarchisches Clustern, 2) k-means, 3.) Self Organizing Maps, 4) Principal Component Analysis und 5) Support Vector Machines. Mehr als 10 verschiedene Arten von Ähnlichkeitsdistanzmessungen wurden implementiert, von der simplen Pearson Korrelation zu aufwendigeren Verfahren wie Mutual Information. Weiters ist es möglich, Genexpressionsdaten auf chromosomale Sequenzen zu projizieren. Die Flexibilität, Auswahl an Analysewerkzeugen und Datenvisualisierungen machen diese Software zu einem wertvollen Werkzeug für zukünftige Studien von Genomen.

Schlüsselworte: Mikroarray, Clusteranalyse, Genomische Forschung, Bioinformatik, Java

Table of Content

Table of Figures

1 Introduction

2 Methods

3 Results

Glossary

API Application Programming Interface

Base Pair Two bases which form a "rung of the DNA ladder." A DNA nucleotide is made of a molecule of sugar, a molecule of phosphoric acid, and a molecule called a base. The bases are the "letters" that spell out the genetic code. In DNA, the code letters are A, T, G, and C, which stand for the chemicals adenine, thymine, guanine, and cytosine, respectively. In base pairing, adenine always pairs with thymine, and guanine always pairs with cytosine.

BMU Best Matching Unit of an SOM.

cDNA Complementary DNA, just one strain of DNA

Chromosome One of the threadlike "packages" of genes and other DNA in the nucleus of a cell. Different kinds of organisms have different numbers of chromosomes. Humans have 23 pairs of chromosomes, 46 in all: 44 autosomes and two sex chromosomes. Each parent contributes one chromosome to each pair, so children get half of their chromosomes from their mothers and half from their fathers.

Codon Three bases in a DNA or RNA sequence which specify a single amino acid.

Cy3, Cy5 Fluorescent dyes used in microarray technology

DNA Deoxyribonucleic acid. The chemical inside the nucleus of a cell that carries the genetic instructions for making living organisms.

DNA sequencing Determining the exact order of the base pairs in a segment of DNA.

Double helix The structural arrangement of DNA, which looks something like an immensely long ladder twisted into a helix, or coil. The sides of the "ladder" are formed by a backbone of sugar and phosphate molecules, and the "rungs" consist of nucleotide bases joined weakly in the middle by hydrogen bonds.

Gene The functional and physical unit of heredity passed from parent to offspring. Genes are pieces of DNA, and most genes contain the information for making a specific protein.

Gene expression The process by which proteins are made from the instructions encoded in DNA.

Genetic code The instructions in a gene that tell the cell how to make a specific protein. A, T, G, and C are the "letters" of the DNA code; they stand for the chemicals adenine, thymine, guanine, and cytosine, respectively, that make up the nucleotide bases of DNA. Each gene's code combines the four chemicals in various ways to spell out 3-letter "words" that specify which amino acid is needed at every step in making a protein.

Genome All the DNA contained in an organism or a cell, which includes both the chromosomes within the nucleus and the DNA in mitochondria.

GUI Graphical User Interface

HC Hierarchical Clustering

HCA Hierarchical Clustering Analysis

HGP Human Genome Project. An international research project to map each human gene and to completely sequence human DNA.

Hybridization Base pairing of two single strands of DNA or RNA.

IDE Integrated Development Environment

JDK Java Development Kit

JIT Just In Time

JRE Java Runtime Environment

JVM Java Virtual Machine

MAML MicroArray Markup Language

MIAME Minimum Information About a Microarray Experiment

MGED Microarray Gene Expression Database group

Microarrays A new way of studying how large numbers of genes interact with each other and how a cell's regulatory networks control vast batteries of genes simultaneously. The method uses a robot to precisely apply tiny droplets containing functional DNA to glass slides. Researchers then attach fluorescent labels to DNA from the cell they are studying. The labeled probes are allowed to bind to complementary DNA strands on the slides. The slides are put into a scanning microscope that can measure the brightness of each fluorescent dot; brightness reveals how much of a specific DNA fragment is present, an indicator of how active it is.

mRNA Messenger RNA. Template for protein synthesis. Each set of three bases, called codons, specifies a certain protein in the sequence of amino acids that comprise the protein. The sequence of a strand of mRNA is based on the sequence of a complementary strand of DNA.

NCBI National Center for Biotechnology Information

Nucleotide One of the structural components, or building blocks, of DNA and RNA. A nucleotide consists of a base (one of four chemicals: adenine, thymine, guanine, and cytosine) plus a molecule of sugar and one of phosphoric acid.

Oligo Oligonucleotide, short sequence of single-stranded DNA or RNA. Oligos are often used as probes for detecting complementary DNA or RNA because they bind readily to their complements.

ORF Open Reading Frame.

PC	Principal Component
PCA	Principal Component Analysis.
PCR	Polymerase Chain Reaction. A fast, inexpensive technique for making an unlimited number of copies of any piece of DNA. Sometimes called "molecular photocopying," PCR has had an immense impact on biology and medicine, especially genetic research.
Primer	A short oligonucleotide sequence used in a polymerase chain reaction.
Protein	A large complex molecule made up of one or more chains of amino acids. Proteins perform a wide variety of activities in the cell.
PWM	Position Weight Matrix
RMS	Root Mean Square
RNA	Ribonucleic acid. A chemical similar to a single strand of DNA. In RNA, the letter U, which stands for uracil, is substituted for T in the genetic code. RNA delivers DNA's genetic message to the cytoplasm of a cell where proteins are made.
SD	Standard Deviation
SDK	Software Development Kit
SOM	Self Organizing Map/Maps.
SVD	Singular Value Decomposition.
SVM	Support Vector Machines.
UPGMA	Unweighted Pair-Group Method using Arithmetic averages
VM	Virtual Machine.

CHAPTER 1

Introduction

Exactly a century ago, in 1900, Hugo Marie de Vries, Carl Erich Correns, and Erich Tschermark von Seysenegg independently rediscovered Gregor Mendel's work on the rules of heredity, which can be seen as the beginning of genetic research. More than a decade passed before Thomas Hunt Morgan redefined those ideas to a concept of heritable genetic units strung along chromosomes [1]. In 1953, Francis Crick and James Watson published their famous one-page paper about the discovery of the double-helical structure of DNA, and that began the process of unlocking the secrets of the molecule [2], leading to our present understanding of the idea it plays in heredity.

The rapid development of sequencing and computer technology in the last years lead to the complete sequencing and annotation of many important model organisms including *Haemophilus influenzae* (first genome of a free living organism, 1.830.137 base pairs [3]), *Saccharomyces cerevisiae* (first eukaryotic genome sequence, 12.068 kilo bases [4]), *Caenorhabditis elegans* (first complete sequence of multicellular organism [5]), *Vibrio cholerae* (4.033.460 base pairs [6]), *Drosophila melanogaster* (120 mega bases [7]), and more than 30 microbial genomes, *Arabidopsis thaliana* (an important model plant [8]), as well as parts of *Mus musculus* (the mouse) and of course *Human* [9, 10].

Figure 1.1: Some of the milestones in Genome research and genetics.

This year, in 2000, both private Celera Genomics and the international government consortium of laboratories called the Human Genome Project (HGP) are releasing first draft versions of the sequence of base pairs in human DNA. By the end of 2002 the complete and error free human DNA sequence will be public available in the databases of the National Library of Medicine and other institutes all over the world.

The next challenge will be to determine the location and function of all genes. The aim is to provide biologists with an inventory of all genes used to assemble a living creature, similar to the chemistry's Periodic Table [11]. Understanding the biological systems with tens of thousands or even 100.000 genes will similarly require organizing the parts by properties and will reflect similarities at diverse levels such as [12]:

- Time and place of RNA expression during development
- Subcellular localization and intermolecular interaction of protein products
- Physiological response and disease
- Primary DNA sequence in coding and regulatory regions and
- Polymorphistic variation within a species or subgroup

Because of the sheer magnitude of the problem, traditional gene-by-gene approaches were not sufficient and new methods and technologies have been invented to read out all components simultaneously. These high throughput gene analysis methods are becoming more and more important in many areas of biological and biomedical research as well as in clinical diagnostics, providing biologist with a more global view of biological processes.

1.1 Microarrays

Microarray technology [13-18] is one very promising approach for high throughput analysis and gives the opportunity to study gene expression patterns on a genomic scale.

It all began about a quarter century ago, with Ed Southern's key insight that labeled nucleic acid molecules could be used to interrogate nucleic acid molecules attached to a solid support [19]. Today, thousands or even tens of thousands of genes can be spotted on a microscope slide and relative expression levels of each gene can be determined by measuring the fluorescence intensity of labeled mRNA hybridized to the arrays, facilitating the measurement of RNA levels for the complete set of transcripts of an organism.

Applied to functional genetics and mutation screening, microarrays give us the opportunity to determine thousands of expression values in hundreds of different conditions [20], allowing the contemplation of genetic processes on a whole genomic scale to determine genetic contributions to complex polygenic disorders and to screen for important changes in potential disease genes [21].

cDNA microarrays exploit the preferential binding of complementary, single stranded nucleic acid sequences. Basically, a microarray is a specially coated glass microscope slide to which cDNA molecules are attached at fixed locations, called spots. With up to date computer controlled high-speed robots 19200 and more spots can be printed on a single slide, each representing a single gene. RNA from control and sample cells is extracted. Fluorescently labeled cDNA probes are prepared by incorporating either Cye-3 or Cye-5-dUTP using a single round of reverse transcription, usually taking the red dye for RNA from the sample cells and the green dye for that from the control population. Both extracts are simultaneously incubated on the microarray, enabling the gene sequences to hybridize under stringed conditions to their complementary clones attached to the surface of the array [22,23].

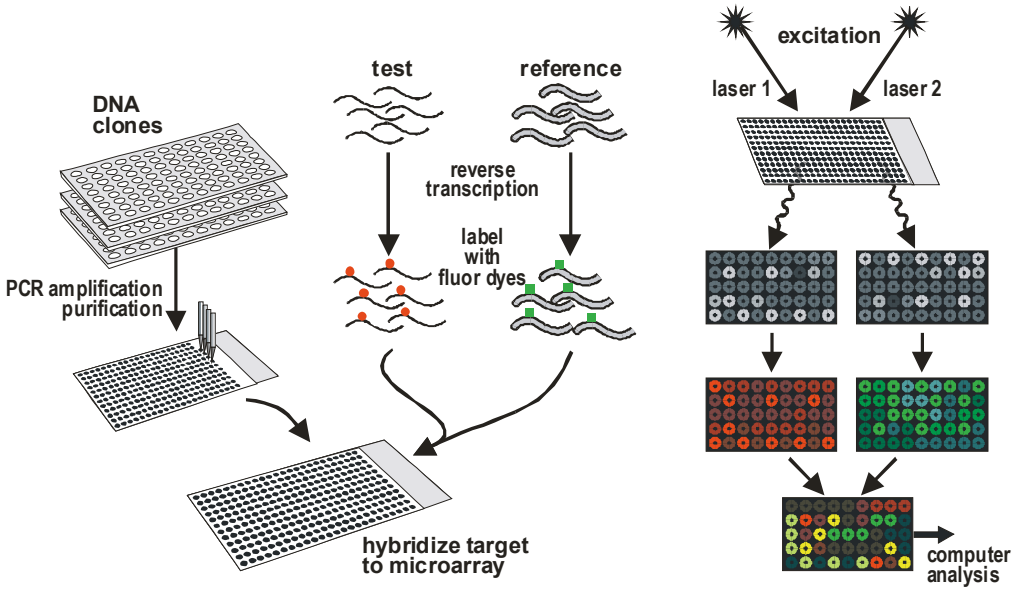


Figure 1.2: cDNA microarray schema. First DNA clones are spotted onto a microscopic glass slide. After hybridization the slide is scanned using laser excitation resulting in two images as a basis for further analysis [23].

Laser excitation of the incorporated targets yield an emission with a characteristic spectra, which is measured using a scanning confocal laser microscope. Monochrome images from the scanner are then imported into software in which they are pseudo-colored and merged. A spot will for instance appear red, if the corresponding RNA from the sample population is in greater abundance and green, if the control population is in greater abundance. If both are equal, the spots will appear yellow, if neither binds, the spot will appear black. Thus, the relative gene expression levels of sample and reference populations can be estimated from the fluorescence intensities and colors emitted by each spot during scanning.

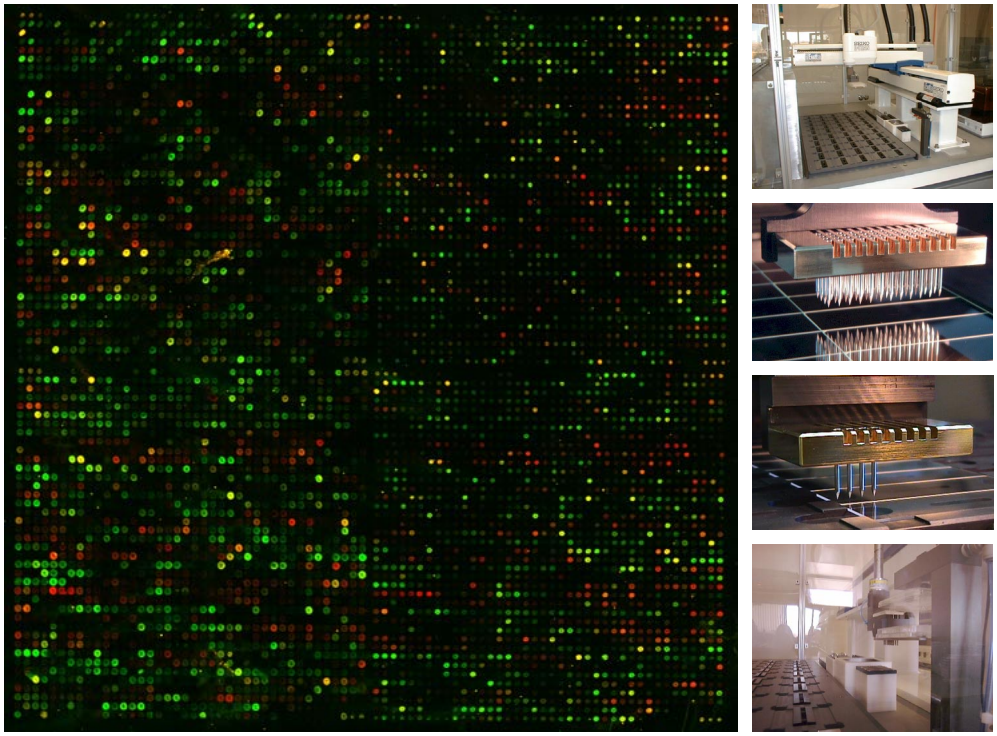


Figure 1.3: (left) Scan of a cDNA microarray containing the whole yeast genome. (right) microarray spotting device at The Institute for Genomic Research (top, bottom) and examples of commonly used print heads (middle).

The production and hybridization of slides is just one pace in a pipeline of many steps necessary to gain meaningful information from microarray experiments. Because of the vast amount of data produced by a microarray experiment, sophisticated software tools are used to normalize and analyze the data [24, 25].

First the scanned images are analyzed using image analysis software, which evaluates the expression of a gene by quantifying the ratio of the fluorescence intensities of a spot. The quantified intensities provide information about the activity of a specific gene in a studied cell or tissue. High intensity means high activity, low intensity indicates low or no activity.

The next step is to extract the fundamental patterns of gene expression inherent in the data in a mathematical process called clustering, which organizes the genes into biological relevant clusters with similar expression patterns (coexpressed genes). There are three reasons for interest in coexpressed genes [26-28]:

First, there is evidence that many functionally related genes are coexpressed [29, 30]. For example, genes coding for elements of a protein complex are likely to have similar expression

patterns. Hence, grouping genes with similar expression levels can reveal the function of those which were previously uncharacterized (guilt-by-association).

Second, coexpressed genes may reveal much about regulatory mechanisms. For example, if a single regulatory system controls two genes, then the genes are expected to be coexpressed. In general there is likely to be a relationship between coexpression and coregulation.

Third, gene expression levels differ in various cell types and states. The interest is in how gene expression is changed by various diseases or compound treatments, respectively. For example one can investigate the differences in gene expression between a normal and a cancer cell [31-39].

Several clustering techniques were recently developed and applied to analyze microarray data. However, to the best of my knowledge, there is no single tool, which integrates the common clustering and visualization methods. Such a tool would be valuable for comparison and evaluation of clustering algorithms and their result and would help biologists to gain biological meaningful information out of microarray datasets in a less costly way.

1.2 Conceptual formulation

The aim of this work is the implementation and evaluation of several commonly used clustering algorithms for large-scale gene expression data. The program has to fulfill the following requirements:

- Import and export of multiple experiment datasets from flat files
- Graphical representation of the dataset in a user friendly and intuitive way
- Tools for data adjustment to gain a best possible representation for further analysis
- Facilitation of several distance measuring procedures
- Implementation of Hierarchical Clustering, k-means Clustering, Self Organizing Maps (SOM), Principal Component Analysis (PCA) and Support Vector Machines (SVM)
- 2 and 3-dimensional representation of the clustering results including the ability to export the results as images and data.

To show the usefulness of clustering software programs all clustering algorithms will be evaluated by using a public available dataset and verifying the results with published knowledge gained from that dataset.

Chapter 2 - Methods

2.1 Data representation

2.1.1 Expression matrix

The relative expression levels of *n* genes, which may constitute almost the entire genome of an organism, are probed simultaneously by a single microarray. A series of *m* arrays, which are almost identical physically, probe the genome wide expression levels in *m* different samples - i.e., under *m* different experimental conditions. Let the matrix **X**, of size (*n*-genes x *m*-arrays), tabulate the full expression data, where x_{ij} is the $\log_2$ of the expression ratio of the *i*th gene in the *j*th sample as measured by the array *j*. The vector in the *i*th row of the matrix **X** lists the fluorescence ratio of the *i*th gene across the different samples, which correspond to the different arrays. The vector in the *j*th column of the matrix **X**, lists the genome wide fluorescence ratios measured by the *j*th array [40].

$$x_{ij} = \log_2 \frac{C5_{ij}}{C3_{ij}} \tag{2.1}$$

$C5_{ij}$ ..... Cye-5 fluorescence measurement of gene *i* in microarray experiment *j*

$C3_{ij}$ ..... Cye-3 fluorescence measurement of gene *i* in microarray experiment *j*

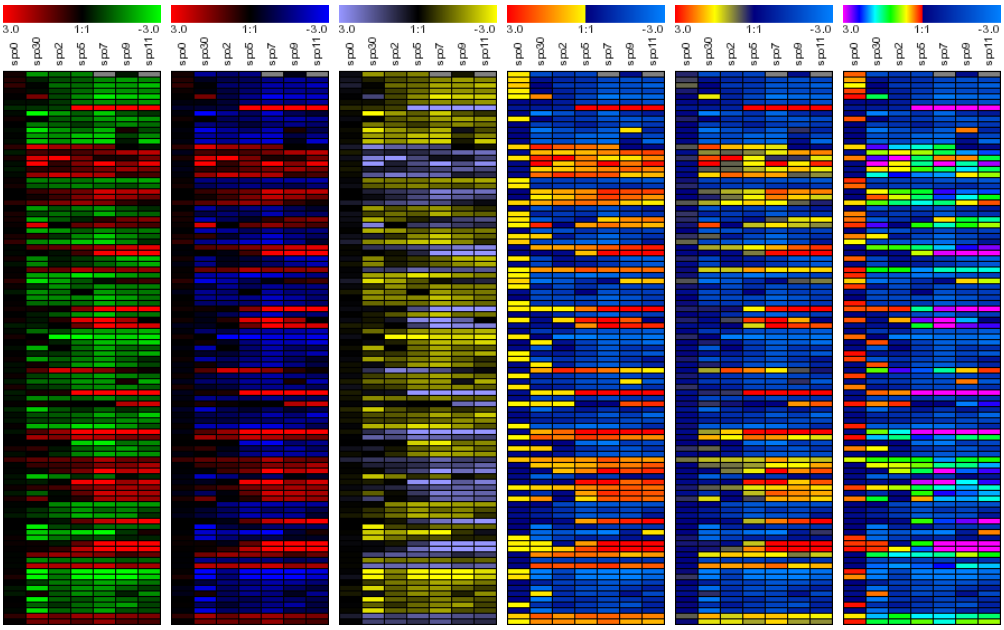
x_{ij} is negative if C3>C5, 0 if C3=C5, and positive if C3<C5. In other words: x_{ij} is positive, if gene *i* in experiment *j* is over expressed, and negative if gene *i* in experiment *j* is under expressed relative to the control sample.

2.1.2 Graphical representation

Because the massive collection of numbers is difficult to assimilate, the primary data is combined with a graphical representation by representing each data point with a color that quantitatively and qualitatively reflects the original experimental observation, i.e. each data point x_{ij} is colored on the basis of the measured fluorescence ratio. This yields to a representation of complex gene expression data that allows biologists to assimilate and explore the data in a more intuitive manner [41].

In the literature used color scales range usually from saturated green (max neg. value) to saturated red (max. pos. value). Cells with log ratio of 0 (genes unchanged) are colored black, increasingly positive log ratios with reds of increasing intensity, and increasing negative log ratios with greens of increasing intensity. Missing values usually appear gray.

However, the green/red color scheme is in many cases not necessarily the ideal choice. Colorblind persons for instance are not able to distinguish green and red colors. It is also very difficult to say if a very low expressed gene is over or under expressed (see first column of the matrix in Figure 2.1). To overcome these drawbacks, 5 additional color schemes have been implemented, each with better properties for a specific problem (Figure 2.1).



(a) (b) (c) (d) (e) (f)

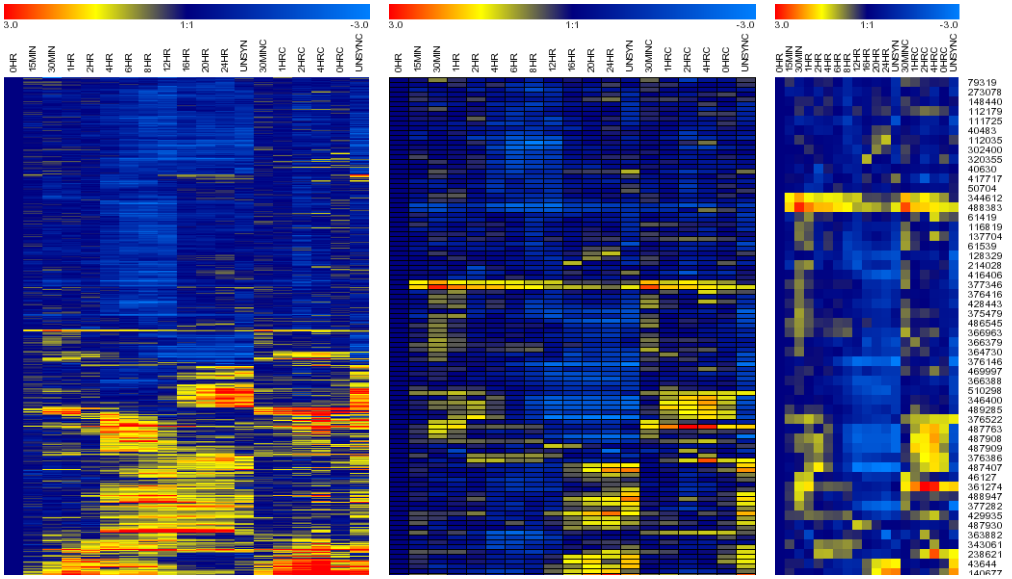
Figure 2.1: 6 different graphical representations of the same primary data set (a) most common color scheme used in literature (b) alternative color scheme for color blind persons (c) more contrast at lower ratio levels (d) for better differentiation of over or under expressed genes at low ratio levels (e) better contrast for over expressed genes (f) false color representation of over expressed genes. Gray fields represent missing values.

A second very important factor is the size of each data point in the graphical representation. This in turn depends on the purpose of the visualization, if the endeavor is to get a good overview of the data, or if a good detail view is favored (Figure 2.2).

It seems that the human brain prefers rectangular structures with an aspect ratio of approximately 4 to 1. Structures and areas of interest are much easier to see, compared to a

data representation with square cells. The best general visibility is gained by using rectangular color cells (20x5 Pixels) in combination with a black raster for a better cell separation (Figure 2.2b).

In contrary, if detail information like accession numbers or general ORF descriptions have to be added to each gene for documentation purposes, the quadratic representation (10x10 Pixels) as often used in publications provide the best results (Figure 2.2c).



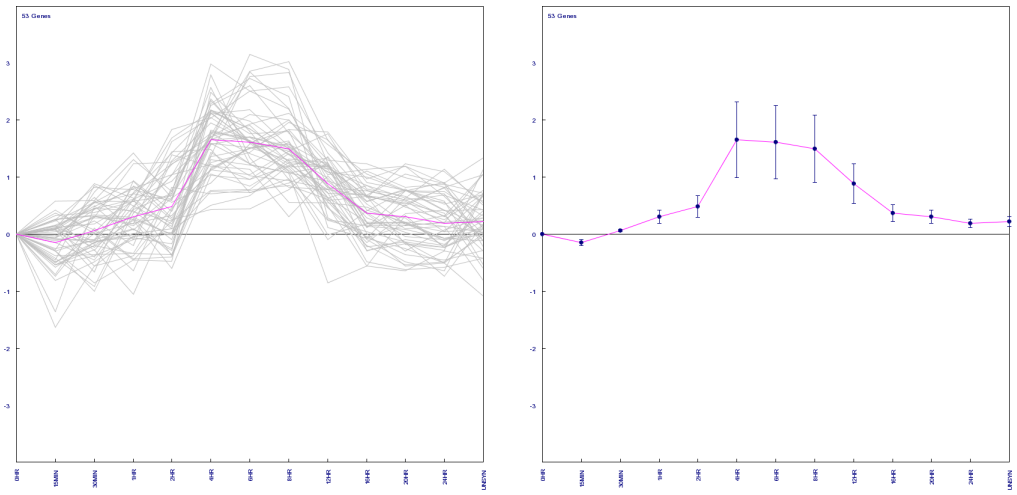
(a) (b) (c)

Figure 2.2: dataset representation (a) 1x20 pixel cells for a good overview of the whole dataset (b) 20x5 pixel cells for a good general overview (c) 10x10 pixel cells for adding additional detail information like gene accession numbers or general ORF descriptions.

If the experiments have a relation to each other like for example a temporal relationship, it is possible to plot the expression progression along different conditions for each gene. This makes of course no sense, if the experiments are independent from each other. If more than one gene is observed, two different kinds of plot representations of the data are common (Figure 2.3):

First: Each gene is plotted separately. This is a very intuitive and easy to read representation. However, if many genes are viewed simultaneously, it becomes very complex and general patterns are difficult to observe.

Second: The mean and normalized standard deviation of all genes are plotted. This so-called centroid view is a less intuitive way of representing trends but it is easier to identify overall properties of complex data.



(a) (b)

Figure 2.3: progression plots of gene expression values: (a) all genes are plotted with the magenta curve representing the mean over all expression values. (b) mean and normalized standard deviation of the genes in (a).

2.2 Data adjustments

Several data adjustment procedures are available and often used prior to statistical analysis of a given data set. It is important to know not only the mathematical background of these procedures, but also the effects such procedures can have on datasets based on biological observations.

2.2.1 Logarithmic transformation

Usually the data is already in logarithmic form as described in 2.1.1. If not all values x_{ij}' in the data matrix can be processed in the following way:

$$y_{ij} = \log_2\left(x_{ij}'\right) \quad \text{with e.g. } x_{ij}' = \frac{C5_{ij}}{C3_{ij}} \tag{2.2}$$

Almost all results from cDNA microarray experiments are fluorescence ratios. That means, that overexpressed genes are represented by a range of $1<x_{ij}<+\infty$, where underexpressed genes are represented by a range of $0\leq x_{ij}<1$. To overcome this discrepancy the data is usually transformed into logarithmic space, where overexpressed genes are assigned to positive values and underexpressed genes are assigned to negative values (Figure 2.4). A second property of this transformation is that the data is represented in a more "natural" way. Let's

consider an experiment where we are looking at gene expression over time, and the results are relative expression levels compared to time 0. Further assume a hypothetic gene is unchanged at time point 1 (x_0=1.0), 2-fold over expressed at time point 2 (x_1=2.0), and 2-fold underexpressed at time point three relative to time 0 (x_2=0.5) (Figure 2.4). In normal space the difference between time point 1 and 2 is d_{01}=+1.0, while between time point 1 and 3 it is d_{12}=-0.5, so that for mathematical operations that use the difference between values, the 2-fold up change is statistically twice as significant as the 2-fold down change. Usually, a 2-fold up and 2-fold down change are preferred to be of the same magnitude, but in an opposite direction. Therefore it is favored to calculating distances in logarithmic space, where y_0=0, y_1=1.0, y_2=-1.0, d_{01}=1.0, and d_{12}=-1.0 [41]. In most cases the *logarithmus dualis* ($\log_2$) is used instead of $\log_{10}$ or *logarithmus naturalis* (ln), because of the better scaling and the more natural understanding of differences in terms of double and half.

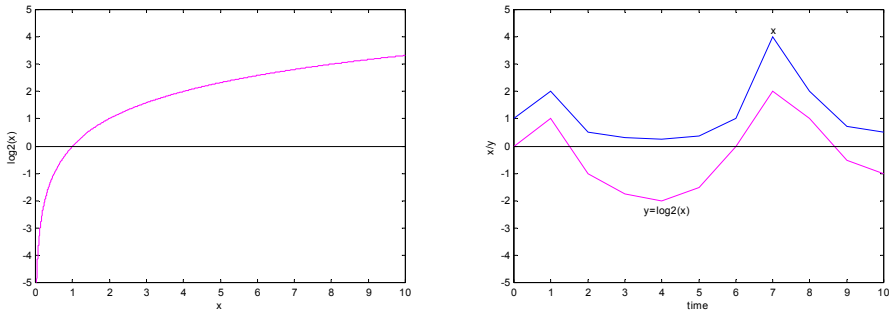


Figure 2.4: (left) log2-transformation. (right) expression curve of a time series in normal (blue) and logarithmic space (magenta).

2.2.2 Mean or median centering

Here the mean or median will be subtracted form each data element:

$$y_{ij} = x_{ij} - \bar{x}_i \qquad \text{or} \qquad y_{ij} = x_{ij} - x_{med,i} \tag{2.3}$$

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \qquad x_{med,i} = \begin{cases} \mathbf{x}'_{i,(n+1)/2} & \text{n odd} \\ \frac{1}{2}\left(\mathbf{x}'_{i,(n/2)} + \mathbf{x}'_{i,(n/2)+1}\right) & \text{n even} \end{cases} \tag{2.4}$$

$\bar{x}_i$ ..... mean of the vector $\mathbf{x}_i$

$x_{med,i}$ ... median of the ascending sorted vector $\mathbf{x}'_i$

which means that the mean or median of a row will be zero (standardization around zero).

Many presently published experiment series consist of a large number of tumor samples all compared to a common reference sample made from a collection of cell-lines. Here each gene is represented by a series of ratio values that are relative to the expression level of that gene in the reference sample. Since the reference sample is usually independent from the experiment, the analysis is preferred to be independent from the gene expression observed in the reference sample, and that is exactly what is achieved by mean and/or median centering. After applying this procedure the values of each gene reflect the variation from some property of the series of observed values such as the mean or median (Figure 2.5).

It makes less sense in experiments where the reference sample is part of the experiment, as it is in many time courses, or if it is important to know whether and how much a gene is over or under expressed and how fare genes are "apart" from each other, since this procedure tends to decrease distances between genes by moving expression patterns from each end of the scale toward the center (Figure 2.5).

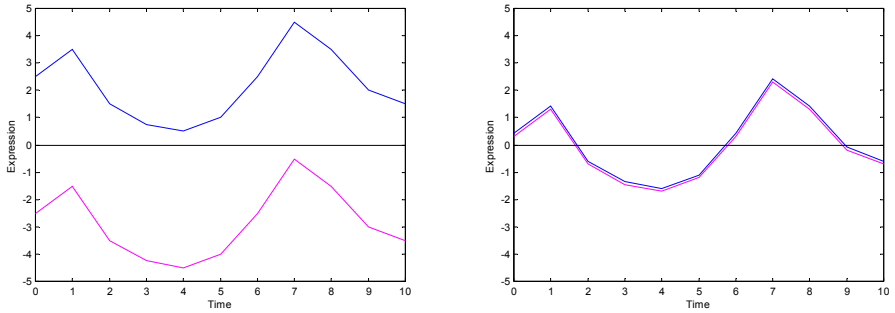


Figure 2.5: (left) Expression patterns of 2 genes, one over expressed, one under expressed. (right) Expression patterns after mean centering: both curves are identical, i.e. the distance between them is 0.

Centering the data for columns/arrays can also be used to remove certain types of biases, which have the effect of multiplying ratios for all genes by a fixed scalar. Mean or median centering the data in log-space has the effect of correcting these biases. However, since in clustering only distances between genes or experiments are used, absolute values play a less important role in the comparison of two genes/experiments. Therefore genes are classified as similar even if the fluorescence ratios are not the same in each experiment due to differences in RNA amounts, labeling efficiency and image acquisition parameters (Figure 2.6).

Nevertheless, it is very important that the fluorescence ratios are normalized within one experiment, since a drift in the intensities within one slide yield to different distances for one and the same gene, depending on the location of the gene on the slide (Figure 2.7).

One way to keep track of differences between experiments is the use of housekeeping genes or external controls. These control genes are genes, which expression levels are invariant in respect of the investigated biological process, i.e. their expression levels should be the same in each experiment. Therefore they can be used for normalization procedures.

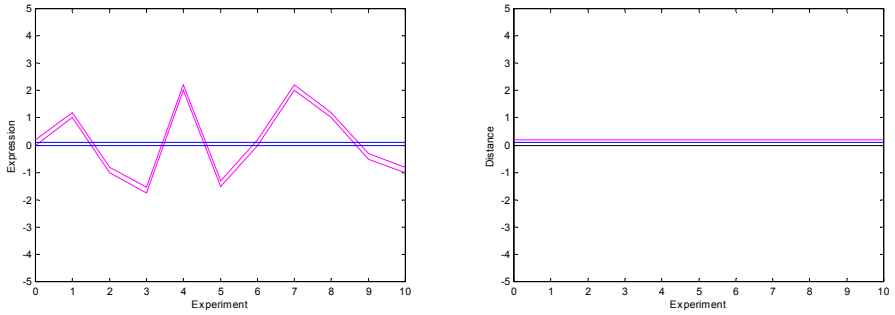


Figure 2.6: (left) 2 equal control genes with constant expression in an ideal (blue) and real (magenta) experiment series. (right) the distances between the two genes are the same, even though the absolute values are different.

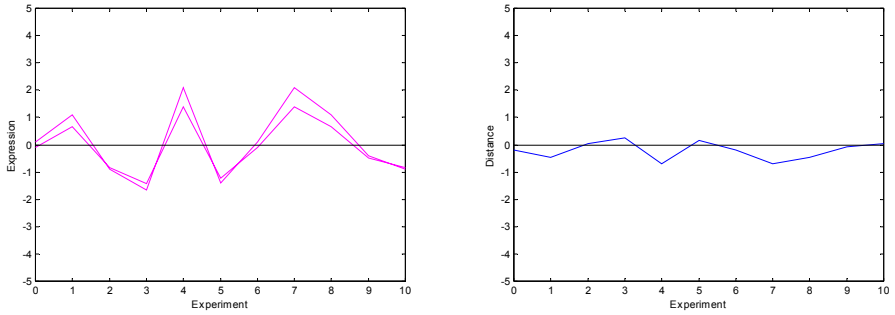


Figure 2.7: (left) two equal control genes with different expression values due to drifts within a slide (background, etc.). (right) the distance between the two control genes is not zero and depends on the drift within the experiment.

2.2.3 Division by RMS/SD

Here, all data elements are divided by a scalar factor, so that the root mean square (RMS) or the standard deviation (SD or σ) of a vector is 1.

$$y_{ij} = \frac{x_{ij}}{RMS_i} \qquad \text{or} \qquad y_{ij} = \frac{x_{ij}}{\sigma_i} \tag{2.5}$$

$$RMS_i = \sqrt{\frac{\sum_{j=1}^{n} x_{ij}^2}{n}} \qquad \sigma_i = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}\left(x_{ij} - \bar{x}_i\right)^2} \tag{2.6}$$

The standard deviation and root mean square characterize the "width" or "variability" of the data. Dividing by the RMS and SD sets these respectively to ones for the genes of interest. This has the effect of amplifying weak signals and suppressing strong signals (Figure 2.8).

However, that also means that if one signal is so weak, that it represents only noise, its amplitude is made comparable to strong signals since all signals are treated equally! Therefore this procedure, like mean/median centering, tends to "create" similarities.

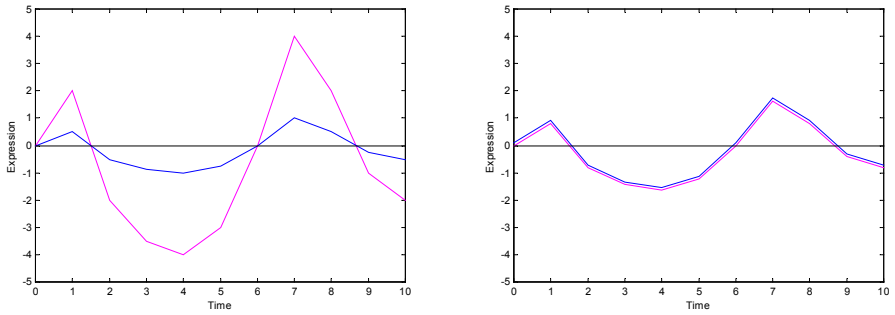


Figure 2.8: (left) two different signals, one representing noise (blue) and one representing a strong signal (magenta). (right) after division by the standard deviation: both signals are treated equally, comparing noise with data.

2.2.4 Discretization of data

Some special distance measuring procedures, such as information entropy, require digital data. This is achieved by executing a discretization on the data set, where the analog data range is divided in equidistant steps (Figure 2.9).

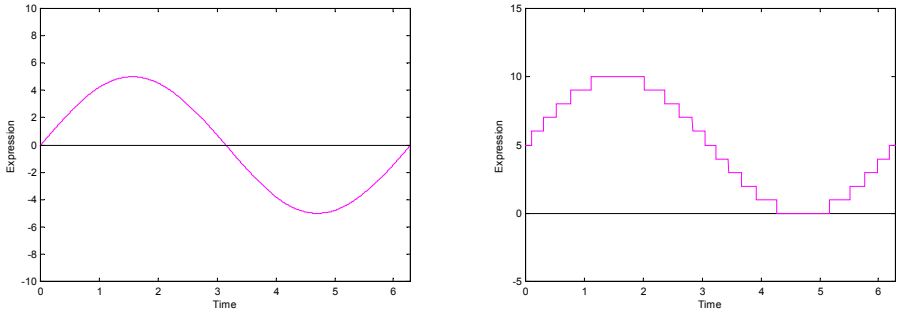


Figure 2.9: (left) analog signal. (right) resulting signal after discretization. Since digital data is nonnegative, the signal is shifted into the positive range.

2.3 Similarity distances

2.3.1 Introduction

All clustering algorithms use a similarity measurement between vectors to compare patterns of expression. Usually an analog value representing the distance between two genes or experiments is computed by summing the distances between their respective vectors. How this value is normalized or how the distance is computed depends on the distance measurement used. There are dozens of algorithms available [42]. In the program described here, the use of the following similarity distance values is available:

- Pearson correlation coefficient
- Uncentered Pearson correlation coefficient
- Squared Pearson correlation coefficient
- Averaged dot product
- Cosine correlation coefficient
- Covariance
- Euclidian distance
- Manhattan distance
- Mutual information
- Spearman Rank-Order correlation
- Kendall's Tau

as well as the absolute value of each distance measurement value. The first 9 are linear correlation coefficients whereas the last 2 are nonparametric or rank correlation coefficients [43].

2.3.2 Pearson correlation coefficient

The linear or Pearson correlation coefficient [43] is the most widely used measurement of association between two vectors. Let **x** and **y** be *n*-component vectors for which we want to calculate the degree of association. For pairs of quantities (x_i,y_i), $i=1,\ldots,n$ the linear correlation coefficient r is given by the formula:

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{2.7}$$

$\bar{x}$ …… mean of the vector **x**

$\bar{y}$ …… mean of the vector **y**

The value r lies between –1 and 1, inclusive, with 1 meaning that the two series are identical, 0 meaning they are completely independent, and -1 meaning they are perfect opposites. The correlation coefficient is invariant under scalar transformation of the data (adding, subtracting or multiplying the vectors with a constant factor).

That means, that using Pearson correlation coefficient has the same effect as using Euclidian distance (see 2.3.13) on standardized (mean centered, unit-variance) data [44]!

2.3.3 Uncentered Pearson correlation coefficient

This is basically the same formula as above, except the mean is expected to be 0.

$$r_{uc} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{2.8}$$

$\bar{x}$ …… mean of the vector **x**

$\bar{y}$ …… mean of the vector **y**

Curves with "identical" shape, but different magnitude have an uncentered correlation coefficient < 1, whereas the standard correlation coefficient in this case would be 1. If just the magnitude of the curve is important, then the uncentered correlation value is better.

2.3.4 Squared Pearson correlation coefficient

The squared Pearson correlation coefficient calculates the square of the Pearson correlation coefficient, so that negative values become positive.

$$r_{sq} = \left(\frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \right)^2 \tag{2.9}$$

$\bar{x}$ …… mean of the vector **x**

$\bar{y}$ …… mean of the vector **y**

If the correlation approaches –1, the squared Pearson will approach 1, i.e. a perfect match. In this way, samples or genes with an opposite behavior will be seen as identical. This is especially good for identifying reciprocal expression profiles (e.g. regulatory or inhibitory mechanisms).

2.3.5 Averaged dot product

The simplest measurement of association between two vectors is the inner product, also referred to as the scalar or the dot product [63]. The inner product between **x** and **y** is defined as the sum of products of components and can be modified by defining an adjusted or "average" value:

$$d = \frac{1}{n}\sum_{i=1}^{n} x_i y_i \tag{2.10}$$

which is independent of the sample size *n*.

2.3.6 Cosine correlation coefficient

When a unit-free measure of association is required, the inner product can be standardized to yield a magnitude-free coefficient, which does not change when the variable is multiplied by a constant. In terms of vector components $\cos\theta$ can be expressed as [63]:

$$\cos\theta = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^{\,2}}\sqrt{\sum_{i=1}^{n} y_i^{\,2}}} \tag{2.11}$$

When $\theta = 0$ both **x** and **y** lie on the same straight line and are thus linearly dependent. For $\theta = 90°$ the vectors are orthogonal, and in general $-1 \le \cos\theta \le 1$. This is similar to the average dot product, except that it has the effect of removing information on the amplitude of the centering.

2.3.7 Covariance

Both the inner product and the cosine correlation coefficient are dependent on the position of the coordinate axes. Since variances and covariance values [63] are defined as second

moments about the mean, however, they are invariant with respect to shifts of the axes effected by addition or subtraction of constants.

$$s = \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n-1} \tag{2.12}$$

$\bar{x}$ …… mean of the vector **X**

$\bar{y}$ …… mean of the vector **Y**

Adding or subtracting constant numbers to the vectors does not alter the magnitude of the covariance, but multiplying by constants does. Covariance values are also independent of the sample size *n*, since the numerator is divided by *n*-1.

2.3.8 Euclidian distance

Euclidian distance [42] is a very commonly used distance measurement. It is basically just the sum of the squared distances of two vector values (x_i,y_i).

$$d_E = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.13}$$

Euclidian distance is variant to both, adding and multiplying all vectors with a constant factor. It is also variant to the dimension of the vector values, for example if missing values reduce the dimension of certain vectors.

2.3.9 Manhattan distance

Similar to the Euclidian distance, the Manhattan distance is the sum of the absolute distances of two vector values (x_i,y_i) on a compound by compound basis.

$$d_M = \sum_{i=1}^{n} |x_i - y_i| \tag{2.14}$$

Basically this is a linear version of the Euclidian distance, with the same advantages and disadvantages.

2.3.10 Spearman Rank-Order correlation

Spearman rank correlation and Kendall's Tau are nonparametric or rank correlations. The key concept of nonparametric correlation is this: If we replace the value of each x_i and y_i by the value of its rank among all the other x_i 's (y_i 's) in the sample, that is, 1, 2, 3, ….. n, then the resulting list of numbers will be drawn from a perfectly known distribution function, namely uniformly from the integers between 1 and n, inclusive. Nonparametric correlation is more robust than linear correlation and more resistant to unplanned defects in the data [42]. For more detailed information see [43].

2.3.11 Kendall's Tau

Kendall's Tau is even more nonparametric than Spearman's Rank correlation. Instead of using the numerical difference of ranks, it uses only the relative ordering of ranks: higher in rank, lower in rank, or the same in rank.

Beware: This is an $O(n^2)$ algorithm. If Kendall's Tau is computed for data sets of more than a few thousand points, some serious computing is required.

For further information on Kendall's Tau the reader is referred to [43].

2.3.12 Mutual Information

Mutual information [45-47], also referred to as rate of transmission, quantifies the reduction in the uncertainty of one random variable given knowledge about another random variable. In order to calculate the information entropy value of the vectors to compare, continuous data is transformed by binning the expression levels into $\log_2(n)$ discrete, equidistant levels. The information entropy H, of each gene expression series **x**, is calculated from the probabilities, $P(b_{xi})$, of the occurrence of one of the $\log_2(n)$ expression levels over the n measurement points:

$$H(\mathbf{x}) = -\sum_{b_{xi}} P(b_{xi}) \cdot \log_2\left(P(b_{xi})\right) \tag{2.15}$$

$$H(\mathbf{y}) = -\sum_{b_{yi}} P(b_{yi}) \cdot \log_2\left(P(b_{yi})\right) \tag{2.16}$$

$H(\mathbf{x}), H(\mathbf{y})$ …… information entropy of vector **x, y**

$P(b_{xi}), P(b_{yi})$…… probability of bin i of vector **x, y**

The joint entropy for each gene expression pair, H(**x**,**y**), is calculated analogously:

$$H(\mathbf{x},\mathbf{y}) = -\sum_{b_{xi},b_{yi}} P(b_{xi},b_{yi}) \cdot \log_2\left(P(b_{xi},b_{yi})\right) \tag{2.17}$$

The mutual information of the two vectors **x** and **y** is determined according to the following definition:

$$M(\mathbf{x},\mathbf{y}) = H(\mathbf{x}) + H(\mathbf{y}) - H(\mathbf{x},\mathbf{y}) \tag{2.18}$$

Since the degree of mutual information is dependent on how much entropy is carried by each gene expression vector, two gene expression vectors exhibiting low information entropies will also have low *M*, even if they are completely correlated. Therefore, *M* is normalized to the maximum entropy of each of the contributing vectors, giving a high value for highly correlated sequences, independent of the individual entropies.

$$M(\mathbf{x},\mathbf{y})_{norm} = \frac{M(\mathbf{x},\mathbf{y})}{\mathbf{max}[H(\mathbf{x}),H(\mathbf{y})]} \tag{2.19}$$

1-M_{norm} is used as distance measure for the distance measurement, since maximum coherence must correspond to minimum distance.

$$D(\mathbf{x},\mathbf{y}) = 1 - M(\mathbf{x},\mathbf{y})_{norm} \tag{2.20}$$

2.3.13 Comparison

For a better understanding of the effects of the various distance measurements, a Principal Component Analysis (PCA, see 2.8) of a sporulation dataset using all the methods described above has been calculated. The results are shown in Figure 2.10 and 2.11. PCA has no random part in its calculation therefore results are reproducible and comparable. For that reason the results vary only because of the different distance measuring methods used. It is important to know that the principal components plotted in Figure 2.10 and 2.11 are Eigenvectors, so they can be multiplied with any real constant, including –1, which means a reflection of the PC curve in respect of the x-axis. It can easily be seen that different distance functions lead to different, but similar principal components. For that reason it is important to choose the best fitting distance measurement by answering the question, what similar should mean in respect of the used data, considering the properties of the different procedures described above. Which genes should be considered similar, which not.

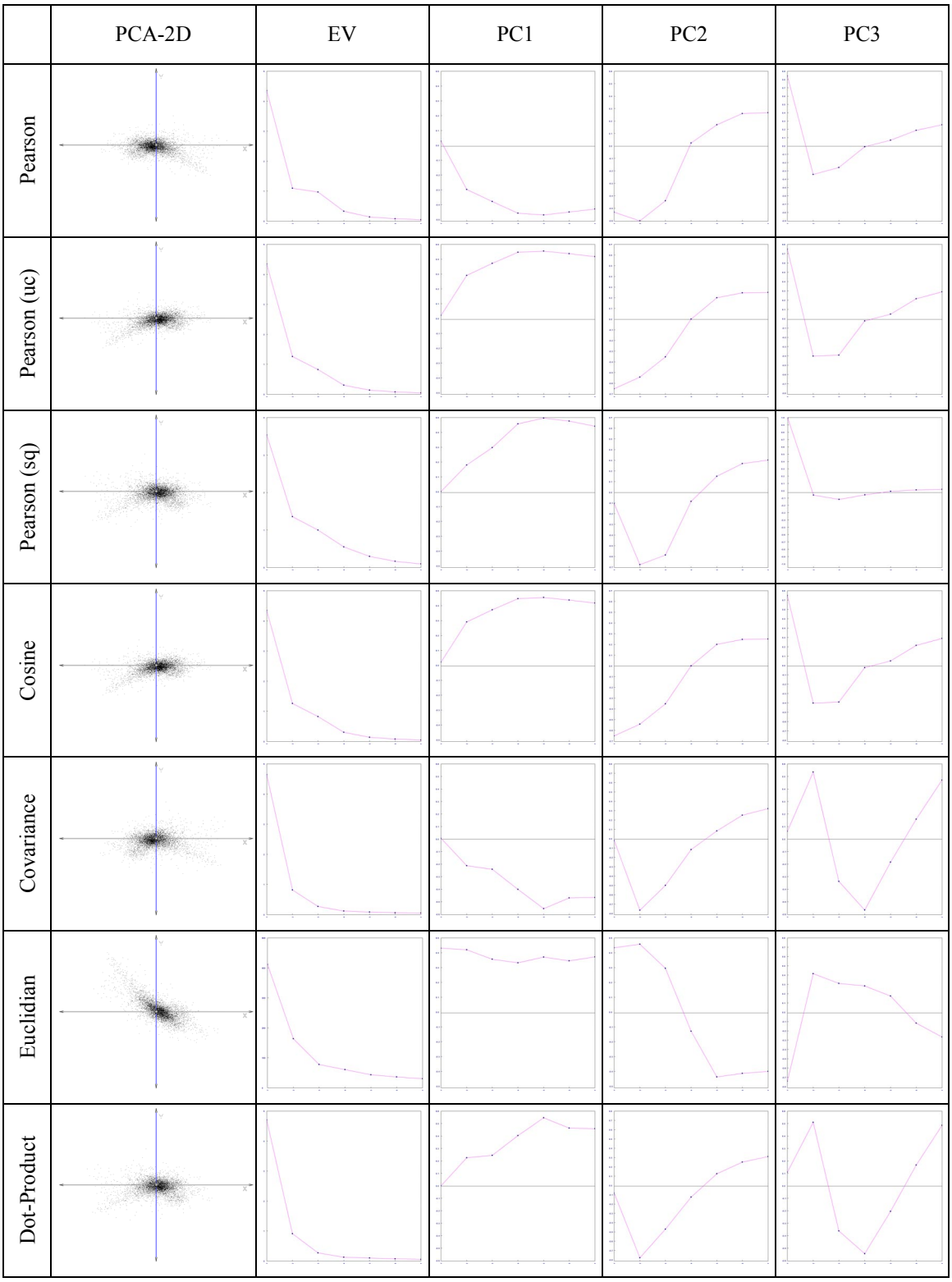


Figure 2.10: Comparison of all mentioned distance measurement procedures using Principal Component Analysis of a yeast sporulation dataset. Plotted are: 2D-view of Principal Component 1 and 2, the Eigenvalue distribution and the first 3 Principal Components.

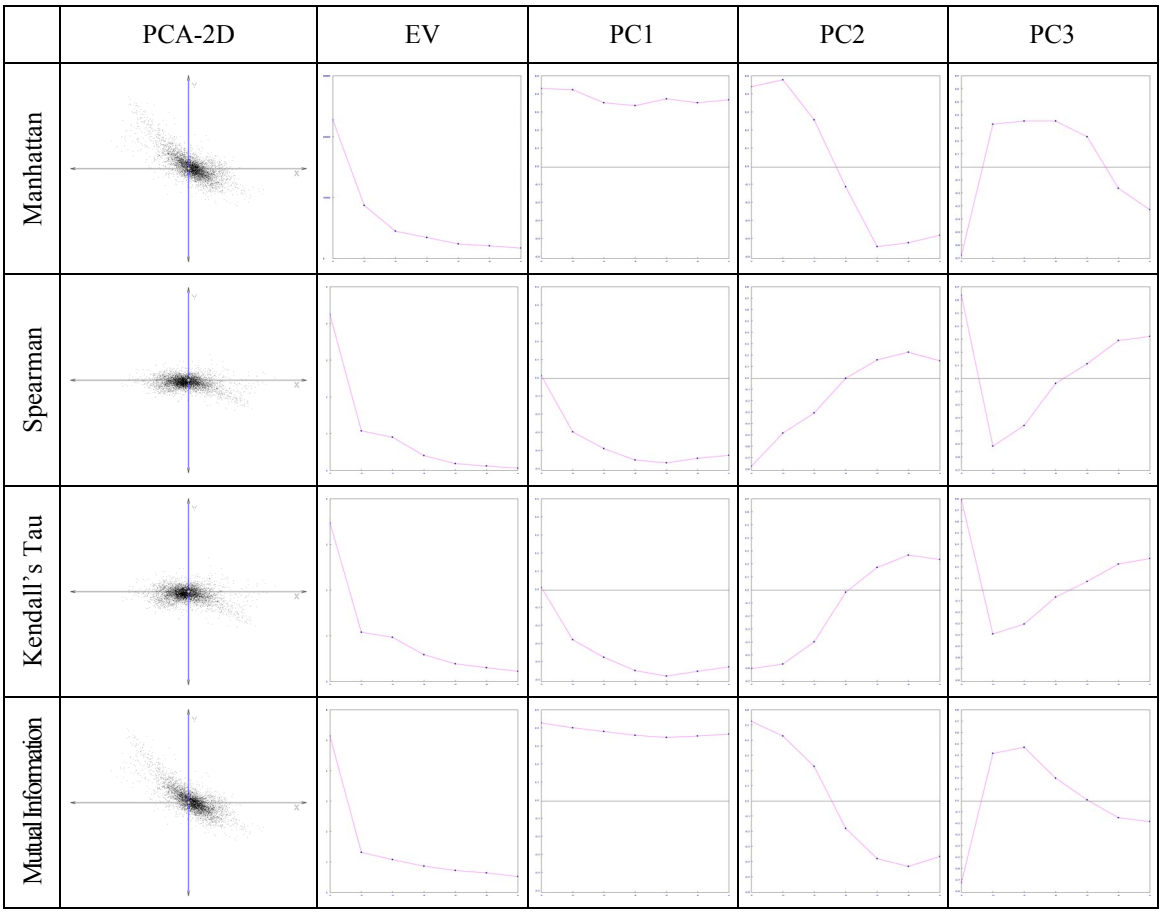


Figure 2.11: Comparison of all mentioned distance measurement procedures using Principal Component Analysis of a yeast sporulation dataset. Plotted are: 2D-view of Principal Component 1 and 2, the Eigenvalue distribution and the first 3 Principal Components.

2.3.14 Missing values

Due to various effects during spotting, hybridization, and data analysis, not each spot can be assigned a meaningful ratio. This results in missing values in the data matrix. To calculate distances, only elements represented in both vectors are used. If there is a missing value in one or both vectors, this dimension is not included in the distance calculation. This can lead to various problems:

The greatest problems occur, if the distance is not independent of the number of vector elements n, as it is the case for Euclidian distance for instance. Vectors with missing values are then differently weighted in comparison with vectors with no missing values. Let's say there are 3 genes:

1. A gene-vector with all values valid
2. A gene-vector with all values present but not equal to 1.
3. A gene-vector with only one value equal to the corresponding value of gene 1

If vector 1 & 2 and 1 & 3 are compared, the following results are obtained:

- 1 & 2: They are not similar so the distance is greater than 0
- 1 & 3: Only values, which are present in both vectors, are used for the distance calculation, so the genes are treated similarly, because the only comparison results in distance 0. But vector 1 and 3 could also be completely different...

For that reasons, missing values are difficult to handle. A few are usually no problem, but if there are too many in comparison to the number of vector-elements *n*, an arbitrary result can be expected.

2.3.15 Underlying assumptions

To calculate distances as described above, two major assumptions have to be fulfilled to get meaningful information:

Any distance measurement provides a quantification of the **linear** relation between **two** random variables (measurements), **x** and **y** [44,48].

First, it is assumed that **y** is only related to a single variable **x**. In all but very simple systems, **y** depends not only on one, but on many input variables. For these systems, calculations of relations using distance measurements as described here are insufficient. However, restriction to these simple tests drastically reduces the calculations required to find relationships between expression patterns. The price of this speed is a lack of consideration of the ability of nature to make decisions based on multiple inputs.

A second problem is the linearity assumption. The best linear calculated relation might be poor, whereas there may be good nonlinear relations. Finding nonlinear relationships in large amounts of data is not trivial and will require years of research in the mathematical background of gene expression clustering.

However, the results obtained by clustering methods using these distance measurements demonstrate that such procedures are a useful way to extract and visualize one-to-one correlations from large datasets [48].

2.4 Clustering introduction

There are two straightforward ways how gene expression matrices can be studied [19]:

- Comparing expression profiles of genes by comparing rows in the expression matrix and
- Comparing expression profiles of experiments by comparing columns in the matrix.

Additionally, if the data normalization allows it, a combination of both are possible. We can look either on similarities or differences. If two genes (rows) are similarly expressed, we can hypothesize that the respective genes are co-regulated and possibly functionally related (Guild By Association) [49]. The comparison of experiments can provide us with the information, which genes are differentially expressed in two conditions and this enables us to study, for example, effects various compounds have on an investigated condition.

Clustering can be defined as the process of separating a set of objects into several subsets on the basis of their similarity [50]. The aim is generally to define clusters that minimize intracluster variability while maximizing intercluster distances, i.e. finding clusters, which members are similar to each other, but distant to members of other clusters in terms of gene expression based on the used similarity measurement. Two clustering strategies are possible: supervised (based on existing knowledge) or unsupervised.

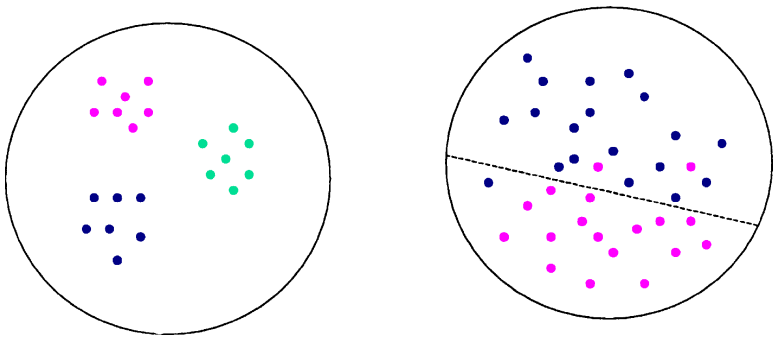


Figure 2.12: Supervised and unsupervised data analysis. In the unsupervised case (left) we are given data points in n-dimensional space (n=2 in the example) and we are trying to find ways how to group together points with similar features. For instance, there are three natural clusters in the example, each consisting of data points close to each other in a sense of Euclidean distance. A clustering algorithm should identify all these clusters. In the supervised case (right), the objects are labeled (e.g. we have magenta and blue points in the example), and the task is to find a set of classification rules allowing us to discriminate between these points as precisely as possible. For instance, the dotted line in the drawing [19].

2.5 Hierarchical Clustering

2.5.1 Introduction

Hierarchical clustering [21,51-53] is an unsupervised procedure of transforming a distance matrix, which is a result of pair wise similarity measurement between elements of a group, into a hierarchy of nested partitions. The hierarchy can be represented with a tree-like dendrogram in which each cluster is nested into the next cluster. Hierarchical algorithms can be further categorized into two kinds:

(1) Agglomerative procedures: This procedure starts with n clusters (each object forms a cluster containing only itself) and iteratively reduces the number of clusters by merging the two most similar objects or clusters, respectively, until only one cluster is remaining. ($n \rightarrow 1$).

(2) Divisive procedures: This procedure starts with 1 cluster and iteratively splits a cluster, so that the heterogeneity is reduced as far as possible ($1 \rightarrow n$).

If it is possible to find a reasonable distance definition between clusters, agglomerative procedures are less computationally expensive than divisive procedures, since in one step two out of maximum n elements have to be chosen for merging, whereas in divisive procedures, fundamentally all subsets have to be analyzed so that divisive procedures have an algorithmic complexity in the magnitude of $O(2^n)$. Agglomerative procedures have the drawback, that an incorrect merging of clusters in an early stage often yields results, which are far away from the real cluster structure. Divisive procedures immediately start with interesting cluster arrangements and are therefore much more robust. Usually agglomerative procedures are used because of their efficiency.

2.5.2 Algorithm

The procedures of agglomerative hierarchical clustering execute the following basic steps:

(1) Calculate the distance between all objects and construct the similarity distance matrix. Each object represents one cluster, containing only itself.
(2) Find the two clusters r and s with the minimum distance to each other.

(3) Merge the clusters *r* and *s* and replace *r* with the new cluster. Delete *s* and recalculate all distances, which have been affected by the merge.

(4) Repeat step (2) and (3) until the total number of clusters become one.

2.5.3 Amalgamation or linkage rules

At the first step, when each object represents its own cluster, the distances between those objects are defined by the chosen distance measure. However, once several objects have been linked together, a linkage or amalgamation rule is needed to determine if two clusters are sufficiently similar to be linked together. There are numerous linkage rules that have been proposed. Here are some of the most commonly used:

Single linkage (nearest neighbor)

In this method the distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters. If there are several equal minimum distances between clusters during merging, single linkage is the only well defined procedure. Its greatest drawback is the tendency for chain building: Only one (random) small distance is enough to enforce the amalgamation of two otherwise very different clusters. Therefore, the resulting clusters tend to represent long "chains" (Figure 2.13a).

Complete linkage (furthest neighbor)

In this method, the distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbors"). Complete linkage usually performs quite well in cases when the objects actually form naturally distinct data clouds in the multidimensional space. If the clusters tend to be somehow elongated or of a "chain" type nature, then this method is inappropriate. Since only one (random) large distance is enough to pretend two clusters from merging, clusters tend to be small and merged together very late with a great error value.

Unweighted pair-group average linkage

In this method, the distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters. This method is very efficient when the objects form natural distinct "clumps," however, it performs equally well with elongated, "chain" type clusters. Since the distance between two clusters lies between the minimum formation of single linkage and the maximum formation of complete linkage this procedure

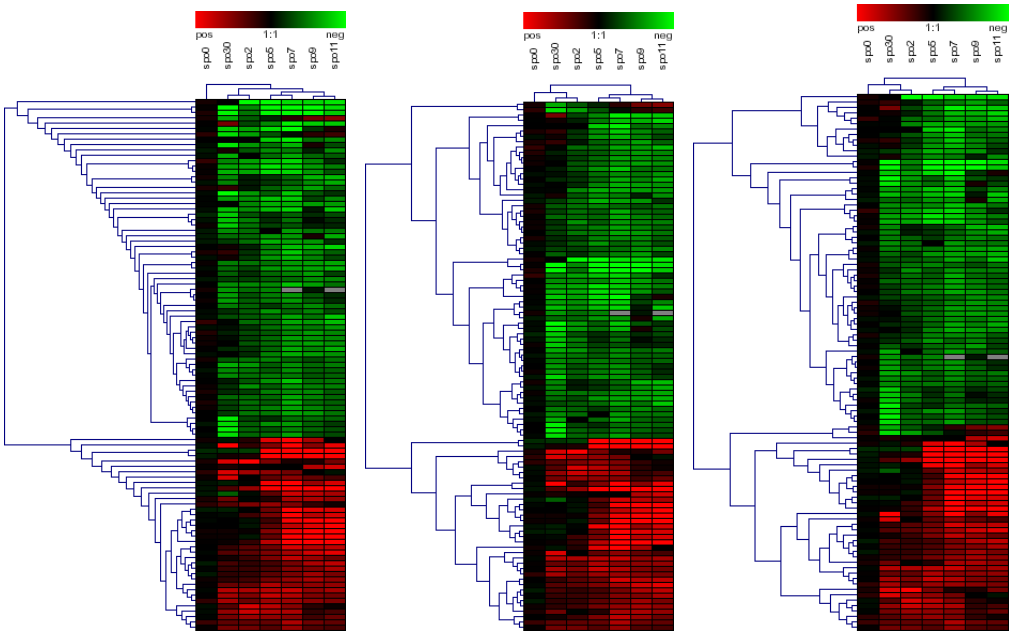
empirically shows no tendencies to either extreme described above, and is therefore more stable to unknown data point distributions. Admittedly, if there are several equal distances, the sequence of the amalgamation is critical. Note that the abbreviation UPGMA is used as well to refer to this method as *unweighted pair-group method using arithmetic averages*.

Weighted pair-group average linkage

This method is identical to the unweighted pair-group average method, except that in the computations, the size of the respective clusters (i.e., the number of objects contained in them) is used as a weight. Thus, this method (rather than the previous method) should be used when the cluster sizes are suspected to be greatly uneven.

Unweighted pair-group centroid linkage

The *centroid* of a cluster is the average point in the multidimensional space defined by the dimensions. In a sense, it is the *center of gravity* for the respective cluster. In this method, the distance between two clusters is determined as the difference between centroids.



(a) (b) (c)

Figure 2.13: Dendrogram representations of Hierarchical Clustering using the amalgamation rules of: (a) single linkage (b) complete linkage (c) unweighted average linkage clustering.

Weighted pair-group centroid linkage

This method is identical to the previous one, except that weighting is introduced into computation to take into consideration differences in cluster sizes.

Ward's method

This method is distinct from all other methods because it uses an analysis of a variance approach to evaluate the distances between clusters. This method attempts to minimize the Sum of Squares of any two (hypothetical) clusters that can be formed at each step. In general, this method is regarded as very efficient and tends to create equally sized clusters of small size.

2.5.4 Properties

Hierarchical clustering is the most commonly used clustering strategy for gene expression analysis at the moment. The biggest advantage is that aside from a choice of the amalgamation rule and the type of similarity distance measurement, no further parameters have to be specified. The result is a reordered set of genes and/or experiments, where similar vectors are close to each other in the tree structure and the distance between vectors and clusters is encoded in the branch length of a subtree. This not only allows estimation of the similarity of neighboring genes, but also of the distance between distant vectors. This is helpful if someone is more interested in distances rather than similarities between two or more investigated conditions.

Hierarchical clustering just rearranges the dataset to a new, better ordered set of data vectors, therefore clusters have to be specified by the user by selecting a subtree as a cluster. A second drawback is the computational complexity. Large datasets are difficult or impossible to calculate due to the vast amount of necessary memory for the similarity matrix and the calculation time needed. Datasets with more than 20.000 vectors are manageable just by very advanced computer hardware.

The software discussed in this paper is able to calculate Single Linkage, Complete Linkage and Unweighted Average Linkage Clustering on both the genes and the experiments.

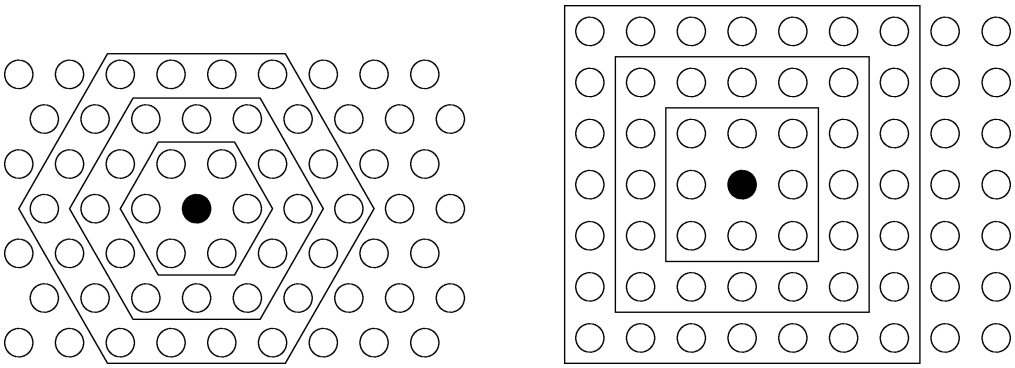
2.6 Self Organizing Maps

2.6.1 Introduction

One of the most popular neural network models today is the principle of a Self-Organizing Map (SOM) [54-58], developed by professor Kohonen at the University of Helsinki. A SOM is basically a multidimensional scaling method, which projects data from input space to a lower dimensional output space. The SOM algorithm is based on unsupervised competitive learning, which means that the training is entirely data-driven and needs no further information.

A SOM is formed of neurons located in a regular, usually 1- or 2-dimensional grid. Each neuron i of the SOM is represented by an n-dimensional weight or reference vector $\mathbf{m}_i=[m_{i1},\ldots,m_{in}]^T$, where n is equal to the dimension of the input vectors.

The neurons of the map are connected to adjacent neurons by a neighborhood relation dictating the structure of the map. Usually the map topology is rectangle or hexagonal (Figure 2.14). The number of neurons determines the granularity of the resulting mapping, which affects the accuracy and the generalization capability of the SOM.



(a) Hexagonal grid (b) Rectangular grid

Figure 2.14: In the 2-dimensional case the neurons of the map can be arranged either on a rectangular or hexagonal lattice. Neighborhoods (size 1, 2 and 3) of the unit marked with black dot.

2.6.2 Initialization

In the basic SOM algorithm, the topological relations and the number of neurons are fixed from the beginning. The number of neurons should be the number of clusters expected, with the neighborhood size controlling the smoothness and generalization of the mapping.

Before the training phase, initial values are given to a weight vector, defined for each neuron The SOM is robust regarding the initialization, but properly accomplished, it allows the algorithm to converge faster to a better solution. The two following initialization procedures are used:

- Random initialization, where the weight vectors are initialized with small random values between the minimum and maximum values of the vector.
- Random Gene initialization, where the weight vectors are initialized with random sample vectors from the training dataset.

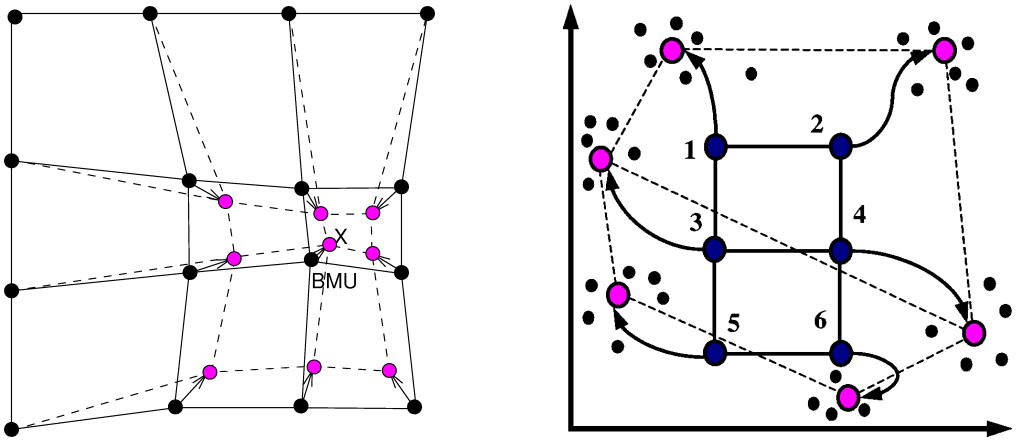
2.6.3 Training

In each training step, one sample vector **x** from the input data set is chosen randomly and a similarity measurement is calculated between it and all the weight vectors of the map. The Best-Matching unit (BMU), denoted as *c*, is the unit whose weight vector has the greatest similarity with the input sample **x**. The similarity is usually defined by means of a distance measurement, typically Euclidian distance. Formally the BMU $\mathbf{m}_c$ is defined as the neuron for which

$$\| \mathbf{x} - \mathbf{m}_c \| = \min_i \{ \| \mathbf{x} - \mathbf{m}_i \| \}, \tag{2.21}$$

where $\| . \|$ is the distance measure.

After finding the BMU, the weight vectors of the SOM are updated. The weight vectors of the BMU and its topological neighbors are moved closer to the input vector in the input space. This adaptation procedure stretches the BMU and its topological neighbors towards the sample vector (Figure 2.15).



(a) (b)

Figure 2.15: (a) Update of the best matching unit (BMU) and its neighbors towards the input sample marked with x. The solid and dashed lines correspond to the situation before and after updating, respectively. The lines show the neighborhood relations. (b) Adaptation of the SOM to the input space after several iterations.

The SOM update rule for the weight vector of the unit *i* is:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)], \tag{2.22}$$

where t denotes time. The $\mathbf{x}(t)$ is the input vector randomly drawn from the input data set at time t and $h_{ci}(t)$ is the neighborhood kernel around the winner unit c at time t.

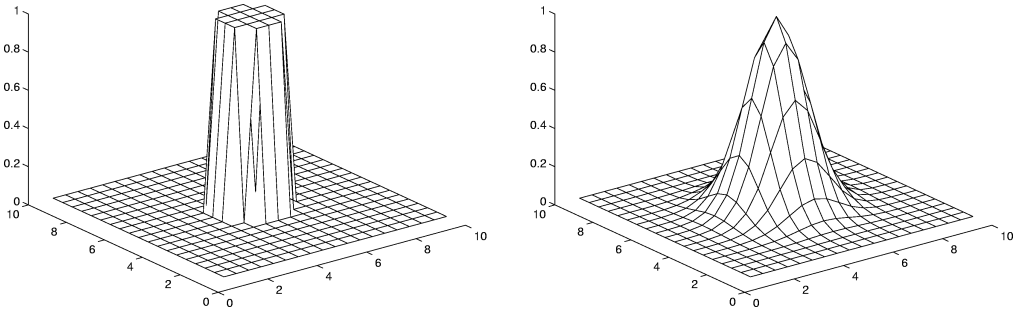
Only the weight vectors in a specific area around the BMU are updated. The neighborhood function specifies the size of the area and therefore the number of vectors that are updated in a training step. It defines the region of influence that the input sample has on the SOM. The neighborhood kernel is a non-increasing function of time and distance of unit i from the winner unit c. The kernel is formed of two parts: the neighborhood function $h(d, t)$ and the learning function $\alpha(t)$:

$$h_{ci}(t) = \alpha(t) \cdot h(\| \mathbf{r}_c - \mathbf{r}_i \| , t) \tag{2.23}$$

where $\mathbf{r}_i$ is the location of unit i on the map grid.

The simplest neighborhood function is the bubble: It is constant over the whole neighborhood of the winner unit and zero elsewhere. The second is the Gaussian neighborhood function (Figure 2.16).

The latter gives slightly better results but is computationally somewhat heavier. The neighborhood radius starts at its initial value and is decreasing linearly to one, during the training.



(a) (b)

Figure 2.16: The two basic neighborhood functions: (a) Bubble neighborhood. (b) Gaussian neighborhood

The learning rate $\alpha(t)$ is a decreasing function of time. Usually a linear function is used.

During training, the SOM behaves like a flexible net that folds onto the point clouds formed by the training data (Figure 2.15b).

The training is terminated after a predefined number of steps has been performed.

An important property of the SOM is, that it can be used even on partial data, or data with missing data component values. If the sample input vector on a specific training step has missing components, they are simply left out from the distance calculation and updating procedure. The algorithm remains statistically valid, until the number of missing components in the vector is too big.

2.6.4 Clustering

The SOM partitions the input space into convex Voronoi regions, each of which corresponds to one unit of the map. The Voronoi region of a map unit i is defined as the union of all input vectors $\mathbf{x}$ to which it is the closest:

$$V_i = \{ \mathbf{x} \mid \|\mathbf{m}_i - \mathbf{x}\| < \|\mathbf{m}_j - \mathbf{x}\|, i \neq j \}. \quad (2.24)$$

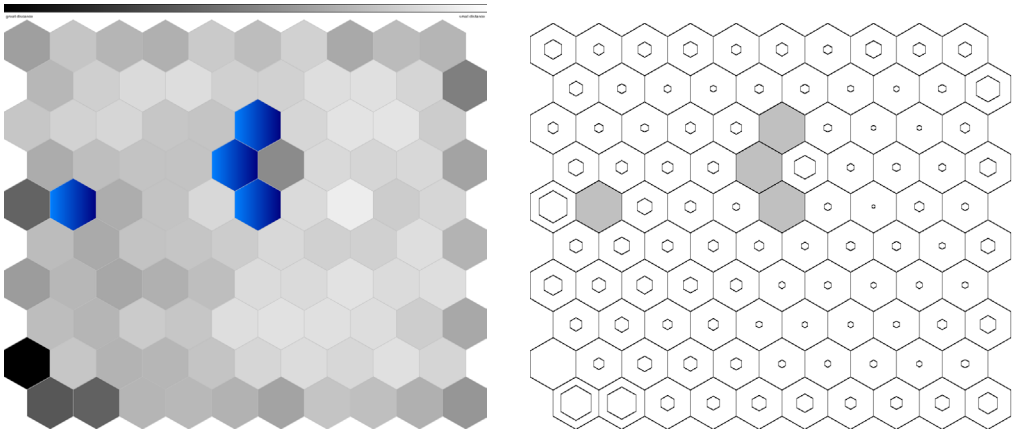
That means that the input vectors can be clustered according to their Voronoi regions. Each Voronoi region is one cluster consisting of very similar vectors. It also denotes, that the number of clusters has to be estimated prior to calculation, because it depends only on the number of nodes in the SOM. This is one of the major drawbacks of the SOM.

2.6.5 Visualization

Clusters are groups of vectors, which are close to each other compared to their distance to all the other vectors. Therefore, the clustering structure of a SOM can be visualized by displaying distances between vectors. The most common method is the unified distance matrix (u-matrix).

In the u-matrix method, a matrix of distances between the weight vectors of a map unit and their neighbors is calculated. By showing this matrix as a gray-level image, the relative distances between adjacent map units on the whole map can be seen. This representation gives information about the distances between neighboring clusters in the SOM.

The method of Kraaijveld uses only one value per map unit: In our case the maximum of distances between the unit (SOM vector) and its cluster vectors (Figure 2.17). The distances are normalized in respect of the maximum distance in all clusters. This representation gives information about the integrity or "size" of a cluster. In the program described here only this representation is calculated.



(a) (b)

Figure 2.17: Kraaijveld representation of a 10x10 SOM. (a) The maximum distance between a unit and its cluster vectors is coded in gray-levels. The greatest cluster size is represented by black, the smallest by white fields. If a cluster has no members it is colored blue. (b) Same SOM as in (a) but here the distance is represented in respect of the greatest cluster (left, lower corner). Gray fields represent empty clusters.

2.6.6 Properties

Some of the properties have already been discussed above. It is a robust clustering procedure, and the results can easily be visualized for a better understanding of how "good" a cluster is.

One of the biggest advantages of the SOM algorithm are the moderate memory and time consumption, although distances must be calculated at each step, since no similarity distance matrix is needed, the memory requirement rises only with O(n). The calculation time is determined by the net size and the number of iterations until the algorithm is terminated. Usually the network fits quickly to the data distribution in the multidimensional space and only a few seconds of calculation are required for a good result. It is also possible to calculate a rough result first to test some parameter effects, and calculate the final result after the best parameters have been found. Important with this approach is, not to forget the random part of the algorithm; a certain amount of calculation is required to get meaningful, comparable results. However, since there is no specific termination criterion, it is led to the user to decide how many iterations have to be calculated.

This indicates maybe the most significant drawback of the SOM algorithm: It requires the specification of many parameters:

- The dimension in X and Y of the map (number of clusters)
- The number of iterations
- The initial learning rate α
- The neighborhood radius
- The type of neighborhood function
- The type of vector initialization prior to training
- The topology of the map

It is sometimes difficult or impractical to test all possibilities, however, usually the default values (see 3.3) yield to good results and changing parameters, like learning rate and neighborhood radius, do not change the result significantly. Important are the net dimensions (number of clusters) and the number of iterations.

SOM also possesses the ability to create fuzzy clusters, where one vector can belong to more than one cluster, but this requires one more input from the user, namely the radius of the area around each SOM-vector that belongs to it. The implementation of the SOM-algorithm discussed in this paper does not have the feature of determining fuzzy clusters yet.

2.7 k-means clustering

2.7.1 Introduction

k-means [59-62] is a commonly used clustering method because it is based on a very simple principle and provides good results. It is very similar to SOM, unsupervised, and can be seen as a Bayesian (maximum likelihood) approach to clustering.

The basic idea is to maintain two estimates:

(1) An estimate of the center location for each cluster and
(2) A separate estimate of the partition of the data points according to which one goes into which cluster.

One estimate can be used to refine the other. If we have an estimate of the center locations, then (with reasonable prior assumptions) the maximum likelihood solution is that each data point should belong to the cluster with the nearest center. Hence, we can compute a new partition from a set of center locations, i.e. make one cluster from the set of vectors in each Voronoi cell.

For this reason, the k-means algorithm proceeds by a sequence of phases in which it alternates between moving data points to the cluster of the nearest center, and moving all cluster centers to the mean of their Voronoi sets.

2.7.2 Clustering

Let $\mathbf{X}=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ denote a set of n vectors and $\mathbf{x}_i=\{x_{i1}, x_{i2}, \ldots., x_{im}\}$ be a vector represented by m attribute values. Let k be a positive integer. The objective of clustering $\mathbf{X}$ is to find a partition, which divides the objects in $\mathbf{X}$ into k disjoint clusters.

For a given n, the number of possible partitions is definite but extremely large. It is impractical to investigate every partition in order to find the optimal one. A common solution to this problem is to choose a clustering criteria or cost function to guide the search for a better partition.

Cost function

One widely used cost function is the trace of the within cluster dispersion matrix, which can be defined as

$$E = \sum_{j=1}^{k} \sum_{i=1}^{n} y_{ij} d\left(\mathbf{x}_i, \mathbf{q}_j\right) \tag{2.25}$$

Here, $\mathbf{q}_j=[q_{j1}, q_{j2}, \ldots, q_{jk}]$ is the representative vector or prototype for cluster *i*. In k-means the prototype $\mathbf{q}_j$ is the mean of all vectors $\mathbf{x}_i$ in a specific cluster. $d(\mathbf{x}_i,\mathbf{q}_j)$ is a similarity measure between $\mathbf{q}_j$ and $\mathbf{x}_i$ and finally y_{ij} is an element of a partition Matrix **Y**.

Y has the following two properties:

(1) $0 \le y_{ij} \le 1$ and (2.26)

(2) $\sum_{1}^{k} y_{ij} = 1$ (2.27)

Y is called a hard partition if $y_{ij} \in \{0,1\}$. Otherwise it is a fuzzy partition. In a hard partition, $y_{ij}=1$ indicates that object $\mathbf{x}_i$ is assigned to cluster *j* by **Y**. In this paper only hard partitions are considered.

The inner term $\sum_{i=1}^{n} y_{ij} d\left(\mathbf{x}_i, \mathbf{q}_j\right)$ in Eq. (2.25) is the total cost of assigning $\mathbf{x}_i$ to cluster *j*, i.e. the total dispersion of objects in cluster *j* from prototype $\mathbf{q}_j$.

Algorithm

The essence of the k-means clustering algorithm is now to minimize the cost function of all clusters by executing the following steps:

(1) Put each vector $\mathbf{x}_i$ of **X** in one of the *k* clusters.
(2) Calculate the mean for each of the *k* clusters.

(3) Calculate the distance between an object and the mean of a cluster.
(4) Allocate an object to the cluster whose mean is the nearest to the object.
(5) Re-calculate the mean of the clusters affected by the reallocation.

(6) Repeatedly perform the operations (3) to (5) until no more reallocations occur.

2.7.3 Properties

The k-means algorithm has the following important properties:

It is efficient in processing large data sets, due to the fact that the computational complexity of the algorithm is *O(tkmn)*, where *n* is the number of vectors in **X**, *m* is the dimension of the vectors $\mathbf{x}_i$, *k* is the number of clusters and *t* is the number of iterations over the whole data set. Usually, $k, m, t << n$. It takes usually just a few seconds to calculate even datasets with 10000 elements and more, making it a valuable tool for the investigation of datasets that are too big for hierarchical clustering for instance.

Another big advantage is the moderate memory requirement for k-means clustering. Since there is no similarity matrix to calculate the memory requirements rise with *O(n)*.

The clusters have convex shapes. Therefore, it is difficult to use the k-means algorithm to discover clusters with non-convex shapes

It often terminates at a local optimum. To find the global optimum, techniques such as deterministic annealing and generic algorithms can be incorporated with the k-means algorithm.

The major drawback of the k-means algorithm is that the number of clusters has to be specified in advance. However, this is the only input needed for the clustering besides an abortion criterion to prevent infinite calculation, like an input for the maximum number of iterations to compute.

Some variants like the ISODATA algorithm include a procedure to search for the best *k* at the cost of performance and therefore overcome the problem of estimating *k*. However, in this paper we concentrate on the standard k-means implementation.

An additional advantage of k-means is the possibility to create fuzzy clusters, where one vector can belong to more than one cluster. This is a better model for the regulatory system that controls gene expression in a cell. One gene affects more than one other gene, i.e. it can be part of many different pathways and therefore has to belong to different clusters or no one.

2.8 Principal Component Analysis

2.8.1 Introduction

Principal Component Analysis (PCA), also known as Singular Value Decomposition (SVD) [63-65] is an exploratory multivariate statistical technique that allows the identification of key variables (or combinations of variables) in a multidimensional data set that best explain the differences between observations. Given *m* observations (experiments) on *n* variables (genes), the goal of PCA is to reduce the dimensionality of the data matrix by finding $r \leq n$ new variables. These *r* principal components account together for as much of the variance in the original *n* variables as possible while remaining mutually uncorrelated and orthogonal. The goal is to reduce dimensionality while filtering noise in the process, making the data more accessible for visualization and analysis.

2.8.2 Mathematical background

Consider *m* observations on *n* random variables represented by the matrix **X**. **D** is a distance matrix of the input matrix **X**. Let **P** denote a *(m x m)* matrix of unknown coefficients such that the quadratic form $\mathbf{P}^T\mathbf{DP}$ is maximized subject to the constraint $\mathbf{P}^T\mathbf{P} = \mathbf{I}$. This is equivalent to maximizing the Lagrangean expression

$$\mathbf{\Phi} = \mathbf{P}^T\mathbf{DP} - \lambda\mathbf{I}(\mathbf{P}^T\mathbf{P} - \mathbf{I}) \tag{2.28}$$

where $\lambda\mathbf{I}$ is a diagonal matrix of Lagrange multipliers (Eigenvalues). Differentiating with respect to **P** and setting the equation to zero we are receiving

$$\frac{\partial \Phi}{\partial \mathrm{P}} = 2\mathrm{D} - 2\lambda\mathrm{P} = 0 \tag{2.29}$$

or

$$(\mathrm{D}-\lambda\mathrm{I})\mathrm{P}=0 \tag{2.30}$$

The normal equations in (2.30) yield estimates for Eigenvalues and Eigenvectors. To compute the principal components, the *m* Eigenvalues and their corresponding Eigenvectors are calculated from the *(m x m)* distance matrix **D** using for example Singular Value Decomposition (SVD). When **D** is nonsingular, all latent roots are strictly positive and each Eigenvector defines a principal component.

SVD methods are based on the following theorem of linear algebra: Any *(n x m)* matrix **A** whose number of rows *n* is greater than or equal to its number of columns *m*, can be written as the product of a *(n x m)* column-orthogonal matrix **U**, a *(m x m)* diagonal matrix **W** with positive or zero elements (the singular values), and the transpose of an *(m x m)* orthogonal matrix **V**.

$$\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^{T} \tag{2.31}$$

SVD now explicitly constructs orthonormal bases for the nullspace and range of a matrix. Specifically, the columns of **U** whose same-numbered elements w_j are nonzero are an orthonormal set of basis vectors that span the range; the columns of **V** whose same-numbered elements w_j are zero are an orthonormal basis for the nullspace.

The matrices **U** and **V** are each orthogonal in the sense that their columns are orthonormal.

$$\mathbf{U}^{T}\mathbf{U} = \mathbf{V}^{T}\mathbf{V} = \mathbf{V}\mathbf{V}^{T} = \mathbf{I} \tag{2.32}$$

The vectors of **U** contain our Eigenvectors and the diagonal elements of **W** contain the corresponding Eigenvalues. Now the Eigenvectors of **U** are ordered regarding the value of their corresponding Eigenvalues. Each Eigenvector defines a principal component. Principal Component 1 (PC1) is the Eigenvector with the greatest Eigenvalue; PC2 is the Vector with the 2nd largest Eigenvalue and so on.

The new ordered **U** matrix is the requested matrix **P** of equation (2.30); **W** contains the Eigenvalues of $\lambda\mathbf{I}$.

Since **U** is an orthonormal matrix, it can be seen as a Transformation matrix, which transforms a vector from the input space into the space spanned by the Principal Components.

$$\mathbf{Y} = \mathbf{X}\mathbf{U} \tag{2.33}$$

Each component can be viewed as a weighted sum of conditions, where the coefficients of the Eigenvectors are the weights. The projection of gene *i* along the axis defined by the *j*th principal component is:

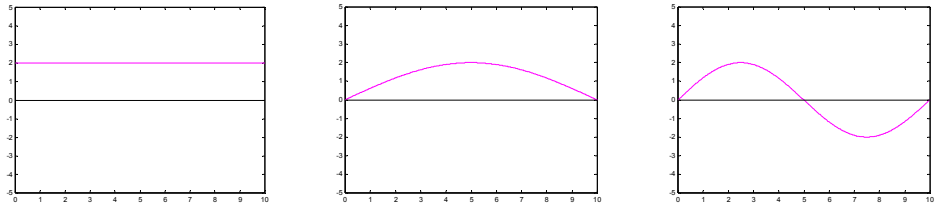
$$y_{ij} = \sum_{t=1}^{m} x_{it} u_{tj} \tag{2.34}$$

Where u_{tj} is the tth coefficient for the jth principal component; x_{it} is the expression measurement for gene i under the tth condition. **Y** represents the data in terms of principal components and is a rotation of the data from the original space of observations to a new space with principal component axes (PC Space).

The variance accounted for by each of the components is its associated Eigenvalue; it is the variance of a component over all genes. Consequently, the Eigenvectors with large Eigenvalues are the ones that contain most of the information; Eigenvectors with small Eigenvalues are uninformative.

2.8.3 Visualization

Each variation of a gene can be written exactly as a linear combination of these m characteristic principal components. Let's say, we have the following 3 principal components:



We can now use a 3-dimensional coordinate system, where the x-axis represents the Principal Component 1, the y-axis PC2 and the z-axis PC3. Plotted in this space are the rotated gene expression data points y_{ij}. The position of y_{ij} in Principal Component Space gives us information of with patterns the specific gene expression observation consists (Table 2.1).

That in turn means that points near to each other in the Principal Component Space are composed of the same basic patterns and are therefore similar in gene expression.

On the contrary, if two points are far apart from each other in the 3D-space, then they are not similar to each other in gene expression. If we use PCA to analyze experiments instead of genes, we can conclude that spatially distant experiments are not similar to each other and therefore represent distinct physical conditions.

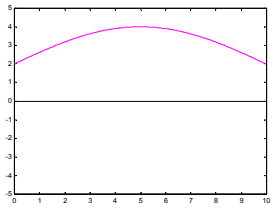
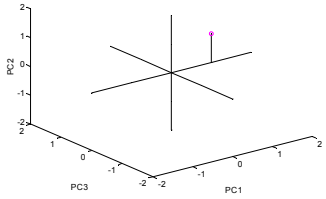
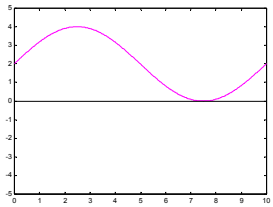
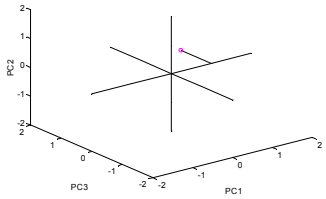
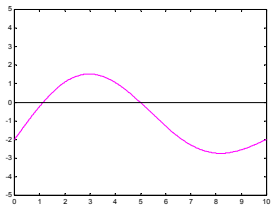
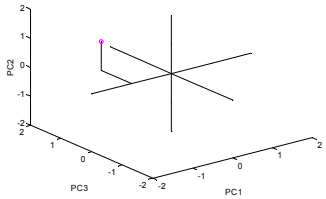
Description	Gene Expression	Point in PC Space
PC1 + PC2		
PC1 + PC3		
-PC1 + PC2 +PC3		

Table 2.1: Examples of gene expression curves composed of the basic patterns described in the first column and their location in PC Space.

2.8.4 Properties

Principal Component Analysis can be used to retrieve the basic patterns of gene expression contained in a given study. It eliminates the noise part of the dataset and concentrates on the most variant aspects of the investigated observation.

PCA can also be applied to study clusters of genes from other calculations in PC space. If the clusters are well self-contained, they are usually better than clusters that are spread across the PC Space.

It has to be mentioned at this place, that normalizing data adjustments remove a lot of the variation in the dataset and therefore impair the PCA. Normalization tends to project the genes/experiments on a circular or spherical shape in the PC Space.

Since this is an exact mathematical calculation, no parameters have to be specified.

2.9 Support Vector Machines

2.9.1 Introduction

Every of the clustering approaches we have discussed so far attempts to identify functionally significant classifications of genes in an unsupervised fashion. A learning method is considered unsupervised if it learns in the absence of a teacher signal. In contrast, Support Vector Machines (SVM) [66-70] uses a training set to specify in advance, which data should cluster together. It is therefore referred to as a supervised clustering method.

Let's assume there is a set of genes that share a common function or are part of the same pathway. In addition, a separate set of genes that are known not to be members of the investigated functional group or pathway is specified. These two sets of genes are combined to form a set of training examples, in which the genes are labeled positively, if they are in the studied functional class and are labeled negatively if they are known to be not in this functional class.

Using this training set, an SVM is able to learn to discriminate between the members and non-members of a given functional class. Having learned the expression features of the class, the SVM can recognize new genes as members or as non-members of the class based on their expression data. The SVM can also be reapplied to the training examples to identify outliers that may have previously been assigned to the incorrect class in the training set. Thus, an SVM uses the biological information in the investigators training set to determine what expression features are characteristic of a given functional group and use this information to decide whether any given gene is likely to be a member of the group or not (Figure 2.18).

SVMs are very helpful if only a fraction of the genes in an organism is functionally annotated. This information can be used to train a SVM to recognize and classify additional genes. In this scenario a subset of known genes of related function is set to be positive, the genes not part of this class are defined negative. After the SVM classifies the annotated genes correctly, the trained model can then be used to find possible candidates in the unknown set of genes, which are functionally related to the positive training examples.

SVMs have been proven to be superior to other neural network classification approaches such as Fisher's linear discriminant, Parzen windows, and decision tree learners [67].

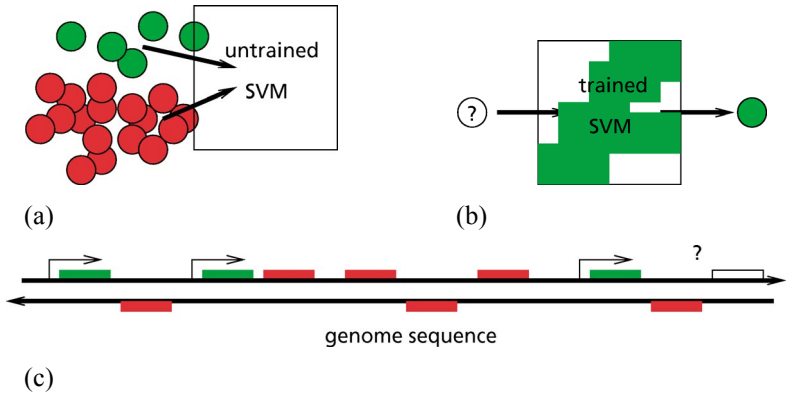


Figure 2.18: An untrained support vector machine (a) is trained with positive examples (green) and negative examples (red) to build a trained machine (b) that can take an unknown object (white) and determine whether or not it is similar to the training set. For functional classification of microarray data, positive and negative examples are drawn from annotations of coding regions in genome sequence data (c) [66].

Each vector **x** of the gene expression matrix can be seen as a point in a *m*-dimensional space (later referred to as input space), where *m* is the number of elements in **x**. A binary classifier can be built by constructing a hyperplane that separates class members (positive examples) from non-class members (negative examples) in this space (Figure 2.19). Unfortunately, real-world problems often contain nonseparable data, so that it is often not possible to find a hyperplane in the input space, which classifies all examples correctly. However, SVM possess the ability to transform the input space non-linearly into a higher dimensional feature space. In this feature space it is theoretical trivial to find a linear optimal separating hyperplane. A hyperplane is called optimal, if it separates all training vectors without error and the distance between the closest vectors to the hyperplane is maximal.

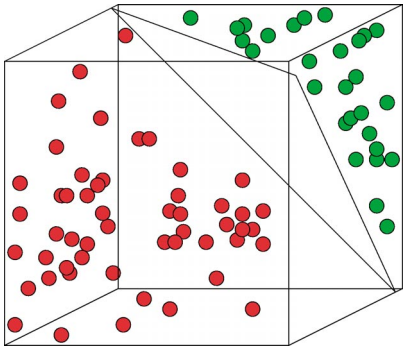


Figure 2.19: A SVM tries to find an optimal hyperplane that separates all training samples correctly and maximizes the margin (maximizes the distance between it and the nearest data point of each class). If this is not possible in the input space (for example in 2 dimensions) a hyperplane can be found in the higher dimensional feature space (e.g. 3D-space) [66].

2.9.2 Kernel Function

The main problem that arises with this approach is how to handle the problem of mapping all vectors from the input space into feature space. This mapping would generate high dimensional vectors entailing large storage and computational expense. However, since it turns out that just the dot-product of two vectors in feature space is required [68,69], the problem can be solved by introducing a so called kernel-function that describes the dot-product in feature space without ever representing the space explicitly. In other words; the idea of the kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. There are more than a dozen of different kernel functions available [69].

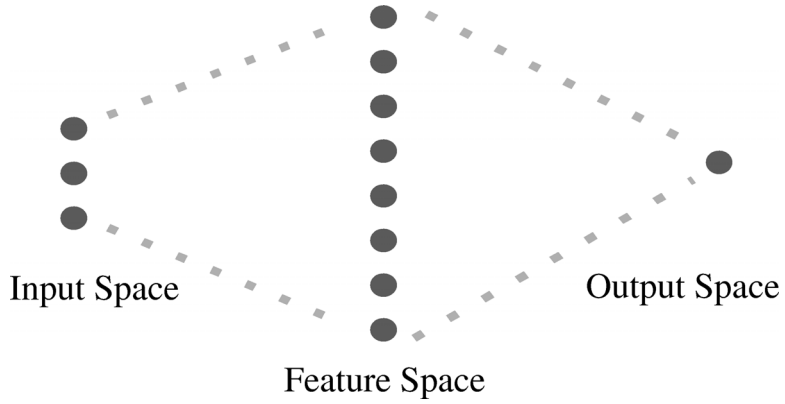


Figure 2.20: Mapping of m-dimensional input space into k-dimensional feature space, where $k \geq m$. The classification is calculated in feature space and yields to a result $y \in \{1,-1\}$ for positive or negative classification.

If we choose a mapping from the original space X into higher-dimensional space Z ($k:X \rightarrow Z$),

$$\mathrm{k}(\mathbf{x}) = \mathbf{z} \tag{2.35}$$

where $\mathbf{x} \in X$ and $\mathbf{z} \in Z$, the classifier that decides whether a vector is part of the positive or negative side of the hyperplane only needs the vectors in input space and their dot-product in the feature space [68,69]. For that reason we do not have to know $\mathrm{k}(\mathbf{x})$ as long as we know

$$K(\mathbf{x},\mathbf{y}) = \mathrm{k}(\mathbf{x}) \cdot \mathrm{k}(\mathbf{y}) \tag{2.36}$$

Depending on the kernel function used, different kinds of SVMs can be facilitated. Among acceptable mappings are polynomials, radial basis functions and certain sigmoid functions.

Polynomial Kernel

A polynomial mapping is a popular method for non-linear modeling:

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^d \text{ or} \tag{2.37}$$

$$K(\mathbf{x},\mathbf{y}) = ((\mathbf{x}\cdot\mathbf{y})+1)^d \tag{2.38}$$

d = 1, …

Gaussian Radial Basis Kernel

Radial basis functions have received significant attention, most commonly with a Gaussian of the form,

$$K(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right) \tag{2.39}$$

σ Is the width of the Gaussian

2.9.3 The generalized optimal separating hyperplane

So far the discussion has been restricted to the case where the training data is linearly separable. However, in general this will not be the case. For some data sets, the SVM may not be able to find a separating hyperplane in feature space, either because the kernel function is inappropriate for the training data or because the data contains mislabeled training vectors. The latter problem can be addressed by using a soft margin that allows some training examples to fall on the wrong side of the separating hyperplane [68,69].

2.9.4 Properties

The software described in this paper is able to calculate SVM with Polynomial and Radial Kernels. The biggest problem with SVM is the specification of all necessary parameters. It is often very difficult to choose the right kernel and find the best kernel-parameters. Different parameters yield to completely different classifications and different distance measurements need different kernel parameters to get correct classifications. It is therefore necessary to iteratively increase the kernel complexity, until the desired classification precision is reached.

Very high kernel orders on the contrary tend to artificially separate the data and may expose the learning system to the risk of finding trivial solutions that overfit the data.

2.10 Programming tools

2.10.1 Introduction

Java (Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA, www.sun.com) has been chosen as programming language. Additionally to the Java 2 Software Development Kit (SDK), Standard Edition Version 1.3.0, the Java 3D API Version 1.2 (Sun Microsystems) has been used for the three-dimensional representation of PCA results. For the program development the integrated development environment (IDE) JBuilder (Inprise Corporation, 100 Enterprise Way, Scotts Valley, CA 95066 USA) has been used.

2.10.2 Java

Java technology is both a programming language and a platform [71].

Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords [72]:

- Simple
- Architecture-neutral
- Object-oriented
- Portable
- Distributed
- High-performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

In the Java program language a program is first compiled into an intermediate language called Java bytecodes - the platform-independent codes that are then interpreted by the interpreter on the Java platform.

Java bytecodes can be seen as the machine code instructions for the Java Virtual Machine (Java VM), which parses and runs each Java bytecode instruction on the computer. After compiling a program into bytecodes with a Java compiler, the bytecodes can then be run on any platform that has a Java VM, no matter if it is a Windows 2000 or Solaris workstation, or an iMac (Figure 2.21).

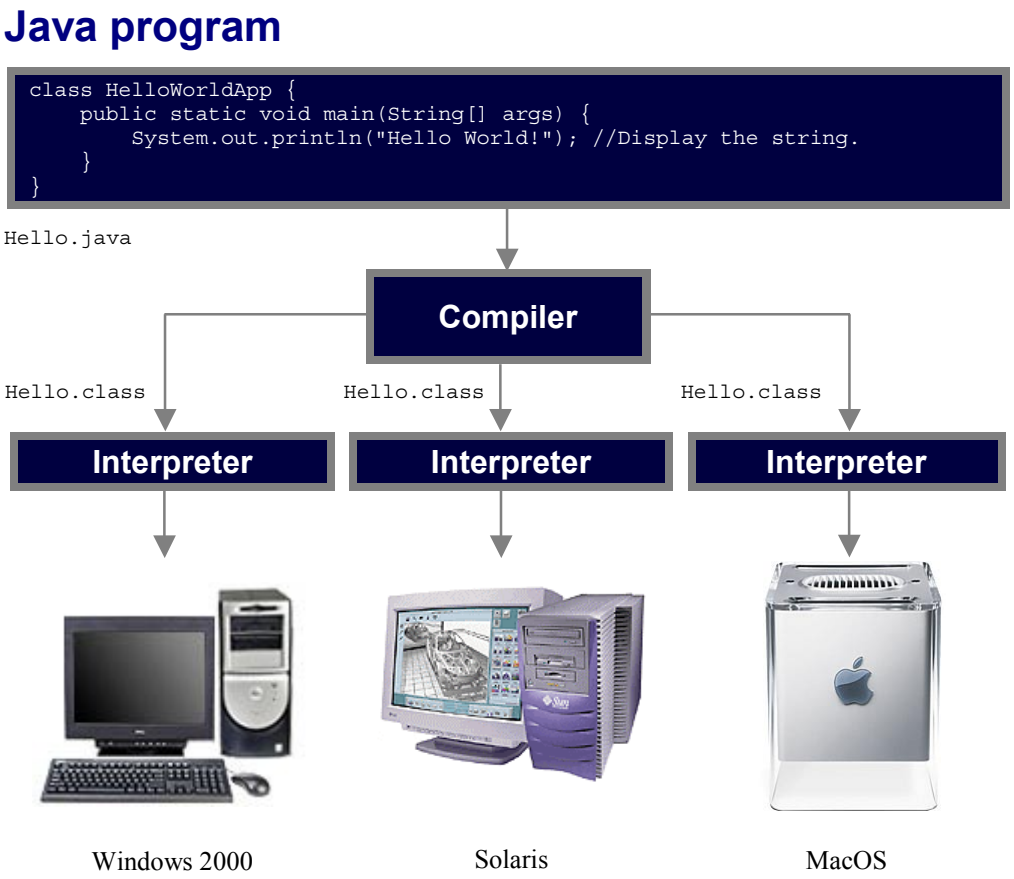


Figure 2.21: The compiler builds one class file (bytecode) that can be interpreted by the VM of all platforms [71].

The Java Platform

The Java platform is a software-only platform that runs on top of other hardware-based platforms like Windows 2000, Linux, Solaris, Irix and MacOS.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *package*s.

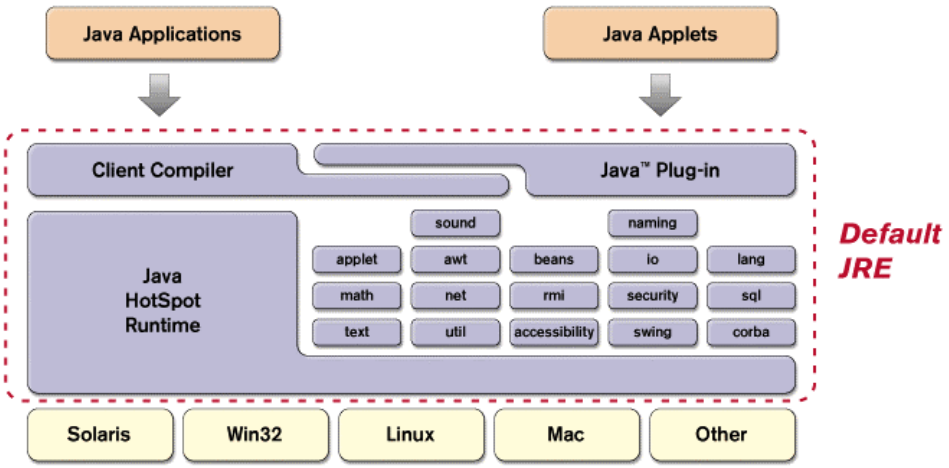


Figure 2.22: Java™ 2 Runtime Environment, Standard Edition Version 1.3: The Java API and the virtual machine build the Java Runtime Environment (JRE) that insulates the program from the hardware.

Native code is code that after it is compiled, runs directly on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time bytecode compilers can bring performance close to that of native code without threatening portability.

The Solaris, Linux and Microsoft Windows versions of the Java 2 Platform, Standard Edition version 1.3 contain the Java HotSpot Client VM, including a specially tuned client compiler. This configuration delivers:

- Better performance than any previous Java platform.
- Faster start-up.
- Smaller memory footprint.
- Faster interactive performance.
- Virtually pause-less garbage collection.

Since clustering of gene expression data is extremely computational intensive Java 1.3 was used because of the speed gain in comparison to Java 1.2. Java 1.3 has proven to be factor 4 to 5 faster than the previous release, enabling Java applications to be almost as fast as native codes form c and c++ compilations.

Chapter 3

Results

In this chapter we survey results, which are obtained by analyzing gene expression data with the software presented in this paper. For demonstration purposes we use published and publicly available data and compare the results with the published analysis gained from that dataset.

3.1 Sources of Experimental Data

The Gene expression of primary human fibroblasts stimulated with serum following serum starvation was studied by Iyer et al. using a microarray with 9,800 cDNAs (Figure 3.1) [73].

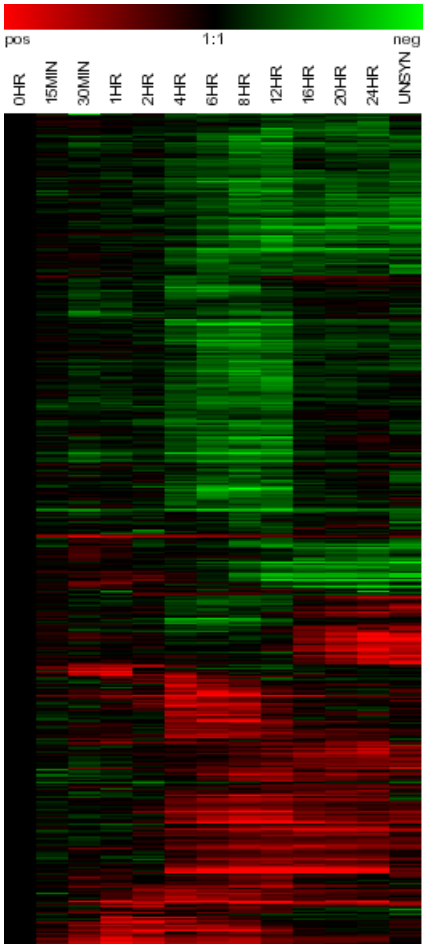


Figure 3.1: Expression image showing 517 genes in 12 different time points of serum stimulation of primary human fibroblasts. Foreskin fibroblasts were grown in culture and were deprived of serum for 48 hr. Serum was added back and samples taken at time 0, 15 min, 30 min, 1 hr, 2 hr, 3 hr, 4 hr, 8 hr, 12 hr, 16 hr, 20 hr, 24 hr. The final data point was from a separate unsynchronized sample. Data were measured by using a cDNA micro-array with elements representing 8613 distinct human genes. All measurements are relative to time 0. Genes were selected for this analysis if either (1) their expression level deviated from that in quiescent Fibroblasts by at least a factor of 2.20 in at least two samples from serum-stimulated cells or (2) the standard deviation for the set of 13 values of $\log_2$ (expression ratio) measured for the gene in this time course exceeded 0.7. In addition, observations in which the pixel-by-pixel correlation coefficients for the Cy3 and Cy5 fluorescence signals measured in a given array element were less than 0.6 were excluded. The color scale ranges from saturated green for log ratios -3.0 and below to saturated red for log ratios 3.0 and above. Each gene is represented by a single row of colored boxes; each column represents a single time point [73].

DNA microarray hybridization was used to measure the temporal changes in mRNA levels of 8613 human genes at 12 time points, ranging from 15 min to 24 hours after serum stimulation. From this 8613 a subset of 517 genes whose expression changed substantially in response to serum is chosen for a cluster analysis using (1) Hierarchical Clustering, (2) k-means, (3) Self Organizing Maps, and (4) Principal Component Analysis. The entire detailed data set is available as a tab-delimited table at the Stanford Web site (genome-www.stanford.edu/serum).

Since this is a time series and precalculated dataset, no data adjustments are used. For the distance measurement the program default parameter (see Appendix A) is used, so that for each clustering procedure the usually best distance measurement is chosen automatically.

3.2 Hierarchical Clustering

Average Linkage, Complete Linkage and Single Linkage Clustering have been implemented. The clustering can be processed on genes and/or experiments. The algorithm has been tuned for optimal performance: whole yeast genome (6116 genes) in 66 seconds (PIII 750, Windows NT, Java 1.3), which is about 40 to 50 times faster than the straightforward implementation without reference tables to prevent unnecessary searches in the matrix.

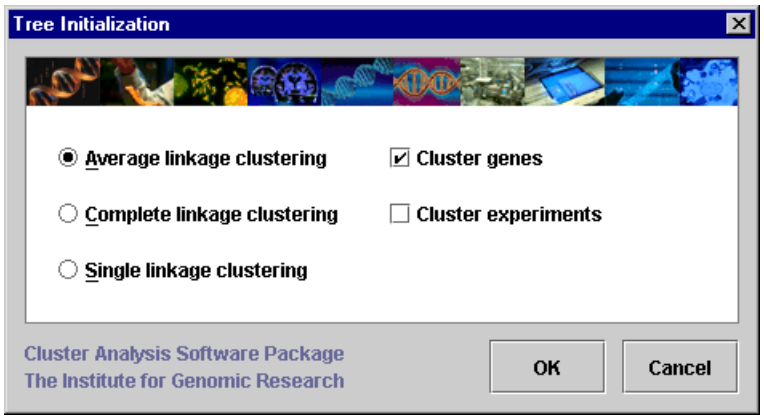


Figure 3.2: Hierarchical Clustering Dialog.

After the calculation is finished, the tree result is shown and clusters can be interactively specified. For our sample data Average Linkage Clustering with Euclidian Distance was used, since Average Linkage usually gives the best results for gene expression data. The tree is well distributed and clusters are easily visible by inspecting the ordered expression image. For the fibroblast dataset we can distinguish 10 different clusters, marked A to J in Figure 3.3.

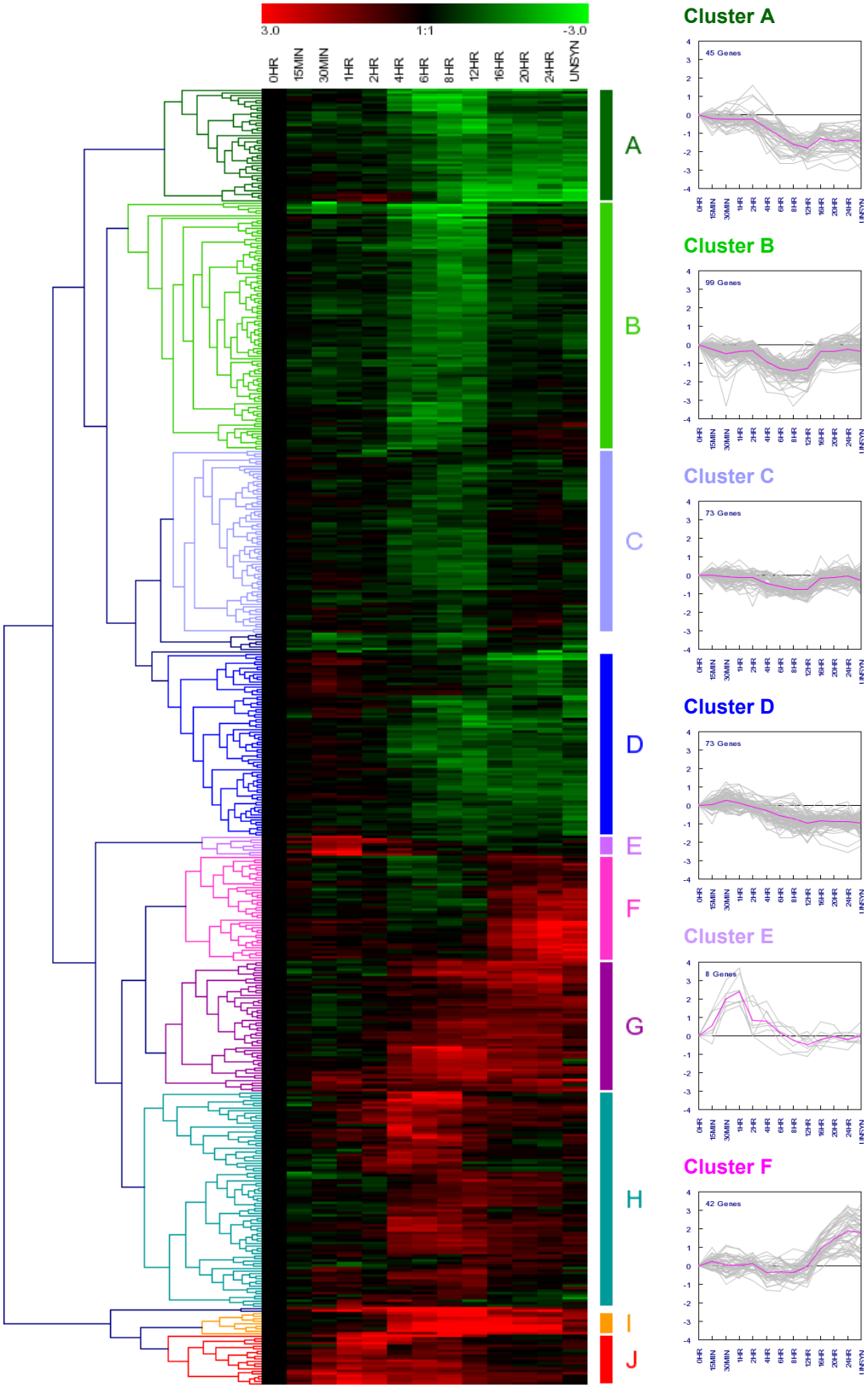


Figure 3.3: Hierarchical clustering result using Average Linkage Clustering and Euclidian Distance. (l) Tree and cluster image (r) Expression plots of all genes and median of a cluster.

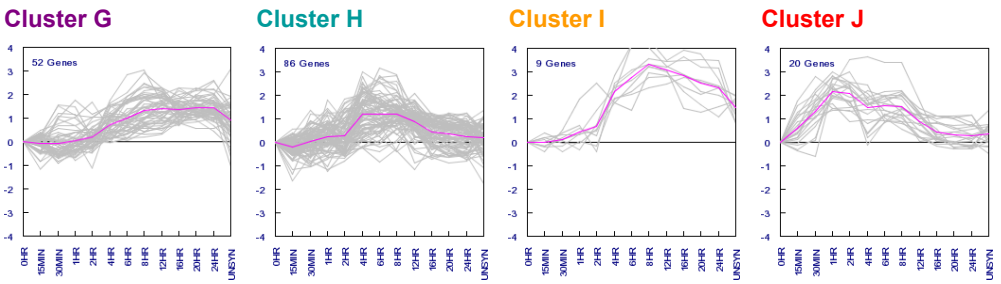


Figure 3.4: Expression plots of all genes and the median of a cluster.

Every pattern can easily be distinguished from others. Some are completely different (E and F); some are just different in magnitude compared to other clusters (B and C). The clusters have been interactively specified by looking on the patterns in the expression image and then selecting the corresponding sub tree on the left side. Each cluster can be assigned with a color and a text. The clustering result can be saved as an image or data. It is also possible to change the appearance of the tree in various ways.

3.3 k-means Clustering

As described in chapter 2.7, the number of clusters has to be predetermined. Additionally, the maximum number of iteration cycles has to be specified. Usually the algorithm converges (no more reallocations between the clusters) before 50 cycles, but it can occur that the convergence criterion is not reached in a specific time. Therefore, to prevent infinite calculation, the maximum number of cycles has to be declared.

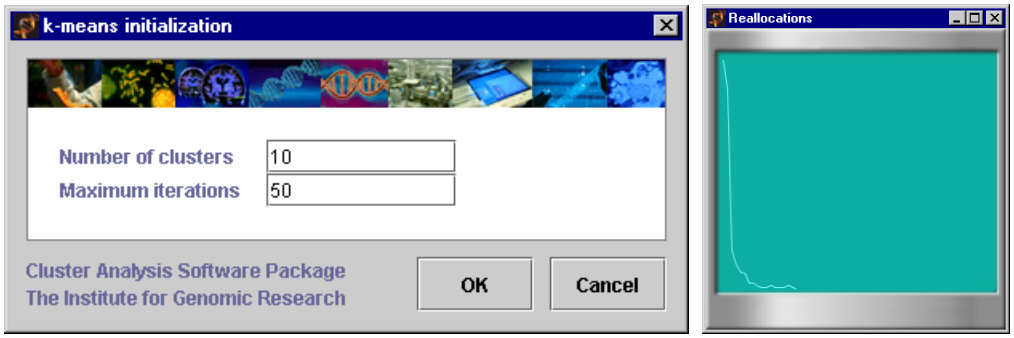


Figure 3.5: (left) k-means clustering initialization dialog. (right) k-means convergence plot.

For this analysis, k=10 was used for the number of clusters, since 10 basic patterns have been found by hierarchical clustering. As it is demonstrated here, hierarchical clustering can be used to predetermine the number of clusters for a specific dataset.

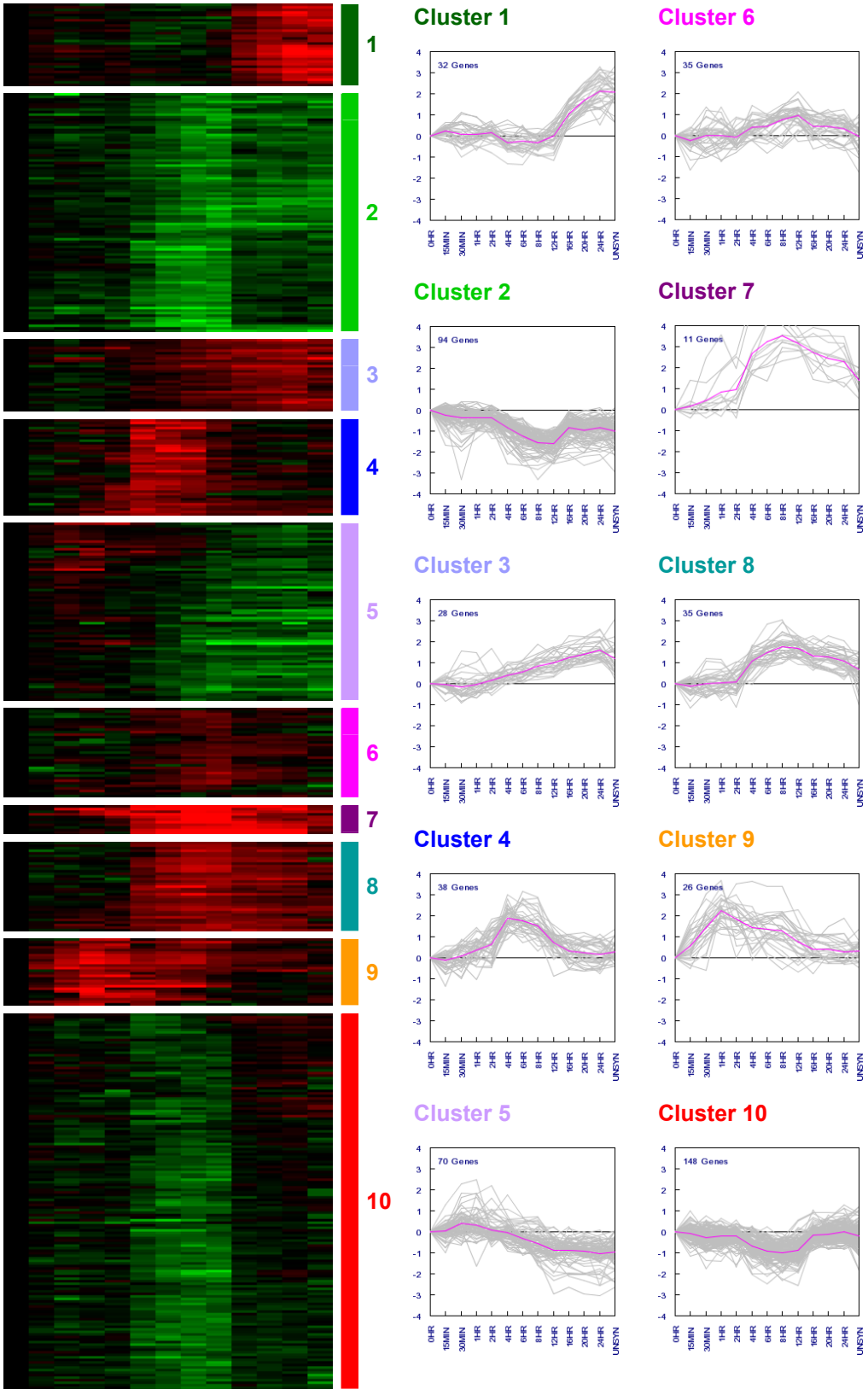


Figure 3.6: k-means clustering result using Euclidian Distance and 10 clusters. (l) Expression images of the clusters. (r) Expression plots of all genes and the median of a cluster.

As we can easily see, the clusters are very similar to the clusters found with hierarchical clustering. Cluster sizes and gene distribution are of course different, but both algorithms have found the same basic patterns.

3.4 SOM Clustering

SOM needs much more information from the user than the previously discussed algorithms. Figure 3.7 shows all the parameters that can be specified. All fields have default values, but it is recommended to vary the parameters to achieve the best clustering, since changing parameter yields different results. In general, the resulting patterns do not change very much by changing the learning and initialization parameters; what usually changes is the assignment of a gene to the one or other cluster. Most important are the dimensions of the SOM network and the number of iterations. The net dimensions specify the number of clusters by $n=\dim_x * \dim_y$, the iterations field how many steps are calculated until the algorithm stops. The calculation time therefore depends primarily on the number of clusters and the number of iterations, not on the number of genes in the dataset! Here 517,000 iterations are chosen to hit each gene vector 1000 times on average, since there are 517 genes in the dataset. A 5x2 network is chosen to get an n=10 like in the previous clustering calculations. All the other parameters are let on their default value as shown in Figure 3.7.

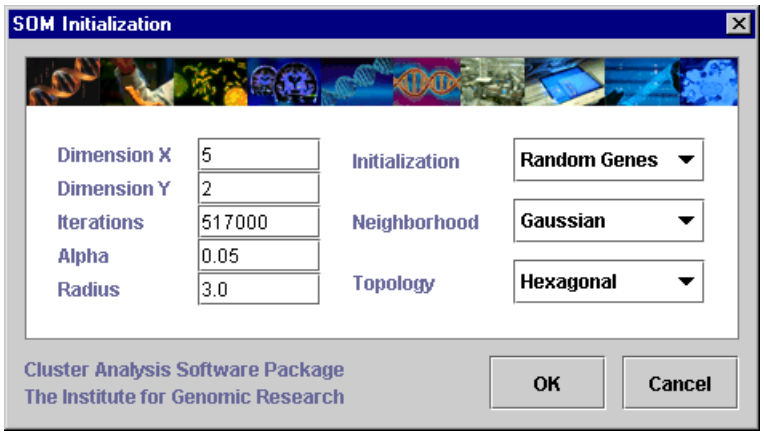


Figure 3.7: SOM clustering initialization dialog.

Since the initialization is random, different calculations lead to different results. SOM and k-means have the disadvantage that they can find local instead of global minimums. Therefore several SOMs (and also k-mean clusters) have been calculated and compared to get the best possible result. The basic patterns have been found here too, however, cluster integrity, size, and gene distribution differ slightly (Figure 3.8).

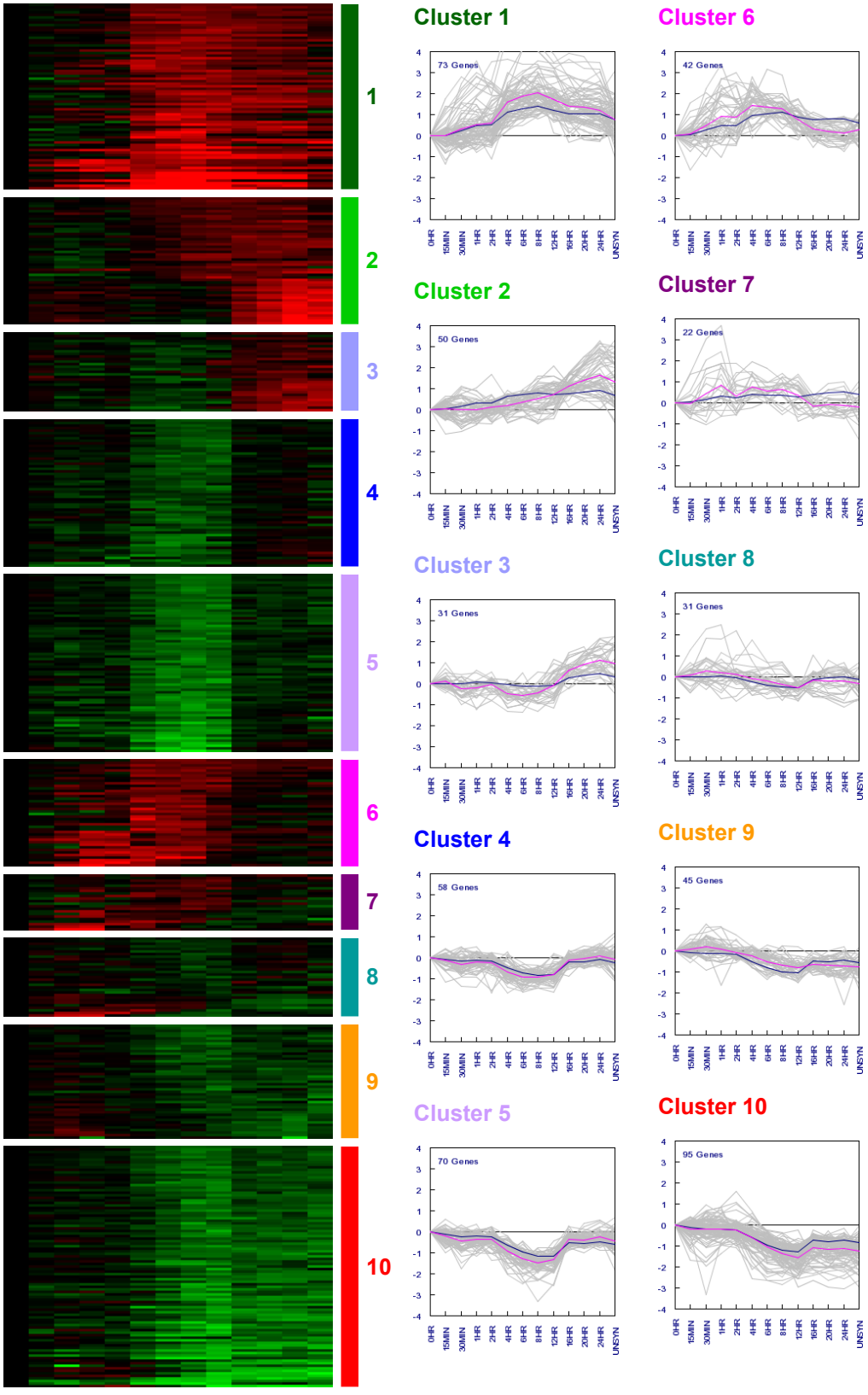


Figure 3.8: SOM clustering result using Euclidian Distance. (l) Expression images of the clusters. (r) Expression plots of all genes, the SOM vector(blue), and the median of a cluster.

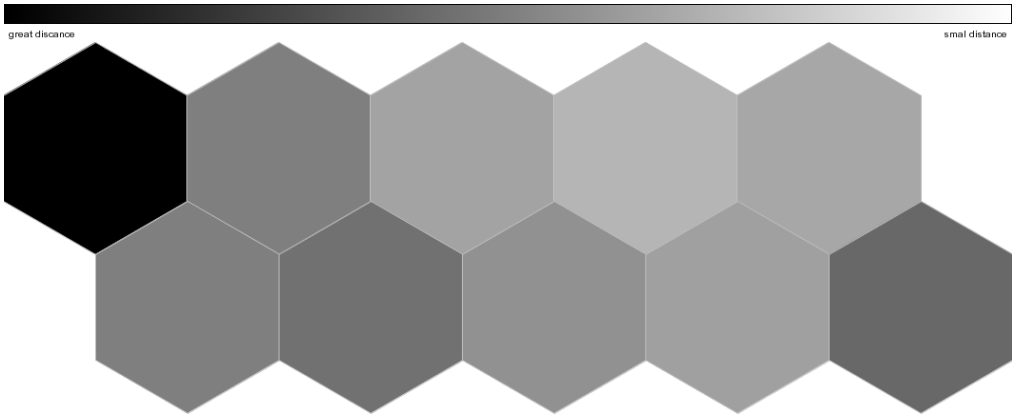


Figure 3.9: Kraaijveld representation of the 5x2 SOM. The maximum distance between a unit and its cluster members is coded in gray-levels. The greatest cluster size is represented by black, the smallest by light gray fields. Cluster 1 to 5 in first row, 6 to 10 in second row.

Figure 3.9 nicely demonstrates the variance of the genes in a specific cluster. We can immediately see that Cluster 4 has the least variance, and cluster 1 has the greatest. The expression plots in Figure 3.8 support this conclusion.

3.5 Comparison

For a better visibility of the similarity of clusters from different algorithms, two sets of clusters are chosen for a comparison. As it can easily be seen cluster size and gene distribution vary, but the means of the clusters are very similar to each other!

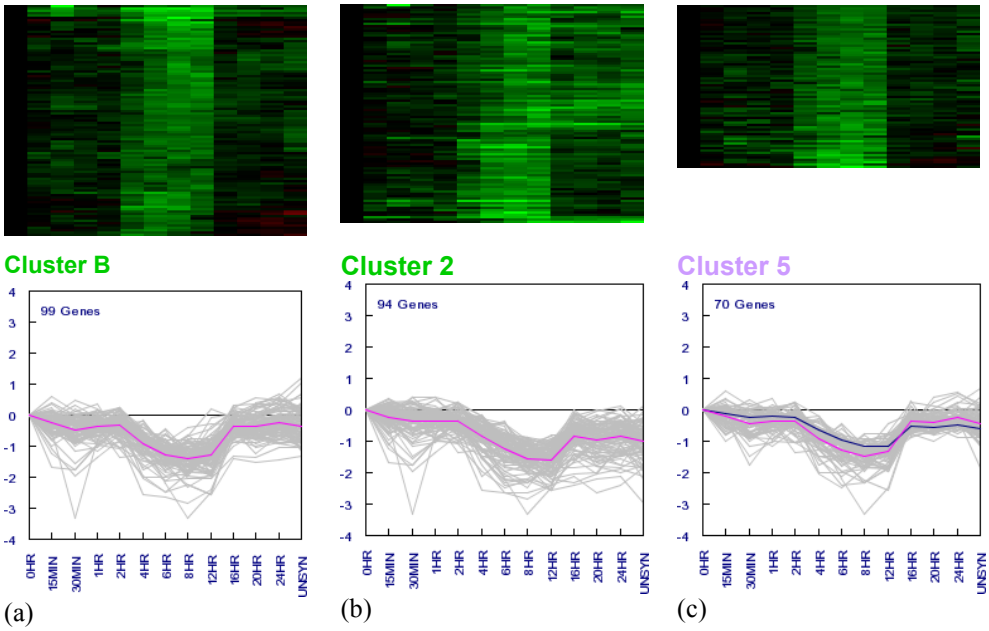
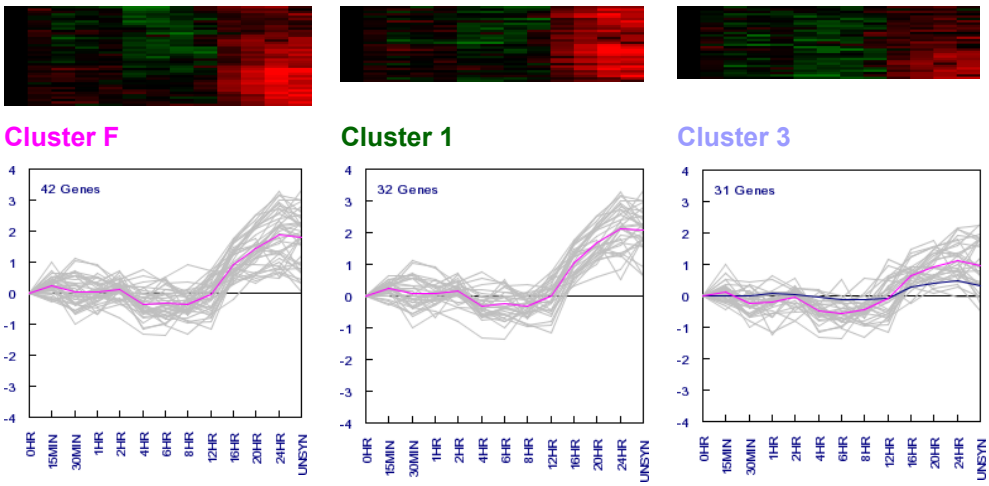


Figure 3.10: Comparison of clusters with under expressed genes. (a) Cluster B from Hierarchical Clustering (b) Cluster 2 from k-means and (c) Cluster 4 from SOM clustering.

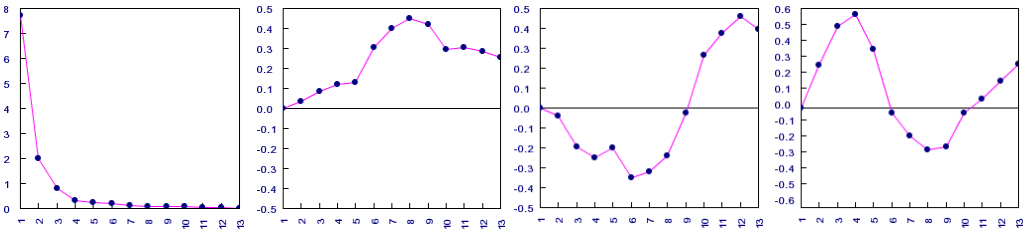


(a) (b) (c)

Figure 3.11: Comparison of clusters with over expressed genes. (a) Cluster F from Hierarchical Clustering (b) Cluster 1 from k-means and (c) Cluster 3 from SOM clustering.

3.6 PCA Genes

No parameters are necessary for calculating a Principal Component Analysis (PCA). PCA is a very efficient method and it takes just about one second to calculate the result for this dataset. Figure 3.12 shows a plot of the Eigenvalues and the first 3, most significant patterns in the dataset. Pattern 1 is even with just looking onto the expression image visible (see tree clusters B and C for instance). It contains about two third of the information in the dataset. Principal components 2 and 3 have oscillating patterns around the x-axis. The first 3 PCs combine more than 90% of the variance, so that the components 4 to 13 have all together less than 10% of the information. Their patterns mainly describe the noise component in the dataset; PC13 has less than 5 ppm of the information!



(a) (b) (c) (d)

Figure 3.12: PCA results. (a) Eigenvalue plot (b) plot of Principal Component 1 (c) plot of Principal Component 2 (d) plot of Principal Component 3

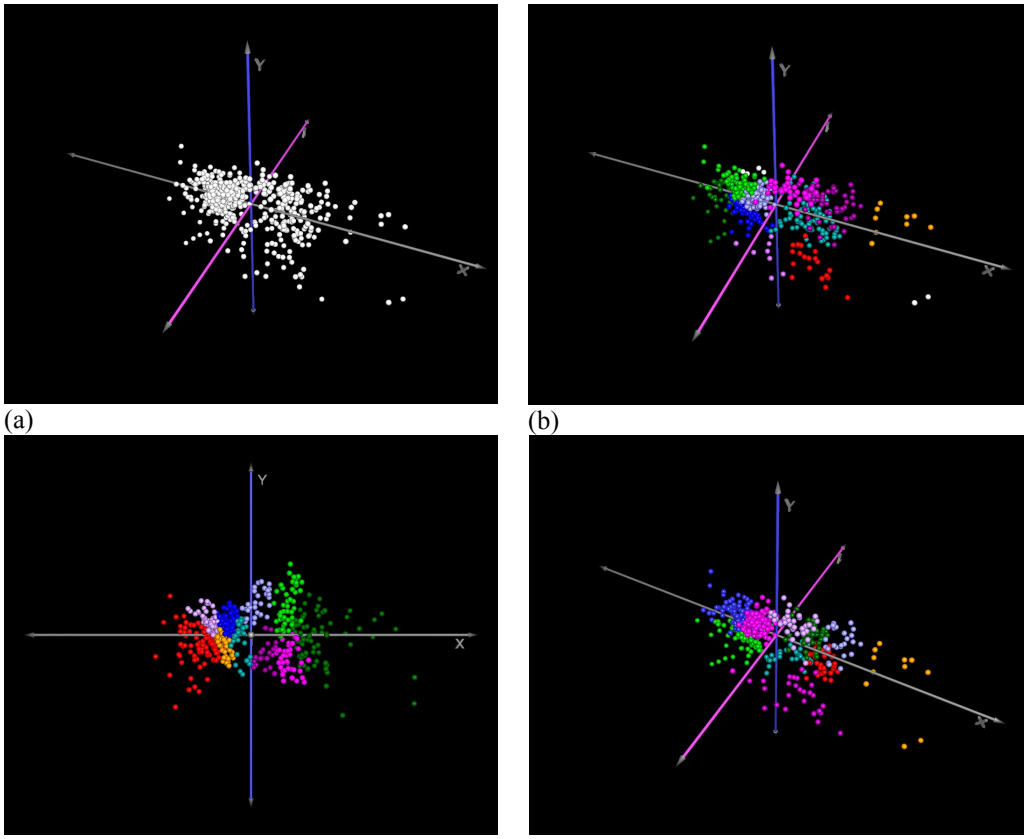


Figure 3.13: PCA from Genes. The x-axis represents PC1, y-axis PC2 and z-axis PC3. (a) plain result (b) HCA clusters colored in the PC space (c) SOM clusters (d) k-means clusters.

	Eigenvalue	Percentage of variance
Principal Component 1	07.725	66.199 %
Principal Component 2	01.989	17.048 %
Principal Component 3	00.807	06.919 %
Principal Component 4	00.303	02.600 %
Principal Component 5	00.236	02.026 %
Principal Component 6	00.198	01.695 %
Principal Component 7	00.116	00.994 %
Principal Component 8	00.088	00.754 %
Principal Component 9	00.074	00.637 %
Principal Component 10	00.065	00.555 %
Principal Component 11	00.039	00.334 %
Principal Component 12	00.028	00.238 %
Principal Component 13	00.000	00.000 %

Table 3.1: Eigenvalues and Eigenvalue distribution

Figure 3.13 shows the 3-dimensional view of the data in Eigenvector-space. Clusters from different clustering calculations can be visualized using colored points in the 3D-view. In general it can be determined that some clusters generate a compact data cloud in space. Therefore, PCA can be used to determine how compact and self-contained a cluster of genes is.

3.7 PCA Experiments

This is a very good example how PCA can be used to investigate relations between different experiments. If we look on the 3D-view (Figure 3.14), we can see that the experiments "move" through the Eigenvector space along a specific path. Time point 0 is in the center of the coordinate system. In the first 2 hours the cell moves in direction of positive x and negative z, than back until time point 12h and finally ends in a stage represented by the experiments at 14, 20, 24 hours and the unsynchronized sample. This analysis can also be used to determine the most distant or most related experiments in a study. For instance in cancer research it is important to find the most distant experiments, since they represent the difference between a cancer cell and a normal cell best.

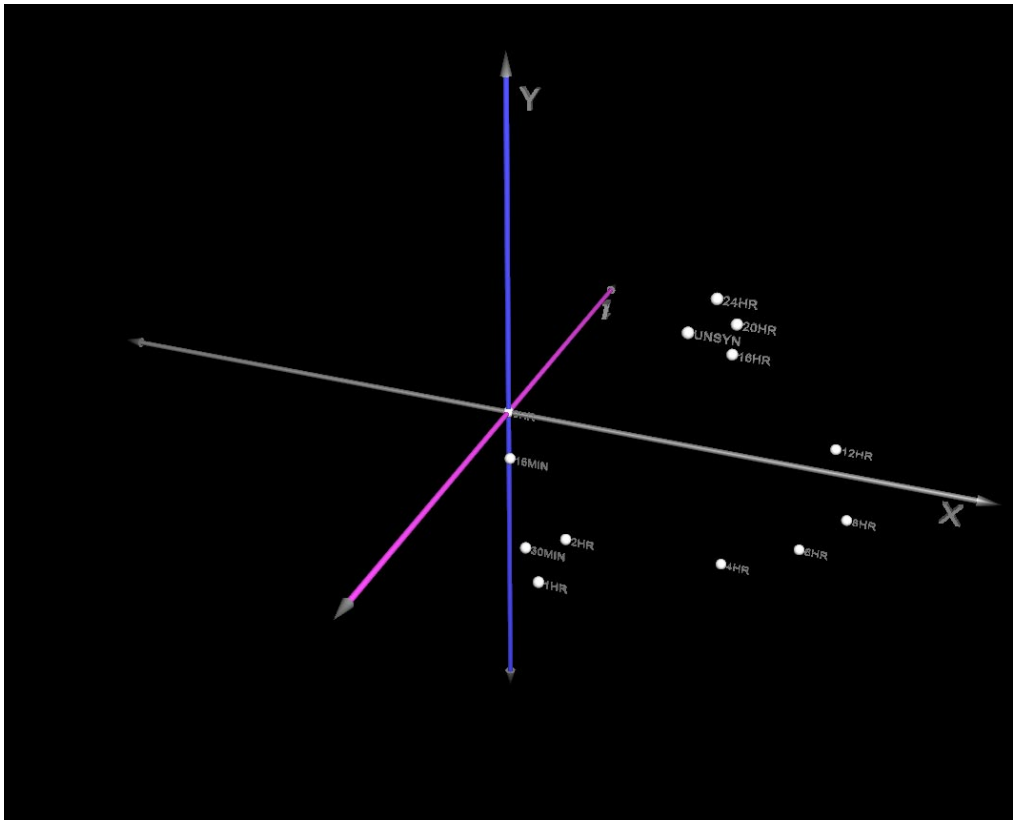


Figure 3.14: PCA result of experiment comparison.

3.8 Support Vector Machines

To demonstrate SVM gene classification, 6 expression vectors from 4 genes involved in the Cholesterol biosynthesis [73] are used for the SVM training with only these 6 vectors classified positive and all others remaining negative (Figure 3.16 top). Although Cytochrome P450, 51 (lanosterol 14-alpha-demethylase) (CYP51) is involved in Cholesterol biosynthesis, it has not been used as a positive example to have a control for the validity of the classification. After training, the dataset has been classified using the trained SVM. Since the dot-product distance measuring method hasn't given good results, the Covariance distance measurement has been used in combination with a Radial Kernel with 1.135 as width factor (Figure 3.15). The radial kernel width factor has been determined by increasing it, until CYP51 has been classified correctly.

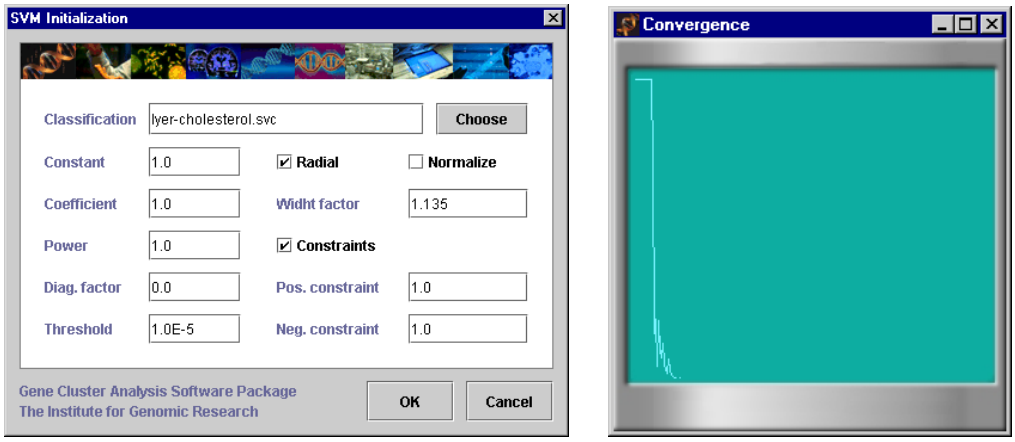


Figure 3.15: (left) SVM training initialization dialog. (right) SMV training convergence plot.

All initial positivly classified genes (including CYP51) have been classified correctly and 46 additional gene vectors have been classified to be potential genes involved in Cholesterol biosynthesis (Figure 3.16). In total, five gene expression vectors representing members of the Cytochrome family have been found.

Similar to the expression profile of the five genes of the Cholesterol biosynthesis pathway, almost all positive classified genes show a sharply diminished expression beginning 4 to 6 hours after serum stimulation of fibroblasts. This suggests that they are either also part of the investigated pathway or part of an other pathway that is active in a similar fashion.

24 of the 46 genes are annotated genes, the remaining 22 genes can now be further investigated to support or reject the hypothesis of their possible role in the Cholesterol biosynthesis pathway.

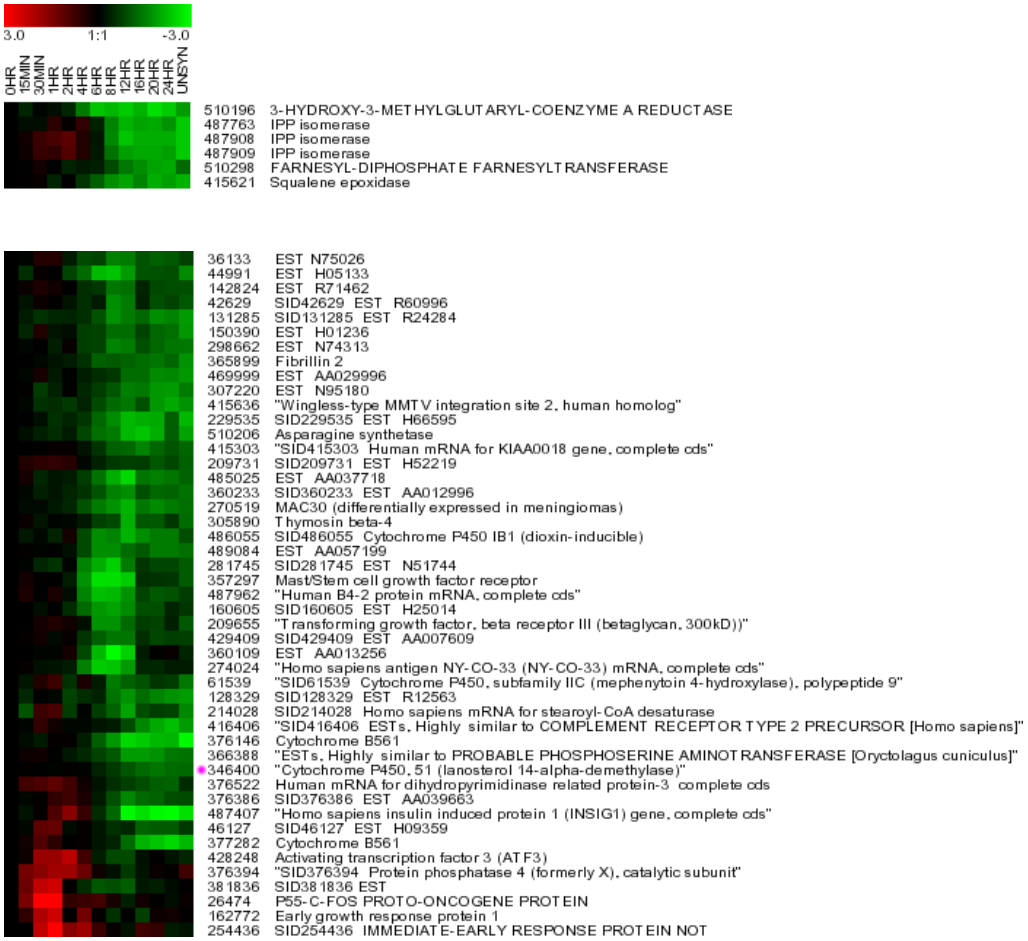


Figure 3.16: SVM classification (top) The six training genes known to be part of Cholesterol biosynthesis. (bottom) 47 genes (including CYP51) positive classified by the SMV as potential genes involved in Cholesterol biosynthesis.

3.9 Chromosomal Mapping

It is also possible to import GenBank flat files into the program by just adding another line to the XML-file that represents the program tree on the left part of the program window. The sequence and annotation information contained in these files can then be used for further analysis. One example is the mapping of gene expression studies onto a chromosome of the studied organism. Consecutive genes are often similarly expressed and can be easily identified by this method (Figure 3.17). This can have various reasons. One is that the genes are so similar that cross-hybridization effects occur, a second could be that the consecutive genes are regulated by the same regulatory network. The latter information can be used for further analysis of the transcriptional program of these genes.

Unfortunately the annotation of the human chromosomes is not finished yet and therefore not available in the Genbank flatfiles. To demonstrate this program feature chromosome III of the model organism *saccaromyces cerevisieae* (yeast) is used, since its sequence is fully annotated. As microarray experiment a sporulation dataset form Stanford University (genome-www.stanford.edu) [74] is chosen, because it contains almost all genes of the genome. Unfortunately the annotation is not very consistent, therefore not all genes from the experiment can be mapped. For that reason it is important that the official gene names from Genbank are used in the microarray data files!

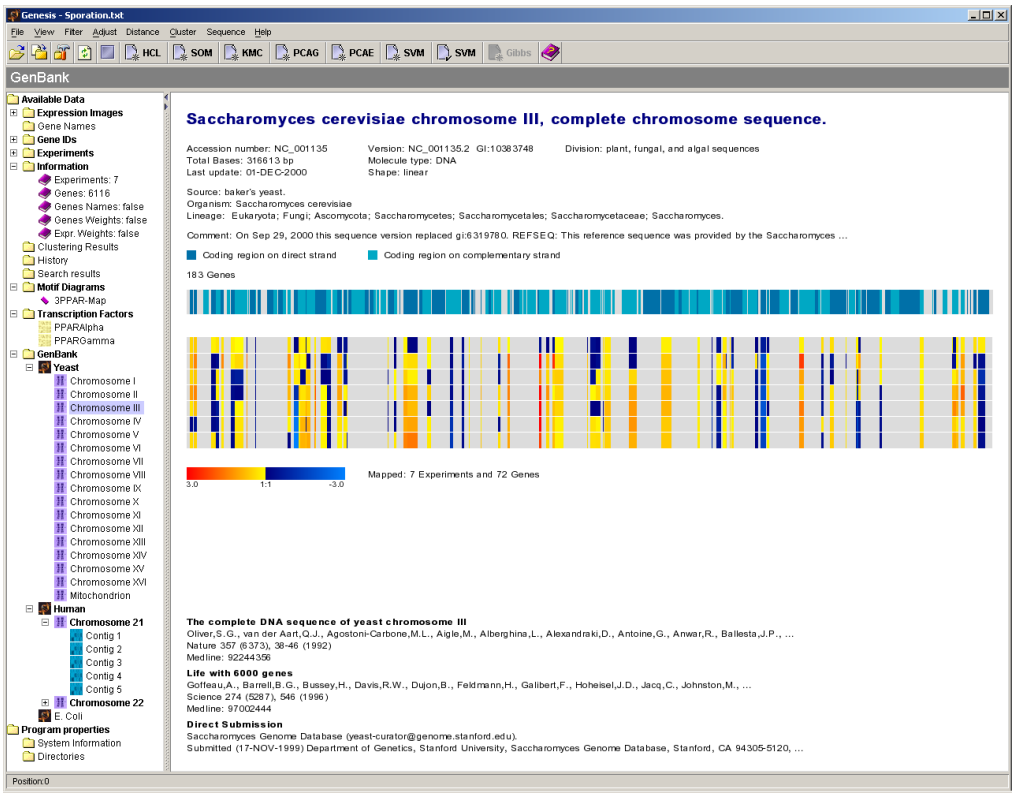


Figure 3.17: Program window showing the chromosomal mapping of a yeast sporulation gene expression study onto chromosome III of saccharomyces cerevisiae. First 3 genes are consecutive and similarly expressed.

Chapter 4

Discussion

In this work a versatile and transparent software suite for large-scale gene expression cluster analysis has been developed. The developed software enables data import and visualization, data normalization, and clustering using: (1) Hierarchical Clustering, (2) k-means, (3) Self Organizing Maps, (4) Principal Component Analysis and (5) Support Vector Machines. Additionally, motif finding and visualization tools have been implemented, including a basic Gibbs sampler, the drawing of sequence logos and motif diagrams as well as the possibility to search motifs in GenBank and FastA files with either a consensus sequence or a position weight matrix (PWM). Finally, mapping of gene expression data onto chromosomal sequences was included to enhance the investigation of regulatory mechanisms.

An important and valuable feature of this software is to calculate and compare clustering results from different algorithmic approaches. One of the challenges in analyzing microarray data is the fact that there is no biological definition of a gene cluster. Moreover, due to the different underlying assumptions for the clustering techniques and the necessity to adjust various parameters, the clustering results can differ substantially. Thus, it is an imperative to apply several clustering techniques on the same data set and to compare the results. The comparison of clusters obtained using several clustering techniques enables the biologists to identify genes and/or experiments that have been rated similar in all clustering results. For example, one can begin with Hierarchical Clustering to get a first impression on the number of patterns hidden in the dataset and then use this information to adjust the parameters for k-means and SOM clustering. PCA can be used to visualize these clusters in 3D space and get an impression on cluster size, integrity, and distribution, and to retrieve the most significant patterns in a study. It can also reveal some information about the number of clusters in the dataset, provided that data clouds of genes in the principal component space representing a cluster can be distinguished. All these clustering and classification procedures enable us to get an impression of what subset of genes or experiments represents the most significant information for a given investigated condition and therefore provides the opportunity for researchers to concentrate on particular target genes or experiments.

The selection of the right similarity distance measurement is very important and difficult task. Researchers have to answer the questions which of their investigated genes or experiments should be considered similar. Are two genes that are similar in shape but different in magnitude similar or not? Is a gene that is overexpressed similar to an underexpressed gene with the same time-course? By answering this questions and knowing the properties of the different procedures the best fitting distance measurement can be found. However, sometimes it is necessary to use several methods and choose one on the basis of the results gained with this function. In general it is useful to start with less complex distance functions and move to more sophisticated procedures if no satisfying results can be achieved.

Since clustering is a statistical approach of getting information out of complex data, the results are often not easy to interpret. Therefore extensive work has been done to visualize the results as intuitive and informative as possible, not only for computer scientists and mathematicians, but also and most important for biologists. The user input in all this mathematical approaches is very important, since different parameters can lead to different results and normalization can change the data in an unintentional way. For that reason the investigating researcher should be familiar with the mathematical procedures and the effects they can have to the biological information gained from such analysis methods. Improper analysis can and will yield to false results and wrong conclusions. It is also essential that results received from clustering analysis can only be seen as an indication of possible relationships. Verifying these presumptions in biological experiments is indispensable!

The techniques introduced in this paper have proven to be useful to unveil meaningful biological information but they can be just the starting point in the analysis of gene expression data. Many problems are at the moment not fully addressed and need to be further investigated. Substantial investment and effort will be necessary to develop new analytic methods that elucidate more complex correlations like non-linear or multi-input relationships. The algorithms presented here can only be seen as first steps towards a more extensive and proper analysis system of the complex processes of gene expression and regulation in living cells.

The field of microarray technology and genomic research in general is developing at an astonishing rate. It will be necessary to continuously improve and adapt the developed software to the newly gained knowledge and standards to maintain the status of a valuable and flexible software package in functional genomics. Some possible improvements for the near future include:

General program features

- Import/Export of complete results for later editing
- Input from XML and database resources
- Implementation of filter and significance tests
- Automatic coloring of clusters for cross-cluster analysis
- Import of saved clusters for cluster comparison
- Thumbnail window for better navigation
- User preference files and user accounts
- Import of user defined queries for each gene, links to GenBank, etc.
- Tracking of the command history and establishing "analysis procedures"
- Additional graphical displays [75]
- Histogram drawing
- License manager

Data adjustment and similarity distance

- Jackknife method [26]
- Linear regression, total ratio and ratio statistics normalization for experiments
- Visualization of similarity distance matrices

k-means and SOM

- K-means and SOM also for experiments
- Fuzzy clustering (one gene can belong to more than one cluster or no one)

Hierarchical Clustering

- Implementation of Ward's method and weighted average linkage clustering
- Implementation of Neighborhood Joining algorithm
- Implementation of Bootstrap algorithms for tree analysis
- Automatic scaling of the tree
- Expression and centroid plots of tree clusters

Principal Component Analysis

- Import/Export of complete results for better editing.
- Gene shaving algorithms [75].
- Better navigation in 3D-result window.
- Automatic search of genes similar to specific patterns (PCs)
- Eigengene and Eigenexperiment Analysis [Alter]

Support vector machines

- Expression image generation of positive classified genes (expression and centroid plots)
- Automatic iterative best parameter search for best classifications
- Additional kernel functions

GenBank module

- Wizards for easy adding, managing and removing of GenBank entities.
- Automatic update of GenBank flat files from the NCBI.
- Calculation and display of Chromosome correlation maps [77]

One very important feature in the near future will be the ability to import data in the currently developing standard format for microarray data. The Microarray Gene Expression Database group (MGED, www.mged.org) consisting of the most influential institutes in microarray research and database management have developed two standards, called MIAME (Minimum Information About a Microarray Experiment) and MAML (MicroArray Markup Language) which will allow the exchange of datasets in a well defined format between research facilities and databases all over the word. It will become very important and helpful for data exchange.

The program has been developed in Java because of the platform independency, which is very important in this field, since biologists often use different platforms like Windows NT and MacOS and it enables the use of very advanced computer hardware like multiprocessor servers and high performance workstations with large memory resources, which are necessary for the clustering of large datasets. The software has been tested on PCs under Windows NT and Linux as well as on a multiprocessor Sun server under Solaris without any adaptation of the code. Although Java is not as fast as native code, the time consumed for calculating a result is little on state of the art computers. Additionally, smart compilers, well-tuned interpreters, and just-in-time byte-code compilers can bring performance close to that of native code without loosing the advantage of portability.

In summary, the flexibility, the variety of analysis tools and data visualizations, as well as the transparency and portability makes this software suite a valuable tool in future functional genomic studies.

Appendix A

Default Distance Functions

For each clustering algorithm a default distance function has been specified for an easier program operation. The default similarity measurement is the measurement that gives usually the best results for a given clustering algorithm and/or is used in the papers. In almost all clustering algorithms Euclidian Distance is used, since it uses no normalization on the vectors and therefore does not manipulate the original data.

Clustering algorithm	**Default similarity distance measurement**
Hierarchical Clustering Genes	Euclidian Distance
Hierarchical Clustering Experiments	Same distance as used for genes
k-means Clustering	Euclidian Distance
Self Organizing Maps (SOM)	Euclidian Distance
PCA Genes	Covariance
PCA Experiments	Covariance
Support Vector Machines training	Average dot product
Support Vector Machines classification	Same distance as used for training

Appendix B

References

Introduction

[1] Rennie J. Bracing for the imminent. Scientific American; July 2000; 6.

[2] Watson JD, Crick FH. Molecular Structure of Nucleic Acids. A Structure for Deoxyribose Nucleic Acid. Nature 1953 April 25; Vol.171;737.

[3] Fleischmann RD, Adams MD, White O, Clayton RA, Kirkness EF, Kerlavage AR, Bult CJ, Tomb JF, Dougherty BA, Merrick JM, et al. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. Science. 1995 Jul 28;269(5223):496-512.

[4] Goffeau A, Barrell BG, Bussey H, Davis RW, Dujon B, Feldmann H, Galibert F, Hoheisel JD, Jacq C, Johnston M, Louis EJ, Mewes HW, Murakami Y, Philippsen P, Tettelin H, Oliver SG. Life with 6000 genes. Science 1996 Oct 25;274(5287):546, 563-7.

[5] The C. elegans Sequencing Consortium. Genome sequence of the nematode C. elegans: a platform for investigating biology. Science. 1998 Dec 11;282(5396):2012-8. Review.

[6] Heidelberg JF, Eisen JA, Nelson WC, Clayton RA, Gwinn ML, Dodson RJ, Haft DH, Hickey EK, Peterson JD, Umayam L, Gill SR, Nelson KE, Read TD, Tettelin H, Richardson D, Ermolaeva MD, Vamathevan J, Bass S, Qin H, Dragoi I, Sellers P, McDonald L, Utterback T, Fleishmann RD, Nierman WC, White O. DNA sequence of both chromosomes of the cholera pathogen Vibrio cholerae. Nature. 2000 Aug 3;406(6795):477-83.

[7] Adams MD, Celniker SE, Holt RA, Evans CA, Gocayne JD, Amanatides PG, Scherer SE, Li PW, Hoskins RA, Galle RF, … , Myers EW, Rubin GM, Venter JC. The genome sequence of Drosophila melanogaster. Science. 2000 Mar 24;287(5461):2185-95.

[8] Meinke DW, Cherry JM, Dean C, Rounsley SD, Koornneef M. Arabidopsis thaliana: a model plant for genome analysis. Science. 1998 Oct 23;282(5389):662, 679-82. Review.

[9] Dunham I, Shimizu N, Roe BA, Chissoe S, Hunt AR, Collins JE, Bruskiewich R, Beare DM, Clamp M, Smink LJ, Ainscough R, Almeida JP, Babbage A, Bagguley C, Bailey J, Barlow K, Bates KN, Beasley O, Bird CP, Blakey S, Bridgeman AM, Buck D, Burgess J, Burrill WD, O'Brien KP, et al. The DNA sequence of human chromosome 22. Nature. 1999 Dec 2;402(6761):489-95.

[10] Hattori M, Fujiyama A, Taylor TD, Watanabe H, Yada T, Park HS, Toyoda A, Ishii K, Totoki Y, Choi DK, Soeda E, Ohki M, Takagi T, Sakaki Y, Taudien S, Blechschmidt K, Polley A, Menzel U, Delabar J, Kumpf K, Lehmann R, Patterson D, Reichwald K, Rump A, Schillhabel M, Schudy A. The DNA sequence of human chromosome 21. The chromosome 21 mapping and sequencing consortium. Nature. 2000 May 18;405(6784):311-9.

[11] Lander ES. The new genomics: global views of biology. Science. 1996 Oct 25;274(5287):536-9.

[12] Lander ES. Array of hope. Nat Genet. 1999 Jan;21(1 Suppl):3-4.

[13] Fodor SP, Read JL, Pirrung MC, Stryer L, Lu AT, Solas D. Light-directed, spatially addressable parallel chemical synthesis. Science. 1991 Feb 15;251(4995):767-73.

[14] Schena M, Shalon D, Davis RW, Brown PO. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science. 1995 Oct 20;270(5235):467-70.

[15] DeRisi JL, Iyer VR, Brown PO. Exploring the metabolic and genetic control of gene expression on a genomic scale. Science. 1997 Oct 24;278(5338):680-6.

[16] Lockhart DJ, Winzeler EA. Genomics, gene expression and DNA arrays. Nature. 2000 Jun 15;405(6788):827-36. Review.

[17] Young RA. Biomedical discovery with DNA arrays. Cell. 2000 Jul 7;102(1):9-15. Review.

[18] Zhang MQ. Large-scale gene expression data analysis: a new challenge to computational biologists. Genome Res. 1999 Aug;9(8):681-8. Review.

[19] Southern EM. Detection of specific sequences among DNA fragments separated by gel electrophoresis. J Mol Biol. 1975 Nov 5;98(3):503-17.

[20] Hughes TR, Marton MJ, Jones AR, Roberts CJ, Stoughton R, Armour CD, Bennett HA, Coffey E, Dai H, He YD, Kidd MJ, King AM, Meyer MR, Slade D, Lum PY, Stepaniants SB, Shoemaker DD, Gachotte D, Chakraburtty K, Simon J, Bard M, Friend SH. Functional discovery via a compendium of expression profiles. Cell. 2000 Jul 7;102(1):109-26.

[21] Francis S. Collins. Microarrays and macroconsequences. Nat Genet. 1999 Jan;21(1 Suppl):2.

[22] Brazma A, Robinson A, Cameron G, Ashburner M. One-stop shop for microarray data. Nature. 2000 Feb 17;403(6771):699-700.

[23] Duggan DJ, Bittner M, Chen Y, Meltzer P, Trent JM. Expression profiling using cDNA microarrays. Nat Genet. 1999 Jan;21(1 Suppl):10-4.

[24] Ermolaeva O, Rastogi M, Pruitt KD, Schuler GD, Bittner ML, Chen Y, Simon R, Meltzer P, Trent JM, Boguski MS. Data management and analysis for gene expression arrays. Nat Genet. 1998 Sep;20(1):19-23.

[25] Bassett DE Jr, Eisen MB, Boguski MS. Gene expression informatics--it's all in your mine. Nat Genet. 1999 Jan;21(1 Suppl):51-5.

[26] Heyer LJ, Kruglyak S, Yooseph S. Exploring expression data: identification and analysis of coexpressed genes. Genome Res. 1999 Nov;9(11):1106-15.

[27] Brazma A, Vilo J. Gene expression data analysis. FEBS Lett. 2000 Aug 25;480(1):17-24.

[28] Epstein CB, Butow RA. Microarray technology - enhanced versatility, persistent challenge. Curr Opin Biotechnol. 2000 Feb;11(1):36-41. Review.

[29] Eisen MB, Spellman PT, Brown PO, Botstein D. Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci U S A. 1998 Dec 8;95(25):14863-8.

[30] Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, Eisen MB, Brown PO, Botstein D, Futcher B. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Mol Biol Cell. 1998 Dec;9(12):3273-97.

[31] Alizadeh AA, Eisen MB, Davis RE, Ma C, Lossos IS, Rosenwald A, Boldrick JC, Sabet H, Tran T, Yu X, Powell JI, Yang L, Marti GE, Moore T, Hudson J Jr, Lu L, Lewis DB, Tibshirani R, Sherlock G, Chan WC, Greiner TC, Weisenburger DD, Armitage JO, Warnke R, Staudt LM, et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature. 2000 Feb 3;403(6769):503-11.

[32] Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Natl Acad Sci U S A. 1999 Jun 8;96(12):6745-50.

[33] 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. Molecular classification of cutaneous malignant melanoma by gene expression profiling. Nature. 2000 Aug 3;406(6795):536-40.

[34] Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science. 1999 Oct 15;286(5439):531-7.

[35] Lossos IS, Alizadeh AA, Eisen MB, Chan WC, Brown PO, Botstein D, Staudt LM, Levy R. Ongoing immunoglobulin somatic mutation in germinal center B cell-like but not in activated B cell-like diffuse large cell lymphomas. Proc Natl Acad Sci U S A. 2000 Aug 29;97(18):10209-13.

[36] Perou CM, Jeffrey SS, van de Rijn M, Rees CA, Eisen MB, Ross DT, Pergamenschikov A, Williams CF, Zhu SX, Lee JC, Lashkari D, Shalon D, Brown PO, Botstein D. Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. Proc Natl Acad Sci U S A. 1999 Aug 3;96(16):9212-7.

[37] Perou CM, Sorlie T, Eisen MB, van de Rijn M, Jeffrey SS, Rees CA, Pollack JR, Ross DT, Johnsen H, Akslen LA, Fluge O, Pergamenschikov A, Williams C, Zhu SX, Lonning PE, Borresen-Dale AL, Brown PO, Botstein D. Molecular portraits of human breast tumours. Nature. 2000 Aug 17;406(6797):747-52.

[38] Ross DT, Scherf U, Eisen MB, Perou CM, Rees C, Spellman P, Iyer V, Jeffrey SS, Van de Rijn M, Waltham M, Pergamenschikov A, Lee JC, Lashkari D, Shalon D, Myers TG, Weinstein JN, Botstein D, Brown PO. Systematic variation in gene expression patterns in human cancer cell lines. Nat Genet. 2000 Mar;24(3):227-35.

[39] Scherf U, Ross DT, Waltham M, Smith LH, Lee JK, Tanabe L, Kohn KW, Reinhold WC, Myers TG, Andrews DT, Scudiero DA, Eisen MB, Sausville EA, Pommier Y, Botstein D, Brown PO, Weinstein JN. A gene expression database for the molecular pharmacology of cancer. Nat Genet. 2000 Mar;24(3):236-44.

Data adjustment and similarity distances

[40] Alter O, Brown PO, Botstein D. Singular value decomposition for genome-wide expression data processing and modeling. Proc Natl Acad Sci U S A. 2000 Aug 29;97(18):10101-10106.

[41] Michael Eisen. Cluster and TreeView Manual. Documentation to the programs Cluster and Treeview by Michael Eisen, Stanford University 1999.

[42] Claverie JM. Computational methods for the identification of differential and coordinated gene expression. Hum Mol Genet. 1999;8(10):1821-32. Review.

[43] Press W, Teukolsky SA, Vetterling WT, Flannery BP. Numerical Recipes in C. The Art of Scientific Computing. Second Edition. Cambridge University Press.

[44] Chen Y, Bittner ML, Dougherty ER. Issues associated with microarray data analysis and integration. Information supplementary to article by Michael Bittner, Jeffrey Trent and Paul Meltzer (Nature Genet. 22, 213–215; 1999).

[45] Michaels GS, Carr DB, Askenazi M, Fuhrman S, Wen X, Somogyi R. Cluster analysis and data visualization of large-scale gene expression data. Pac Symp Biocomput. 1998;:42-53.

[46] Fuhrman S, Cunningham MJ, Wen X, Zweiger G, Seilhamer JJ, Somogyi R. The application of shannon entropy in the identification of putative drug targets. Biosystems. 2000 Feb;55(1-3):5-14.

[47] Korber BT, Farber RM, Wolpert DH, Lapedes AS. Covariation of mutations in the V3 loop of human immunodeficiency virus type 1 envelope protein: an information theoretic analysis. Proc Natl Acad Sci U S A. 1993 Aug 1;90(15):7176-80.

[48] Bittner M, Meltzer P, Trent J. Data analysis and integration: of steps and arrows. Nat Genet. 1999 Jul;22(3):213-5.

[49] D'haeseleer P, Liang S, Somogyi R. Genetic network inference: from co-expression clustering to reverse engineering. Bioinformatics. 2000 Aug;16(8):707-26.

[50] Gilbert DR, Schroeder M, van Helden J. Interactive visualization and exploration of relationships between biological objects. Trends Biotechnol. 2000 Dec 1;18(12):487-494.

Hierarchical clustering

[51] Guenther Gedina, Praktische Methodenlehre Hintergrund-Material, lecture notes for "Praktische Methodenlehre", WS 1999/2000, University of Osnabrueck, Germany

[52] Doug Fisher, Optimization and Simplification of Hierarchical Clusterings, KDD-95:118-123

[53] Wen X, Fuhrman S, Michaels GS, Carr DB, Smith S, Barker JL, Somogyi R. Large-scale temporal gene expression mapping of central nervous system development. Proc Natl Acad Sci U S A. 1998 Jan 6;95(1):334-9.

Self-organizing maps

[54] Vesanto J. Usisng SOM in Data Mining. Licentiate's thesis. Helsinki University of Technology, Department of Computer Science and Engineering. 2000 Apr 10.

[55] Vesanto J. SOM-Based Data Visualization Methods. Helsinki University of Technology, Laboratory of Computer and Information Science. 1999 Nov 19.

[56] Vesanto J. Data Mining Techniques Based on the Self-Organizing Map. Thesis for degree of Master of Science in Engineering. Helsinki University of Technology, Department of Engineering Physics and Mathematics. 1997 May 26.

[57] Kohonen T, Hynninen J, Kangas J, Laaksonen J. SOM_PAK. The Self-Organizing Map Program Package. Manual Version 3.1. Helsinki University of Technology. Laboratory of Computer and Information Science. 1995 April 7.

[58] Tamayo P, Slonim D, Mesirov J, Zhu Q, Kitareewan S, Dmitrovsky E, Lander ES, Golub TR. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. Proc Natl Acad Sci U S A. 1999 Mar 16;96(6):2907-12.

k-means

[59] Zhexue Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In Proc. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, 1997.

[60] Zhexue Huang. Clustering large data sets with mixed numerical and categorical values. Proceedings of the First Pacific-Asia Conference on Knowledge Discovery and Data Mining, Singapore, Word Scientific, 1997.

[61] Tavazoie S, Hughes JD, Campbell MJ, Cho RJ, Church GM. Systematic determination of genetic network architecture. Nat Genet. 1999 Jul;22(3):281-5.

[62] Soukas A, Cohen P, Socci ND, Friedman JM. Leptin-specific patterns of gene expression in white adipose tissue. Genes Dev. 2000 Apr 15;14(8):963-80.

Principal component analysis

[63] Alexander Basilevsky, Statistical factor analysis and related methods, Theory and Applications. Wiley series in probability and mathematical statistics. John Wiley & Sons.

[64] Raychaudhuri S, Stuart JM, Altman RB. Principal components analysis to summarize microarray experiments: application to sporulation time series. Pac Symp Biocomput. 2000;:455-66.

[65] Holter NS, Mitra M, Maritan A, Cieplak M, Banavar JR, Fedoroff NV. Fundamental patterns underlying gene expression profiles: simplicity from complexity. Proc Natl Acad Sci U S A. 2000 Jul 18;97(15):8409-14.

Support Vector Machines

[66] Gaasterland T, Bekiranov S. Making the most of microarray data. Nat Genet. 2000 Mar;24(3):204-6.

[67] Brown MP, Grundy WN, Lin D, Cristianini N, Sugnet CW, Furey TS, Ares M Jr, Haussler D. Knowledge-based analysis of microarray gene expression data by using support vector machines. Proc Natl Acad Sci U S A. 2000 Jan 4;97(1):262-7.

[68] Stitson MO, Weston JAE, Gammerman A, Vovk V, Vapnik V. Theory of the Support Vector Machines. Technical Report CSD-TR-96-16. Department of Computer Science. Royal Holloway University of London. 1996 Dec 31.

[69] Gunn S. Support Vector Machines for Classification and Regression. ISIS Technical Report. Image Speech & Intelligent Systems Group. University of Southampton. 1998 May 14.

[70] Brown MPS, Grundy WN, Lin D, Cristianini N, Sugnet C, Ares Jr. M, Haussler D. Support Vector Machine Classification of Microarray Gene Expression Data. UCSC-CRL-99-09. Department of Computer Science. University of California, Santa Cruz. 1999 June 12.

Programming tools

[71] Campione M, Walrath K et al. The Java TM Tutorial Continued: The Rest of the JDK TM. The Java TM Series. Addison-Wesley Pub Co, December 1998.

[72] Gosling J, McGilton H. The Java TM Language Environment: A White Paper. Sun Microsystems www, May 1996. ftp://ftp.javasoft.com/docs/papers/langenviron-pdf.zip.

Results

[73] Iyer VR, Eisen MB, Ross DT, Schuler G, Moore T, Lee JCF, Trent JM, Staudt LM, Hudson J Jr, Boguski MS, Lashkari D, Shalon D, Botstein D, Brown PO. The transcriptional program in the response of human fibroblasts to serum. Science. 1999 Jan 1;283(5398):83-7.

[74] Chu S, DeRisi J, Eisen M, Mulholland J, Botstein D, Brown PO, Herskowitz I. The transcriptional program of sporulation in budding yeast. Science. 1998 Oct 23;282(5389):699-705.

Discussion

[75] Gilbert DR, Schroeder M, van Helden J. Interactive visualization and exploration of relationships between biological objects. Trends Biotechnol. 2000 Dec;18(12):487-94.

[76] Hastie T, Tibshirani R, Eisen MB, Alizadeh A, Levy R, Staudt L, Chan WC, Botstein D, Brown P. 'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. Genome Biology. 2000 Aug 4: 1(2):research0003.1-0003.21.

[77] Cohen BA, Mitra RD, Hughes JD, Church GM. A computational analysis of whole-genome expression data reveals chromosomal domains of gene expression. Nat Genet. 2000 Oct;26(2):183-6.