# Statistical Learning of Gene Annotations in Saccharomyces Cerevisiae

by

Miles Trochesset

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Department of Computer Science
University of Toronto

# Abstract

Statistical Learning of Gene Annotations in Saccharomyces Cerevisiae

Miles Trochesset

Master of Science

Department of Computer Science

University of Toronto

2004

This work presents an investigation of machine learning techniques which can be applied to quantitative data obtained from experiments on yeast. Our goal is to predict the function of genes, more specifically for those which currently have no known function.

We investigate data-preprocessing methods, and develop two algorithms for filling missing values in the datasets. This is a necessary step before some statistical methods can be used to predict gene function.

We present two standard machine learning algorithms, one used in a non-standard way, for predicting the biological functions of genes in a systematic and comprehensive manner. Determining gene function is simplified to a series of binary classifications and one of the challenges of this learning task lies in the extremely small number of positives, compared with large amounts of negatives samples. We develop a method based on hierarchical clustering used with labeled data to search for regions of high positive concentrations and make predictions for the unlabeled genes. We investigate logistic regression as a baseline for comparing to our technique. Both of these methods are based on different views of the data and we found that depending on the biological processes, one or the other of these approaches performs better, although our method makes more confident predictions for more biological processes.

The outcomes of the research are threefold: first we present two algorithms for missing value estimation. Second we build a new biological data mining method based on existing machine learning tools that are readily accepted in the biological community. Third we make biological predictions of gene functions, each associated with a level of confidence and all above 50% precision.

This thesis is dedicated to my mom and my grandpa.

# Acknowledgements

I would like to thank several people that have helped me with this work. Pr. Anthony Bonner for supervising my research, for meeting with me often, and for introducing me to Pr. Timothy Hughes. Pr. Sam Roweis for teaching the machine learning class which made me want to work in the field, and for meeting with me a few times. Dr. Quaid Morris for helping me with machine learning issues, sending me papers related to my work, telling me matlab tricks and giving me scripts. Hughes Lab for providing the data necessary for this research. Pr. Timothy Hughes for giving me a position in his lab and projects, for making me realize the importance of figures, and most importantly for teaching me the basics about genomics and biology. Dr. Laurent Mignet for making me switch to emacs and fvwm, and helping me with various UNIX related problems I've encountered. Pr. Marc Grynpas for helping me making decisions.

# Contents

# Chapter 1

# Introduction

The research in this thesis focuses on the area of computational biology, and more specifically computational tools applied to functional genomics. Functional genomics is the part of biology that tries to obtain higher meaning and possibly determine function for pieces of a genome. In the traditional approach, the first step in this process is to cut the entire genome of an organism into small pieces, sequence them all and map the pieces to the chromosomes to get their respective locations. Once the full sequence is assembled, the following step is to apply computational methods such as GenScan [BK97] for predicting the number and positions of genes on the chromosomes as well as their exon and intron structures. Finally with the sequence of the predicted genes, we can try to determine their function, either by experimenting on mutant cells for example, often by knocking out the gene of interest and studying the effects under different environmental conditions (this is actually the definition of a gene function), or by studying the protein encoded by the gene (in the case of protein coding genes, which are the majority). At the end, we hope to have an understanding of what role the gene product has on the cell activity and give biologically meaningful names to the putative genes generally referenced by systematic names. For example YOR143C (a systematic name referring to chromosome position) is a gene involved in thiamin biosynthesis whose product is the enzyme thiamin pyrophosphokinase and it was therefore named THI80 which is more easily remembered by biologists.

One of this thesis' goals is to investigate data mining techniques for predicting, in a systematic and comprehensive manner, the possible functions of all putative and known genes (a gene may have several biological functions) in a yeast organism called *Saccharomyces cerevisiae*. Data mining is a discipline which aims at the discovery of patterns in large volumes of data. It came to life in response to the challenges and opportunities provided by the increasing number of very large high-dimensional databases covering important areas of human activity, such as those existing in the field of biology, but it is also used extensively in industrial, social and economical activities. Data mining borrows from several long-established disciplines, among them, database technology, machine learning and statistics. Here we tried to integrate machine learning tools and adapt or develop new ones for the purpose of predicting gene function using measurements made on mutant cells.

We focused more intensely on making predictions for genes for which no biological function had yet been determined, and decided to analyze in the future the predictions for genes which had at least one known function. Systematic approaches for identifying the biological functions of genes are needed to ensure rapid progress from genome sequence to directed experimentation and applications such as drug discovery.

## 1.1 Determining gene function as a classification task

The problem of determining gene functions is simplified to a binary classification task. For each function of interest, we ask whether the gene product has that function or not. Positive examples are genes for which we know the product has the function, and vice-versa for the negative examples. This transformation allows us to use algorithms from the machine learning world to solve this task; binary classification is thoroughly studied in that field. There are two drawbacks of using this simplification. First we are now faced with many classification tasks, one for each function of interest. Last but not least, functions are now learned independently, and therefore we will lose all the information contained in the correlation between functions. Nonetheless, this is an obvious first step.

There are several sorts of gene functions available in the biology community. We used the *ontologies* defined by the Gene Ontology Consortium [GO04], which are *molecular function*, *biological process* and *cellular component*. *Biological processes* are reasonably well correlated with gene expression profiles, and they reflect the general nature of gene function, therefore these were the labels we used for learning function. We will refer to the genes whose products are known to belong to a particular biological process as positives, and to the remaining genes as negatives.

Much of the classification work using machine learning in biology has been done in cancer classification [AED$^+$00, ABN$^+$99, FCD$^+$00, KWR$^+$01, LL03, NR02, RTR$^+$01, WBD$^+$01] rather than predicting ontologies. This task is investigated in [BGL$^+$00] but only for 6 classes (which were not defined by the Gene Ontology). Our approach is designed for making prediction for any of the classes in the Gene Ontology, on the order of a thousand classes.

## 1.2 Unbalanced classes

Since relatively few genes are involved in a typical biological process, there are far more negatives than positives, as little as a single positive gene and approximately 6,300 negatives (size of the yeast genome) for some biological processes (although other processes involve up to 60% of the genes in the genome). The learning task is even harder since the samples we have comprise only about 10% of the genes in the genome (but required tremendous amounts of biological work to obtain nonetheless, more than two years of lab work), 15% of which were unlabeled. As a result the number of positives available in our samples can be extremely small for some biological processes and this constitutes the main difficulty in this study. Most algorithms available in machine learning cannot function or would be expected to lead to poor results in these circumstances.

## 1.3 Two models for the structure of the data

In order to predict gene function, we examined two different methods based on two views of the data. The first view is that the positives and negatives can be separated by a hyperplane, which we fit using logistic regression. In the second view, the data constitutes a sea of negatives with some small islands of positives of unknown size and number. We identify these concentrations of positives using hierarchical clustering on labeled data, which is not the standard unsupervised way of using this algorithm. We found that for some biological processes, one or the other method performs better, although our hierarchical method produces more confident predictions for more biological processes. Also, the method we develop allows the analysis of biological processes for which we have as few as 5 positive samples, unlike logistic regression which was unable to make predictions when the number of positives fell below 20.

We had little choice but to use leave-one-out cross validation for evaluating the learning procedures because, having so few positives in our samples, we could not afford to waste labeled data by separating it into training and test sets. Moreover, in this application, the cost associated with experimentally testing predictions justifies leave-one-out cross-validation, not only to measure how well the classifiers were behaving, but to build decision rules for classifying the unlabeled samples. This is a main point in our methodology and we will explain it in detail later.

## 1.4 Types of data used

Our analysis used two types of data, gene expression from cDNA microarrays and growth phenotype data. Whole-genome expression profiling, facilitated by the development of DNA microarrays [SSDB95, HMJ+01], represents a major advance in genome-wide functional analysis. A single assay can measure the transcriptional response of thousands of genes, and often a full genome, to a change in cellular state such as disease, cell-cycle, cell division, response to stress and chemical compounds, or genetic perturbation and mutations. Global transcriptional response of all the genes in the genome constitutes a detailed molecular phenotype and predicting whether a sample is cancerous is a possible application [AED+00, ABN+99, KWR+01, WBD+01]. However gene expression alone does not give a full picture of the cell state. Transcripts are predominantly mRNAs (messenger RNAs) which need to be translated into proteins which sometimes need to be activated and each of these steps can be regulated by the cell. Therefore more data types are needed to analyze regulation at a finer level of granularity which is one of the reasons why we chose to include several sources of phenotype data in this study.

In general, when the cellular state of two cells affect the same biological process, similar gene expression profiles are observed [HMJ+00]. For example, mutants affecting the same cellular process often display related transcript profiles, even if the mutations affect different gene products. In most cases, the global profile similarity is sufficient to cause association in a clustering analysis. As a result, clustering has been used extensively in functional genomics to analyze gene expression data [ABN+99, BDSY99, ESBB98, HMJ+00, SS00] and is what biologists use most for exploratory analysis of microarray data. This is why we have investigated clustering for making automatic predictions of gene function in chapter 5.

## 1.5 Outline

The outcomes of this research are threefold. First we study techniques that can be applied to preprocess the datasets, reduce noise, fill in missing values and reduce the dimensionality. Second we adapt and develop learning algorithms for mining our biological datasets with the intent to produce a set of predictions of gene function. This set should be easily understandable and all predictions should be testable and have some measure of confidence. It is important whenever possible to explain why the algorithms made these predictions, which helps in finding new biological insights and may improve the performance of future biological assays for discovering gene functions. We want to determine which experiments and which measurements or combination of measurements were most informative for the purpose of predicting gene function. Last we make predictions of gene function, each associated with a level of confidence and all above 50% precision, on the basis of cross-validation results.

The area of functional genomics is a challenging environment for computational methods and specific challenges are :

- to use more of the full range of biological data available, many new techniques are providing data on a genome-wide scale, from high-throughput experiments including microarray experiments, phenotypic growth experiments, protein-protein interactions, protein secondary structure predictions and sequence similarity searches. This data is noisy and techniques for dealing with that are needed; we will describe the methods we used and developed for denoising our datasets in chapter 3.

- Genes can have multiple functions and not much research has be done on machine learning algorithms for handling multiple labels for each sample.

- As we will see in Chapter 2, the biological functions we try to predict are organized in a directed acyclic graph, therefore use of hierarchical class information will be important for improving the accuracy of prediction making algorithms.

- Some genes are known to be involved in a particular biological process, but it is not often the case that we know which genes are not involved in that biological process. In fact it should be considered that only positive labels are known for certain, all negatives are possible positives and were labeled as negative only because no annotation in the biological process was available for these samples in the *Saccharomyces Genome Database*. We are faced with learning in the context of positive samples only.

- The volume of quantitative biological data now available imposes a need for scalable solutions; complexity analysis of algorithms is also very important.

- It happens that the number of features measured for each sample is orders of magnitude greater than the number of samples (e.g. in gene expression from DNA microarrays there can be up to tens of thousands of probes on each array, each corresponding to a measurement, but the number of samples rarely exceeds a hundred). This is a special case of high dimensional data and it requires dimensionality reduction techniques.

- Running experiments is costly. Therefore biologists are not interested in the global accuracy of a prediction set, but are in fact more interested in prediction sets where predicted positives are indeed true positives. Predicted negatives have almost no utility since experiments cannot confirm them. Moreover, since relatively few predictions can be verified experimentally, it is important for the prediction sets to be ordered, with the proportion of true positives being very high in the top of the list. It is not so important that the classifiers are accurate at the bottom of the list since the corresponding predictions will not be tested.

The organization of this thesis is as follows:

- An overview of the datasets used in this research is given in Chapter 2.

- Chapter 3 describes the data pre-processing steps applied to these datasets, including cDNA microarray normalization, imputing missing values and dimensionality reduction.

- In Chapter 4 we use logistic regression to make some predictions of gene function. We investigate the challenge of having very few positive samples, and based upon the cost of making wrong predictions, we use cross validation for designing customized decision rules.

- Chapter 5 introduces a novel algorithm based on hierarchical clustering for making predictions of gene functions.

- Finally we draw conclusions about this work and present ideas for future work.

# Chapter 2

# An overview of the data

The data used in this thesis was gathered at Hughes Lab at the Banting and Best Department of Medical Research in the University of Toronto. Nearly 20% (1,105) of the approximate 5,800 protein-coding genes of the budding yeast *Saccharomyces cerevisiae* are required for viability (under standard laboratory conditions, growth at 30 degrees in rich medium with glucose as the carbon source), hindering genetic analysis with knockouts. The precise molecular and genetic functions of many essential yeast proteins have not been studied in detail, because it is difficult to study essential genes using deletion mutants.

Transcription, splicing, ribosome biosynthesis, translation, cell wall and membrane biogenesis, DNA replication, nuclear transport, and basic cytoskeletal functions are all required for cell proliferation, and genes involved in these processes tend to be essential [MDH$^+$04]. Specific aspects of other cellular functions are also required for viability.

Essential genes are generally more highly conserved in humans; 38% of essential yeast proteins have counterparts in humans, versus 20% for nonessential genes (with a Blastp E value of E-50) [Hug02].

In order to investigate the functions of these genes, the Hughes Lab constructed promoter-shutoff strains for over two thirds of the essential genes and subjected them to size profiling, morphological analysis, drug sensitivity screening and gene expression profiling. These data were used to ask which phenotypic features characterized different functional classes and also to infer potential functions for uncharacterized genes. In this thesis, we focus on inferring potential functions. The analysis of how phenotypic features characterized different functions can be found in [MDH+04].

## 2.1 Analysis of essential genes through promoter-shutoff strains

Several techniques can be used to perform genetic analysis on essential genes. The Hughes Lab used tetracycline-regulatable promoters (TET) [GPAH97] to create mutant alleles of essential genes. The promoter region of a gene is upstream of the protein coding region, and therefore a major advantage of the promoter replacement technique is that the native open reading frame (ORF) is conserved. With the TET system, expression of a mutant gene is controlled by adding the drug doxycycline to the growth medium, which has little effect on the yeast physiology. The concentration of doxycycline affects the rate of transcription of genes whose promoters have been replaced with the TET version.

Hughes Lab has created TET-mutants for two thirds of the essential genes (602) which allows direct experimentation on these genes. Each constructed strain is mutated for exactly one essential gene, so there is a one-to-one correspondence between strains and genes. Next we briefly describe the individual datasets used in this thesis, all available from [MNA].

## 2.2 Phenotypic data

### 2.2.1 Cell size distributions

When growing a colony of a particular mutant, all cells in the colony are not the same size, but the distribution of sizes is informative. This dataset consists of the distributions of cell sizes for 591 of the 602 mutant strains (sampled at 256 sizes). Strains were grown by batches. Normalization is necessary as there are variations in the size of cells grown on different batches (due to the amount of time cells where allowed to grow in each batch, or the media used for growth etc…). We normalized the batches by equalizing the median (over all mutants in a specific batch) of the distristributions' medians (median of the distribution of a specific mutant colony). Validation of this normalization was done by verifying that the distribution of control wild-type strains grown in all batches coincided. The dimensionality was reduced from 256 to 8 by PCA.

### 2.2.2 Drug Sensitivity

The sensitivity of the mutant strains to different chemical compounds in 27 experimental conditions is measured. 685 samples, corresponding to 585 mutant strains with replicates, were grown on plates with one drug and the size of the colonies were compared to wild-type grown with the same drug. The value reported in the dataset is the log P-value that a difference existed between the two groups.

### 2.2.3 Cell morphologies

These measurements represent the morphological features of the mutant cells which were visually inspected for 17 different characteristics such as elongated, budded or pointed cells. This data is the only type which is categorical. A 1 indicates that the feature was slightly observed for all mutant cells, a 0 indicates it was not. On rare occasion other types appear, 0.5 means the feature was slightly observed but the phenotype was not penetrant, 2 moderately observed for all cells, 2.5 moderately observed but the phenotype was not penetrant, 3 severely observed for all cells.

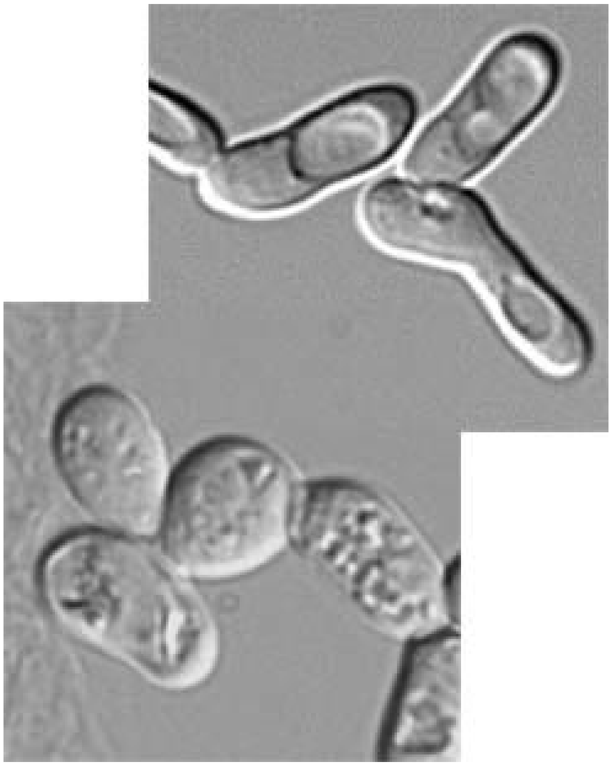

Figure 2.1: Example of cell morphologies

## 2.3 Gene expression data from cDNA microarrays

DNA microarrays is a recent technology which is used for providing an estimate of the abundance of chosen RNA transcripts in a sample. It is often the case that a single microarray tests for the expression of all the genes in the genome of an organism. Typically there are between 5,000 and 20,000 spots on a microarray, each one measuring the abundance of a particular transcript. Most microarrays are designed to contain one measurment per gene, but this is not necessarily the case.

There are two stages in using this technology, designing the microarray and then hybridizing samples. The first stage consists of choosing which transcripts will be measured and for each one, designing a short oligonucleotide (typically of length 20 to 60 nucleotides), or constructing long complementary DNA (cDNA) strands, and finally printing all spots on the array. The second stage consists of collecting the samples (e.g. collection of tissues, collection of mutant cells) then labeling the RNA in the samples with dyes and hybridizing to the arrays. At this stage, the RNA in the samples will bind to the corresponding oligonucleotides or cDNAs (depending on the technology used). The abundance of hybridization to each spot can be measured by scanning (this is called quantification). The dyes in bound RNAs become fluorescent when lit by a laser. We then have an estimate of the abundance of all the RNA in each sample, which can be used to infer the cellular state. Usually two samples are hybridized to each microarray, each sample being marked with a different dye. The expression of genes in one sample are compared to the expressions in the other sample. These two-channel microarrays are the most common type of gene expression measurements.

Here we used two-channel cDNA microarrays to measure the abundance of transcripts of the mutant cells relative to the wild-type strain for the entire genome (approximately 6,300 spots on each array). Each experiment was done twice, with dye swapping in the second experiment and then gene expression was averaged. The 292 samples, corresponding to 218 essential genes with replicates for a couple of the genes. So out of the 602 constructed mutants, the Hughes Lab has gene expression measurements for 218 mutants only. This dataset is publicly available on NCBI Gene Expression Omnibus (GEO) [GEO] accessible through the identifier GSE1404. After quantification, hybridized samples were normalized using background subtraction, followed by a LOWESS smoother to correct for dye discrepancies and by a high-pass filter to remove any sorts of spatial artifacts ( scratches, dust, gradient across the array or red corners ...) After investigating several techniques for imputing missing values, which we describe in chapter 3, we used BPCAfill [OST$^+$03], which seemed to perform the best on simulated datasets with the same proportion of missing data (approximately 13%). In the end the dimensionality of the data was reduced from 6,307 to 20 using principal components analysis (PCA)

[HTF01, RSA00] by selecting the eigenvectors associated with the 20 largest eigenvalues of the covariance matrix.

Each dataset covers a different set of the 602 constructed mutants but these sets intersect, therefore a simple solution was to use these datasets independently.

## 2.4 Labels

### 2.4.1 Gene ontologies

The GO consortium defined biological functions which are called *ontologies*. All biological functions are assembled into a hierarchy called the *Gene Ontology* [GO04]. In this hierarchy, functions are organized in a directed acyclic graph (DAG), a function can therefore have several parents and children. As we move down into the hierarchy, functions become more specific and vice-versa, functions higher in the hierarchy are more general. There are three types of gene ontologies: *GO biological process*, *GO molecular function* and *GO cellular component*. Molecular functions describe very specific chemical properties of the gene product, such as binding to a specific compound. Biological processes are more general, they constitute activities which can require several steps and involve many compounds. Finally, cellular component describes where the gene product is found in the organism, it can be a location in the cell (e.g. cytoplasm), or the type of cell (e.g. germ cells).

Each of these types group together several thousands of functions. For example the functions *DNA replication initiation [GO:0006270]*, *peptidyl-tryptophan bromination [GO:0018080]* are GO biological processes. The functions *microtubule plus-end binding [GO:0051010]* and *iron superoxyde dismutase activity [GO:0008382]* are GO molecular functions. Finally, *rough endoplasmic reticulum membrane [GO:0030867]* and *female germ cell nucleus [GO:001674]* are GO cellular components.

The version of the Gene Ontology we used (which has since been updated) contained

1484 biological processes. Some were very broad and general such as *protein metabolism [GO:0019538]* or *cell organization and biogenesis [GO:0016043]*. Top-level (high in the GO hierarchy) categories are often not specific enough to verify experimentally. One way to restrict this study to the more specific ones was to eliminate biological process involving more than 600 genes (10% of all yeast genes). This number was manually chosen so as to include the interesting categories and eliminate the general ones.

### 2.4.2 Gene annotations

Although the gene ontologies are independent of the organism and do not provide functional information for any gene, they provide a basis for studying the functions of genes of any organism. The actual knowledge comes in the form of gene annotations and, for a few dozen organisms, there are public databases from which anyone can download the current knowledge. We downloaded the yeast gene annotations from SGD, the *Saccharomyces Genome Database* [CWB$^+$04, DHD$^+$02].

We call *unlabeled*, genes for which SGD did not provide any annotation and *labeled*, genes for which there was at least one annotation. Labeled genes are therefore involved in at least one biological process. Unlabeled genes represent almost 40% of the genome of *Saccharomyces Cerevisiae*. We also call *positives*, the set of genes involved in a particular biological process, and *negatives*, the set of labeled genes which are not involved in it. Notice that the negatives do not intersect with the unlabeled genes.

One point to be noted here is that the annotations only provide knowledge of positives. The negatives are not known for certain since they are defined in each biological process as the genes which are neither positive nor unlabeled. The labels have some noise, because some of these negatives are in fact positives. Nevertheless it is believed that these mislabeled genes are rare. This is supported by the fact that most biological processes involve very few genes, so most negatives are indeed real negatives. On the other hand there should be extremely low noise in the positive labels since these are manually curated by experts and a positive annotation is always the result of experimentation.

### 2.4.3 Some statistics about the current knowledge on yeast

Out of the 6271 open reading frames (ORF), 2326 were unlabeled in SGD (Saccharomyces Genome Database), which represents 37% of the total. This proportion shows how little we know about the cellular functions of the yeast genes. Much is still to be discovered. Even for such a simple unicellular organism, we need to discover the functions of over a third of the genome.

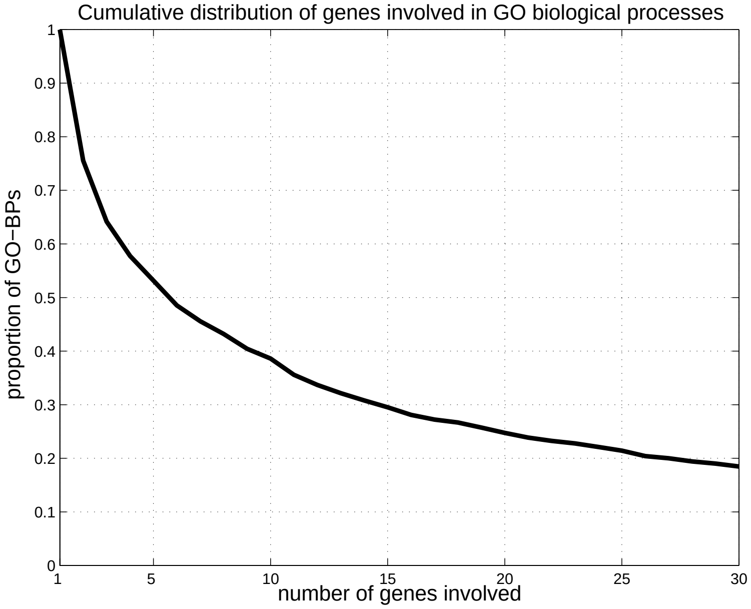

Figure 2.2: Cumulative distribution of the number of genes involved in biological processes

From the 602 mutants constructed at the Hughes Lab, 90 were unlabeled, which represents 15%. These are the genes for which our research tries to determine one or more potential functions. We are left with 512 labeled genes to be used as training samples for the learning algorithms, some of which will be positives but most of which will be negatives in most biological processes.

If we analyze the GO biological processes in detail, we notice that a quarter involve fewer than three genes and half involve fewer than 5 genes. This reflects the extremely high abundance of negatives compared with very few positives. Recall that the genome of the budding yeast has roughly 6,300 genes. On the other hand, a quarter of the GO biological processes involve more than 20 genes so there are some relatively broad categories in the gene ontology. The cumulative distribution of the number of genes involved in biological processes is represented in figure 2.2. The vertical axis represents the proportion of biological processes involving more than $x$ genes.

Investigating the genes in further detail, we notice that 1% (approximately 60 genes) are involved in more than 40 GO biological processes, 10% in more than 24 GO biological processes, half are involved in more than 11 biological processes. A histogram showing the number of categories a gene is involved in is given in figure 2.3. This illustrates how a gene can have several cellular functions (actually the number of distinct functions is smaller since the SGD annotations were propagated up the GO hierarchy), and why this problem is not a straightforward machine learning classification task.

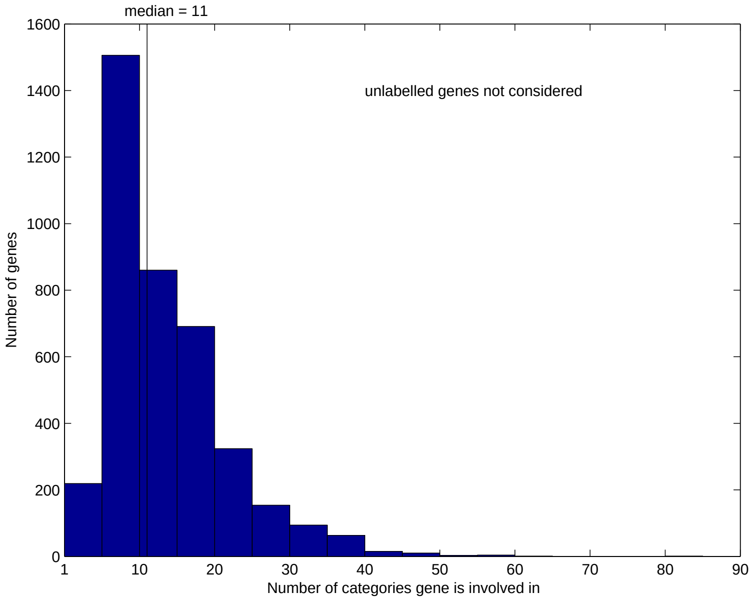

Figure 2.3: Histogram of the number of biological processes genes are involved in

To summarize these statistics about the current knowledge available for *Saccharomyces cerevisiae*, more than a third of the genome, and 90 of our samples are unlabeled. Roughly half of the biological processes involve fewer than five positives. This illustrates how complicated functional genomics is and how little we know so far. It also reminds us that making accurate predictions is a tough challenge, having so few positives in each biological process to learn from, without mentioning the noise in the labels.

# Chapter 3

# Data Preprocessing

## 3.1 Imputing missing values

The presence of incomplete data is a commonplace occurrence in large real-world databases. Missing values can occur for a variety of reasons, especially in gene expression data. When scanning DNA arrays, dust, scratches or fingerprints can blur the images and biologists usually manually flag spots on their arrays when they detect anomalies [AED$^+$00]. During normalization of the data in the arrays, background subtraction is often used, which leads to missing values when the background intensity is greater than the foreground intensity of a spot. This would typically happen if the gene in a spot is not expressed at all. Missing values can happen as a result of several other factors such as printing of the spots,poor or uneven hybridization, or poor dye labeling of the RNA, etc...

Many algorithms used for supervised or unsupervised learning, like classification, regression, or clustering, are not robust to the presence of even a few missing values. For example algorithms like K-nearest neighbors, K-means or hierarchical clustering require the computation of distances or similarities between pairs of elements or between cluster centers and every element. These cannot be computed as soon as one the two elements has a missing value in one of it's attributes. Therefore we need methods for completing datasets with missing values.

In this chapter we will see which techniques we have tried for filling the missing values present in the Hughes gene expression data. We will also look at error estimation which is needed for measuring the performance of the different schemes.

### 3.1.1 Removing incomplete samples

Several approaches can be taken when one needs a complete dataset. The most obvious one is to remove all samples that have missing values for one or more of their attributes. In our context we can rule out this solution because of dimensionality reasons. As it is often the case in high-throughput experiments such as cDNA arrays, the number of samples is much lower than the number of features measured for each sample. For example the TET_Promoter yeast project at Hughes Lab has around 200 arrays with roughly 6300 spots. Learning algorithms need more samples as the dimensionality increases. A rule of thumb is to try to have 10 samples per dimension to be learned. Therefore in our case, we need to keep incomplete samples.

Our aim is to describe a collection of techniques that are more generally appropriate than the complete-case analysis when missing entries in the data set mask underlying values.

### 3.1.2 Replacing missing values by a constant

A widely used solution is to replace all missing entries by a constant. For gene expression data this constant is often taken to be zero (*i.e.* no change) because it is generally assumed that the expression of most genes is not changed in most experiments, therefore the log transformed ratio of expressions is zero with high probability [AED$^+$00].

### 3.1.3 Using the average

A simple solution that is more acceptable is to replace a missing attribute by the sample mean for that attribute.

#### Gene sample mean

The most obvious way of applying this to expression data is to consider each array as a sample in which the genes are attributes. Unfortunately gene average assumes the expression of a gene in an experiment is similar to the expression of that gene in all the experiments which is often not true.

#### Array sample mean

Another way of using the sample mean to fill in missing values in the case of gene expression data is to consider each gene as a sample in which arrays are different measurements or attributes. This makes sense because for example, if most genes are up-regulated in an array, there are reasons to believe the expression of one missing gene is also up regulated. But in general the expression of a gene in an array is not similar to the expression of all other genes in that array and this method performs poorly [TCS$^+$01] and gene sample mean is a better solution.

### 3.1.4 Computing Pearson's correlation when values are missing

Replacing missing values by a constant or by a sample mean is not optimal because these methods do not take into consideration the correlation structure of the data.

Although little work has been done on estimating missing values for microarray data, this problem has been widely studied in other fields [TCS$^+$01]. Most commonly applied statistical techniques for dealing with missing data are model-based approaches. We tried to focus on techniques that had minimal specific modeling assumptions. It would be interesting to build a model for the generation of missing values in microarray data.

It is already known that dark spots are more likely to induce missing values, for example, because of background subtraction or because a nearby bright spot might have a comet and prevent a correct scanning of the dark spot.

Next we will try to use more of the correlation available in the data in order to impute missing elements. The correlation of 2 random variables $X$ and $Y$ for which we have $N$ samples $(X_i, Y_i)$ is given by

$$r = \frac{\sum X_i Y_i - \frac{\sum X_i \sum Y_i}{N}}{\sqrt{(\sum X_i^2 - \frac{(\sum X_i)^2}{N})(\sum Y_i^2 - \frac{(\sum Y_i)^2}{N})}} \tag{3.1}$$

$$r = \frac{\sum z_{X_i} z_{Y_i}}{N} \qquad \text{where} \begin{cases} z_{X_i} = \dfrac{X_i - \mu_X}{\sigma_X} \\ z_{Y_i} = \dfrac{Y_i - \mu_Y}{\sigma_Y} \end{cases} \tag{3.2}$$

$\mu$ and $\sigma$ being the mean and standard deviation of the random variables. In the case of complete data, we could approximate $r$ by

$$r' = \frac{\sum \frac{X_i - \bar{X}}{S_X} \frac{Y_i - \bar{Y}}{S_Y}}{N} \tag{3.3}$$

where $\bar{X}$ and $\bar{Y}$ are the sample means and $S_X^2$ and $S_Y^2$ are the sample variance of the random variables, i.e.

$$\begin{aligned} \bar{X} &= \frac{\sum X_i}{N} & \qquad S_X^2 &= \frac{1}{N-1} \sum (X_i - \bar{X})^2 \\ \bar{Y} &= \frac{\sum Y_i}{N} & \qquad S_Y^2 &= \frac{1}{N-1} \sum (Y_i - \bar{Y})^2 \end{aligned}$$

In the case of incomplete datasets we approximated $r'$ by

$$r'' = \frac{\sum_{\forall i \text{ s.t}(X_i, Y_i) \in \mathfrak{X}\mathfrak{Y}} \frac{X_i - \bar{X}'}{S'_X} \frac{Y_i - \bar{Y}'}{S'_Y}}{|\mathfrak{X}\mathfrak{Y}|} \tag{3.4}$$

where $\bar{X}'$ and $\bar{Y}'$ are the approximations of the sample means and $S'_X$ and $S'_Y$ are approximations of the standard deviations of the random variables, i.e.

$$\bar{X}' = \frac{\sum_{X_i \in \mathfrak{X}} X_i}{|\mathfrak{X}|} \qquad S'_X = \frac{1}{|\mathfrak{X}| - 1} \sum (X_i - \bar{X}')^2$$

$$\bar{Y}' = \frac{\sum_{Y_i \in \mathfrak{Y}} Y_i}{|\mathfrak{Y}|} \qquad S'_Y = \frac{1}{|\mathfrak{Y}| - 1} \sum (Y_i - \bar{Y}')^2$$

and $\mathfrak{X}$ (and $\mathfrak{Y}$) is the set of all available measurements $X_i$ of $X$ (resp. of $Y_i$ of $Y$) and $\mathfrak{X}\mathfrak{Y}$ is the set of all available couples $(X_i, Y_i)$.

### 3.1.5 K-nearest neighbors for imputing missing data

In this setting we consider each sample to be a point in a D-dimensional space, D being the number of attributes of the samples.

The idea behind $k$-NN is to find $k$ samples that are close to the sample S which is missing attribute A. The distances are evaluated considering only the available coordinates. The estimate of the missing value of sample S is chosen as the average attribute A of these $k$ neighbors weighted by their similarity to the sample S. One problem is finding a distance metric that suits the type of data being analyzed. Another one is to find $k$. The latter can be done by using cross validation. Several distance metrics can be used, such as Pearson's correlation, Spearman's Rank Correlation, or the Euclidean Distance. It is generally accepted that any of these measures can be appropriate in the case of gene expression data, biologists often use them interchangeably on a case by case basis.

The algorithm just described is exactly the principle used in KNNimpute [TCS$^+$01]. In this program, samples are the genes and attributes are their expressions in the available arrays. Doing the reverse, i.e. using the experiments as samples and the genes as attributes measured for each sample, although this would seem natural, would not be appropriate in this setting. This is because there are many more genes than experiments and therefore $k$-NN would be done in a higher dimensional space with fewer samples, which would not be statistically optimal. According to [TCS$^+$01] Euclidean distance is a sufficiently accurate norm and is used in KNNimpute.

This algorithm is in fact similar to a local linear regression, where local here is only an estimation since distances are computed based on available coordinates only.

### 3.1.6 Least squares regression, the complete case

In the least squares regression setting, we assume the missing attribute $A_j$ is a linear combination of the other attributes $\mathcal{A}\backslash A_j$. In the case of complete data, when a new sample arrives with missing attribute $A_j$, the value $y_j$ of this attribute can be estimated by

$$y_j = \sum_{x_l \in \mathcal{A}\backslash A_j} \beta_l x_l \tag{3.5}$$

where the coefficients $\beta_l$ have been fitted using least squares regression over the training samples that are complete.

### 3.1.7 HLS: heuristic least squares regression

Least squares regression without modifications is not suitable for incomplete datasets because if we need all other attributes to estimate one missing attribute, in general there are not many samples left in the dataset to fit the coefficients $\beta$s, or no samples at all. Therefore we have created a modified version of this model called Heuristic Least Squares regression (HLS).

In the case of a complete dataset, when a new sample arrives with missing attribute $A_j$, instead of expressing $A_j$ as a linear combination of all other attributes, we assume it can be expressed reasonably well as a linear combination of $K$ attributes which are well correlated to $A_j$. This model has the advantage of requiring fewer parameters (the $\beta$s).

But it also necessitates the computation of correlation between attributes, and a choice for $K$.

For an incomplete dataset, we first compute the correlation between all pairs of attributes. For missing attribute $A_j$ in sample $i$, we find the $K$ attributes with highest correlation to $A_j$. Then we subset the attributes of our dataset to keep only these $K$ as well as $A_j$. It is then generally possible to find a non-empty subset of complete samples in this reduced dataset. We then do least squares regression of all $K$ attributes against $A_j$ and finally impute $A_j$ in sample $i$.

```
For all pairs of attributes:
- compute correlation of attributes i and j in C(i,j)
- For all missing values:
  * find K closely correlated attributes to the one missing
  * find complete subset of samples on these K+1 attributes
  * fit a regression model using least squares
  * estimate the missing value
```

Figure 3.1: Details of the HLS algorithm

### 3.1.8 ILS: iterative least square regressions

After comparing HLS to third party algorithms, it became a challenge to try to outperform their performances or if not, improve our own. So we developed an iterative algorithm to estimate missing values. The initialization step consists ofs replacing missing values with the sample mean (fast) or with HLS' output (slower). There are two versions of the iterative step, an online version and a batch version, which are described below. We need to define a termination criterion. We choose to terminate when none of the recomputed values differ from the former by more than a user-defined ratio.

#### Online version

After the initialization step is complete, we keep track of the positions of the missing values and re-estimate each one individually using least squares regression on the completed dataset, replacing the entries as they are computed.

#### Batch version

Another possibility for the iterative step is to keep track of the positions of the missing values, and during the $i^{th}$ step of the iterative algorithm, recompute all of them using the completed dataset obtained at the $(i-1)^{th}$ step , store these values until the end of the $i^th$ step, and replace them at the end of the $i^{th}$ step so they become available for computation only at step i+1.

## 3.2 Measuring performance

### 3.2.1 Error estimation

To measure the accuracy of a missing value estimator, the classic approach is to randomly remove some data from the incomplete dataset, estimate the missing values and measure the predicting error. An inconvenience of this approach is that we are removing values from a dataset which is already incomplete. Therefore our estimator is trained on a dataset which is sparser than the original one, fewer values are available to compute the estimations of the missing ones, and we expect the estimation accuracy to be lower than if we had trained on the original dataset. Consequently the error will be an upper bound on the error of the estimator applied to the original dataset.

Another approach is to remove values (usually at random) from a complete dataset such that the proportion of missing values is similar to the proportion of missing values in the original dataset, estimate the missing values and measure the estimation performance on that dataset. A drawback of this approach is that we might not possess a complete dataset with similar properties of the data, that is, with a similar joint probability distribution over all attributes. Another drawback of this approach is that the locations of the missing values in the original dataset might not be random.

The error function we have used for this study is the normalized root mean squared error:

$$NRMSE = \sqrt{\frac{E[(\hat{y} - y)^2]}{var(y)}} \tag{3.6}$$

When the estimator is good, $\hat{y}$ is close to the true value $y$ and NRMSE is close to 0. When the estimator $\hat{y}$ is set to be the sample mean $\bar{y}$, the expected NRMSE is 1. This measure gives an idea on the performance of an estimator and allows comparisons between estimators.

### 3.2.2 Results on simulated data

Fifty datasets were generated, each consisting of 500 samples drawn from a ten dimensional Gaussian distribution with 10% missing values uniformly located. For each dataset, the covariance matrix of the Gaussian was changed. To test the performance of several algorithms in estimating missing values in such conditions, we removed another 2% of the data at random which we used to test the estimations by computing the normalized root mean squared error (NRMSE).

The performance of six predictors were tested: a random guess with Gaussian distribution with same mean and variance as the attribute being estimated, a constant guess of zero (the mean of the Gaussian simulated data), predicting the sample mean, using the heuristic least squares regression (HLS) estimates, replacing missing values by the BPCAfill estimates and finally using the iterative least squares (ILS) estimates (the online version with sample mean initialization and a 2% stop criterion).

The normalized root mean squared error was computed for all fifty datasets and all six estimators and the median, mean, minimum and maximum normalized root mean squared errors (NRMSE) are reported in table 3.1.

| Missing Value Estimates | Median | Mean | Min | Max |
|---|---|---|---|---|
| Random | 2.2474 | 0.2632 | 1.7347 | 2.8564 |
| Zero | 1.4308 | 1.4341 | 0.8456 | 2.2045 |
| Sample Mean | 0.7259 | 0.7352 | 0.4745 | 1.0801 |
| Heuristic Least Squares (HLS) | 0.3455 | 0.3451 | 0.2095 | 0.4950 |
| BPCAfill | 0.2180 | 0.2273 | 0.0733 | 0.3999 |
| Iterated Least Squares (ILS) | **0.2157** | **0.2201** | 0.1114 | **0.3964** |

Table 3.1: Normalized Root Mean Squared Error on Simulated Data

The results from table 3.1 show that our algorithm HLS performs almost as well as BPCAfill, which we found to be the best publicly available algorithm, although HLS is much faster. Moreover, our algorithm ILS performs better (results shown in bold) in terms of median and mean NRMSE, also it's maximum error over the fifty datasets was smaller than the one of BPCAfill.

It would now be interesting to simulate datasets of the same dimension as those found in gene expression databases and see what the results are. We expect the time gain to be extremely important in higher dimensions but we do not know if the performance would still be as good.

# Chapter 4

# Logistic Regression for predicting gene function

## 4.1 Introduction

The classes we learn here are the biological processes defined by the Gene Ontology Consortium [GO04]. Each gene can be involved in several biological processes and therefore, this is not the classical machine learning approach in which samples can belong to one class only. And, as described in Chapter 2 several genes can be involved in a biological process.

In this work we learn each biological process independently, using a supervised learning algorithm for classification called logistic regression. In this setting, the problem becomes a standard classification exercise, in which we need to discriminate between two classes for each biological process: either a gene is involved in a biological process, or it is not.

The labels were downloaded from the Saccharomyces Genome Database [DHD+02, CWB+04] for all the biological processes (GO-BP). We trained the classifiers on 3 different datasets (cell size distributions, gene expression profiles, drug sensitivity) using the labeled genes for each GO biological process, and then made predictions for the unlabeled genes.

In this chapter we examine the case where the two classes are separable by a hyperplane. This is a strict assumption about the data, but it leads to predictions with high level of confidence for some biological processes nonetheless and represents a baseline for comparing the results obtained with the second view which we describe in the next chapter. We choose to fit the hyperplane using logistic regression [HTF01] because of its simplicity and also because it is well understood, and accepted in the biology community [BCRC04]. In 4.3 we investigate a method by which we can easily build decision rules customized to a particular biological process for classifying samples, precision being the only user-defined parameter. We apply these decision rules to the unlabeled samples in 4.4.

## 4.2 Logistic regression

### 4.2.1 The model

In logistic regression for binary classification, the class label Y is a Bernoulli random variable and the input X is a p-dimensional random vector. The model arises from the desire to write the log of the odds as a linear function in $x$, while ensuring that the conditional probabilities $p(y|x)$ of both classes sum to one and remain in $[0,1]$.

$$log\frac{p(Y=0|X=x)}{p(Y=1|X=x)} = \theta^T x \tag{4.1}$$

Here $\theta$ can include a constant (bias) term if we augment $x$ to a $(p+1)$-dimensional vector by stacking a 1 to it. This leads to the following conditional probabilities:

$$p(Y=0|X=x) = \frac{e^{\theta^T x}}{1+e^{\theta^T x}} \tag{4.2}$$

$$p(Y=1|X=x) = \frac{1}{1+e^{\theta^T x}} \tag{4.3}$$

By defining $\mu(x) = P(Y=1|X=x)$, we can write the Bernoulli conditional probability

$$p(y|x) = \mu(x)^y(1-\mu(x))^{1-y} \tag{4.4}$$

See [HTF01] for further details on the logistic regression.

### 4.2.2 Fitting the parameters to the data

Based on a training set $\mathcal{D} = \{(x_n, y_n), n \in \{1..N\}\}$, we fit the parameter $\theta$ by maximum likelihood. Assuming the samples are independent and identically distributed (iid), the log likelihood is

$$\begin{aligned} l(\theta|\mathcal{D}) =& \log p(y_1,..,y_N|x_1,..,x_N,\theta) && (4.5)\\ =& \log\prod_{n=1}^{N}\mu(x_n)^{y_n}(1-\mu(x_n))^{1-y_n} && (4.6)\\ =& \sum_{n=1}^{N} y_n\log\mu(x_n)+(1-y_n)\log(1-\mu(x_n)) && (4.7)\end{aligned}$$

We set the derivative of the log likelihood to zero

$$\frac{\partial l}{\partial \theta} = \frac{\partial l}{\partial \mu}\frac{\partial \mu}{\partial \theta} \tag{4.8}$$

$$\frac{\partial l}{\partial \theta} = \sum_{n=1}^{N} x_n(y_n - \mu(x_n)) = 0 \tag{4.9}$$

This is a set of $p + 1$ equations. We used Newton-Raphson's method to find an approximate solution for $\theta$. This requires the computation of the second derivative of the likelihood with respect to $\theta$. Another possibility is to do gradient descent to find the minimum of the negative log likelihood.

$$\frac{\partial^2 l}{\partial \theta \partial \theta^T} = -\sum_{n=1}^{N} x_n x_n^T \mu(x_n)(1 - \mu(x_n)) \tag{4.10}$$

$\theta$ is initialized to zero and then updated by the Newton-Raphson's step:

$$\theta^{new} = \theta^{old} - \left(\frac{\partial^2 l}{\partial \theta \partial \theta^T}\right)^{-1} \frac{\partial l}{\partial \theta} \tag{4.11}$$

where the derivatives are evaluated at $\theta^{old}$.

## 4.3 Cross validation for customized decision rules

We trained a logistic regression classifier by leave-one-out cross-validation on the labeled samples of each of the biological processes we chose to learn. Each time we computed the posteriors $P(Y = 1|X = x)$ where $x$ was the sample set aside, Y denotes the class label (which takes the value 1 if a gene is involved in the biological process, 0 otherwise). We had little choice but to use leave-one-out cross-validation because, having so few positives in our samples, we could not afford to waste labeled data by separating it into training and test sets.

In order to classify a sample we need to build a decision rule. One very simple rule could be to classify as positive any sample for which the posterior probability is above 0.5. Here we are faced with a decision making problem which needs a little more attention because of the cost associated with making false predictions. In molecular biology, running experiments is very expensive and we want to be very confident about the prediction being true before testing it in wet lab. All the cost of decisions is biased toward false predicted positives in this application, and false negatives are not given as much importance. In other words the cost of false positives is much bigger than the cost of false negatives. As a result, to increase our confidence on the predicted positives, we computed conservative thresholds for discriminating between the classes, each depending on the particular biological process. A sample will be classified as positive if it's posterior is above that threshold $P(Y = 1|X = x) > t$.

In the logistic regression setting, the classes are separated by the hyperplane defined by the equation $\theta^T x = 0$. When the input $x$ is on the hyperplane,

$$P(Y = 1|X = x) = P(Y = 0|X = x) = 0.5 \tag{4.12}$$

Raising the threshold, corresponds to translating that hyperplane in the direction of $\theta$ (or $-\theta$). Our procedure consists of translating the hyperplane toward the positive samples until the ratio of true positives to false positives is sufficiently high. Therefore we use cross validation, not only to measure how well the classifiers are performing, but really to build decision rules for classifying the unlabeled samples.

The measure of satisfaction we used for translating the hyperplane is precision, which is the ratio of true positives with respect to the number of predicted positives, i.e.

$$\text{precision} = \frac{\text{true positives}}{\text{predicted positives}} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.13}$$

Predicted negatives cannot be confirmed experimentally, at least at the Hughes Lab, which is providing us with the data, so knowledge is gained only when predicted positives are confirmed, and it is indeed precision biologists are interested in and not overall classification performance.

For a particular biological process, one approach could be to choose the threshold that leads to the maximum precision computed using all labeled samples, but we prefer to take a more conservative approach by setting a user-defined precision. That way predictions will only be made for biological processes for which the classifier reaches that precision at some threshold. For biological processes for which logistic regression performed poorly, the classifier is rejected and no predictions are made. Precision is not a monotonic function in $t$ (as $t$ decreases, the number of predicted positive increases, these can be either true or false positives), therefore we chose the lowest threshold leading to the desired precision since this solution maximizes the recall (also known as sensitivity in the signal processing and biological worlds), which is the percentage of positives which are predicted as positives:

$$\text{recall} = \frac{\text{true positives}}{\text{all positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.14}$$

We computed five thresholds for each GO biological process, corresponding to precision levels of 100%, 85%, 75%, 60% and 50% based on the labeled data. The precision level used to classify a sample, along with the distance of that sample to the translated hyperplane leads to different confidence levels.

We ought to be a bit careful about the computation of precision. Without restrictions, there is no way to tell whether a classifier made a few predictions at very high precision by chance or not. It could be that by chance, the labeled gene with greatest discriminant value, during the leave-one-out cross-validation, is a positive sample. Therefore, if we pick that discriminant as the threshold, even if the next few highest discriminant values all correspond to negative samples, we have a precision of 100% based on one predicted positive on the labeled data. Likewise, although the precision would be 50%, how significant would be one true positive out of two predicted positives? To increase the

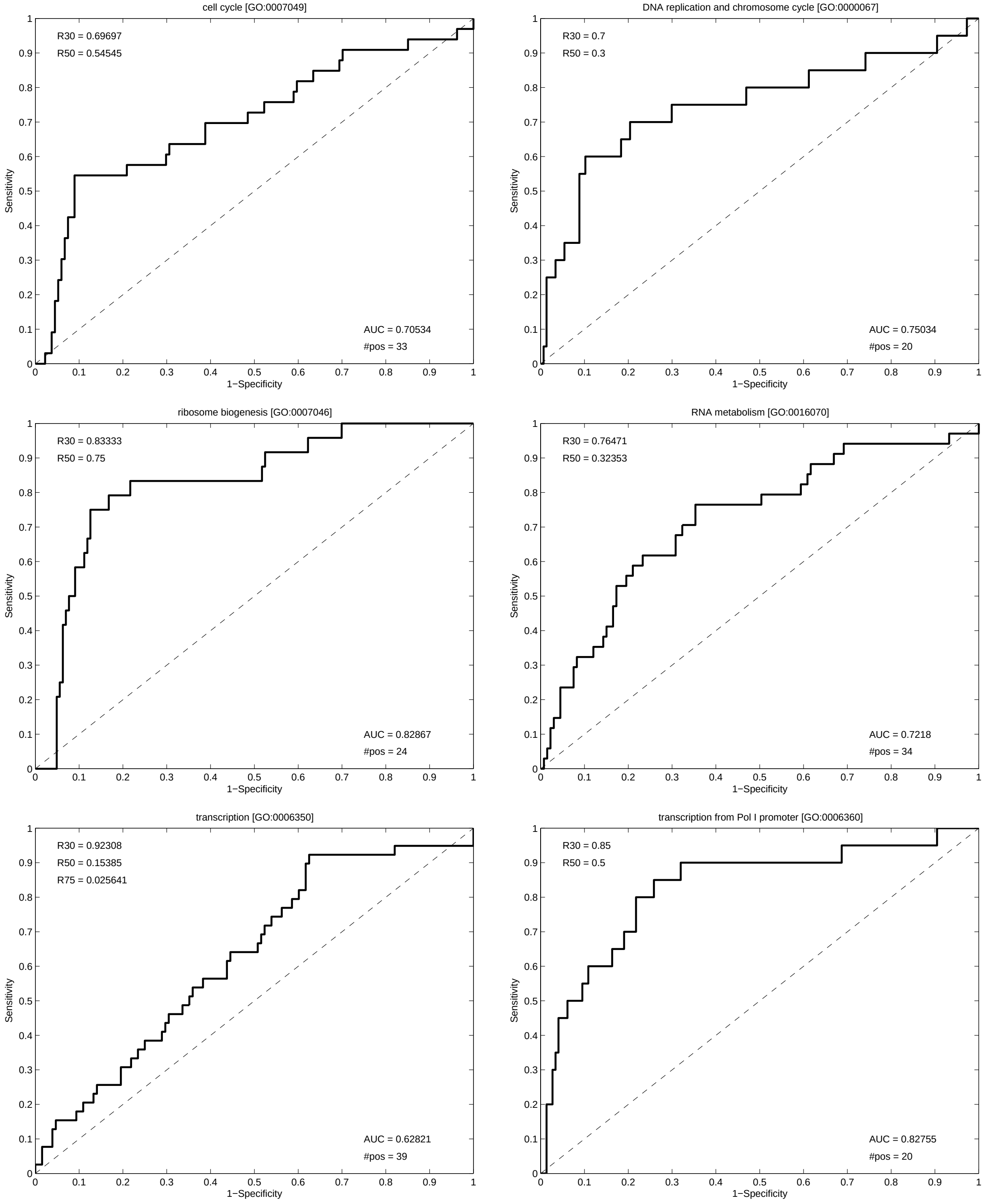

Figure 4.1: ROC curves for some of the classifiers we used for making predictions

significance of precision computed, we chose to report predictions only if their precision was above 50% and was computed using 10 predicted positives or more. We call *confident prediction* one that satisfies this constraint. Because of this constraint and since we are only interested in precisions above 50%, we can never make any prediction for biological processes for which we have fewer than 5 positive samples.

We have actually investigated several constraints for the minimum number of predicted positives for the computation of precisions and chose 10 as a compromise between having sufficient confidence in the precision and not eliminating too many predictions by having a severe constraint.

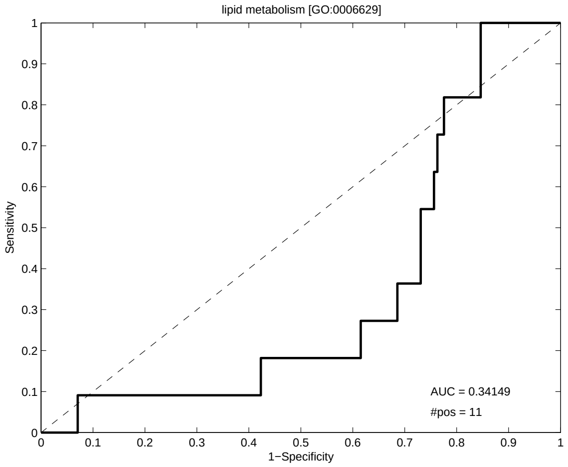

Figure 4.2: ROC curve for one of the rejected classifiers

We can verify in figure 4.1 that this method leads to classifiers having good ROC curves, *i.e.* the curves reach a high true positive rate for a false positive rate which is still very low. As a negative control, an example of the ROC curve of a rejected classifier is shown in figure 4.2.

## 4.4 Predicting functions for the unlabeled genes

For classifying the unlabeled samples, we trained a logistic regression classifier per GO biological process using all the available labeled samples and then computed the posterior probabilities $P(Y = 1|X = x)$ where $x$ were the unlabeled samples. Thresholds leading to the different precision levels were then computed and finally unlabeled samples were classified as positives whenever their posterior were greater than a threshold, and predictions were reported.

| GO biological process | ORF | threshold (minimum precision) | discriminant value |
|---|---|---|---|
| RNA metabolism [GO:0016070] | YHR040W | 0.5 | 0.585 |
| RNA metabolism [GO:0016070] | YNL245C | 0.5 | 0.540 |
| RNA processing [GO:0006396] | YHR040W | 0.5 | 0.561 |
| cell cycle [GO:0007049] | YLR457C | 0.6 | 0.625 |

Table 4.1: Example of predictions

Predictions are grouped by the precision level used and by biological process and are separated into batches depending on which of Hughes' dataset was used. Each prediction has four fields: a GO biological process, a systematic gene name, the precision level used for computing the threshold and finally the posterior probability which characterizes the confidence we have in the prediction.

Predictions have the format shown in table 4.1 and are grouped by GO biological process and sorted by decreasing discriminant values for each threshold level so that the more confidant predictions are at the top of the list in each GO biological process group, once again this is because of the cost associated with experimentally validating these predictions in wet lab. Biologists can therefore start by testing the most confident predictions first. The precisions reported are minimums, that is, the corresponding sample passed the threshold test for that precision level, but it could also have passed a test at a higher precision. All the predictions were assembled in tab delimited text files and are available as supplementary data.

| Number of predictions for GO-BP: | known in GOBP | # pos in samples | precision .6 | precision .5 |
|---|---|---|---|---|
| transcription [GO:0006350] | 534 | 39 | | 6 |
| transcription, DNA-dependent [GO:0006351] | 505 | 39 | | 6 |
| cell proliferation [GO:0008283] | 571 | 37 | 5 | 8 |
| RNA metabolism [GO:0016070] | 336 | 34 | | 10 |
| cell cycle [GO:0007049] | 494 | 33 | 4 | 6 |
| RNA processing [GO:0006396] | 297 | 33 | | 4 |
| mitotic cell cycle [GO:0000278] | 288 | 30 | | 5 |
| ribosome biogenesis and assembly [GO:0042254] | 186 | 26 | | 18 |
| ribosome biogenesis [GO:0007046] | 151 | 24 | | 17 |
| macromolecule biosynthesis [GO:0009059] | 449 | 21 | 1 | 1 |
| protein biosynthesis [GO:0006412] | 442 | 21 | 1 | 1 |
| DNA replication and chromosome cycle [GO:0000067] | 219 | 20 | | 1 |
| transcription from Pol I promoter [GO:0006360] | 149 | 20 | 7 | 8 |

Table 4.2: Summary of confident predictions made by logistic regression on the gene expression data

Several unlabeled genes can be predicted in the same GO biological process and respectively, a gene can be predicted to be involved in several GO biological processes and we have therefore complied with the non-classical classification task we were trying to solve.

## 4.5 The challenge of having few positive samples

Since most genes are not involved in most biological processes, for a particular biological process, there are far more negatives in the labeled samples than positives.

Numerical problems happen when the the classes are highly unbalanced. We observed that the procedure of fitting a hyperplane using logistic regression converged only for biological process involving more than 10 positives samples. This comes from the the log likelihood underflowing because the term $P(Y = 1|X = x)$ is too close to zero in these extreme situations. We later found out that adding a regularization term to the likelihood would solve this problem, but we have not yet tried running our scripts again for these biological processes.

Therefore we have restricted this study to learning GO biological processes for which we had more than 10 positive samples. From 1484 categories in the Gene Ontology, 573 GO biological processes involve ten or more genes.

We reduced the number of classification problems by not considering biological processes which are too broad and general such as *protein metabolism [GO:0019538]* or *cell organization and biogenesis [GO:0016043]*. These large, top-level (high in the gene ontology hierarchy) involve hundreds of genes and are often not specific enough to verify experimentally. Therefore we have restricted this study by not including biological processes that clearly involved too many genes to be interesting.

## 4.6 Results

Instead of reporting here a list of these predictions, which can be found in the supplementary data, we report summaries of these predictions in tables 4.2 to 4.4. These tables give the number of unlabeled genes predicted grouped by biological process and by precision level. We also indicate the number of genes known to be involved in each biological process as well as the number of positive samples available in the dataset used.

We see that the expression data leads to many more predictions with precision 50% and above than the size data. Nevertheless, it is still interesting to use the size data because it leads to predictions for different biological processes. For example, logistic regression on the size data generates some predictions of genes involved in *rRNA processing* or in *organelle organization and biogenesis* with precision above 50%. Logistic regression on the expression data does not produce any predictions for these biological processes. A similar argument could be raised for the use of logistic regression on the drug sensitivity

| Number of predictions for GO-BP: | known in GOBP | # pos in samples | precision .6 | precision .5 |
|---|---|---|---|---|
| transcription [GO:0006350] | 534 | 142 | | 4 |
| transcription, DNA-dependent [GO:0006351] | 505 | 141 | | 4 |
| RNA metabolism [GO:0016070] | 336 | 128 | 4 | 15 |
| RNA processing [GO:0006396] | 297 | 127 | | 17 |
| cell proliferation [GO:0008283] | 571 | 116 | | 5 |
| ribosome biogenesis and assembly [GO:0042254] | 186 | 85 | 3 | 15 |
| ribosome biogenesis [GO:0007046] | 151 | 79 | 3 | 15 |
| transcription from Pol I promoter [GO:0006360] | 149 | 76 | | 14 |
| rRNA processing [GO:0006364] | 121 | 65 | | 12 |
| organelle organization and biogenesis [GO:0006996] | 550 | 61 | | 2 |

Table 4.3: Summary of confident predictions made by logistic regression on the cell size distributions

dataset.

| Number of predictions for GO-BP: | known in GOBP | # pos in samples | precision .6 | precision .5 |
|---|---|---|---|---|
| RNA metabolism [GO:0016070] | 336 | 140 | | 1 |
| RNA processing [GO:0006396] | 297 | 139 | | 1 |
| cell proliferation [GO:0008283] | 571 | 127 | 3 | 5 |
| ribosome biogenesis and assembly [GO:0042254] | 186 | 93 | | 5 |
| ribosome biogenesis [GO:0007046] | 151 | 86 | | 6 |
| rRNA processing [GO:0006364] | 121 | 71 | | 3 |

Table 4.4: Summary of confident predictions made by logistic regression on the drugs dataset

It is worth emphasizing the fact that precision levels reported are minimums. A sample being predicted positive at a precision level could very well have been predicted positive at a higher precision level.

# Chapter 5

# Hierarchical clustering with labels for predicting gene function

## 5.1 Introduction

In this chapter we investigate an unsupervised learning tool for making predictions for the unlabeled genes. We tried to develop an algorithm that would solve several of the problems raised by the logistic regression.

First, it should be able to learn biological processes for which our samples contain fewer positives. We saw in the previous chapter that logistic regression did not make any predictions for biological processes for which we had fewer than twenty positives. We will see that our proposed algorithm is able to predict with high confidence that genes are involved in biological processes that only have five positives.

Second, one of the main limitations of the logistic regression is that it has linear decision boundaries and works well in cases such as in the example represented in figure 5.1, where the two classes are somewhat separable by a hyperplane. But logistic regression has difficulties discriminating between positives and negatives when the two classes are not linearly separable as in figure 5.2. For such biological processes, even if we translate

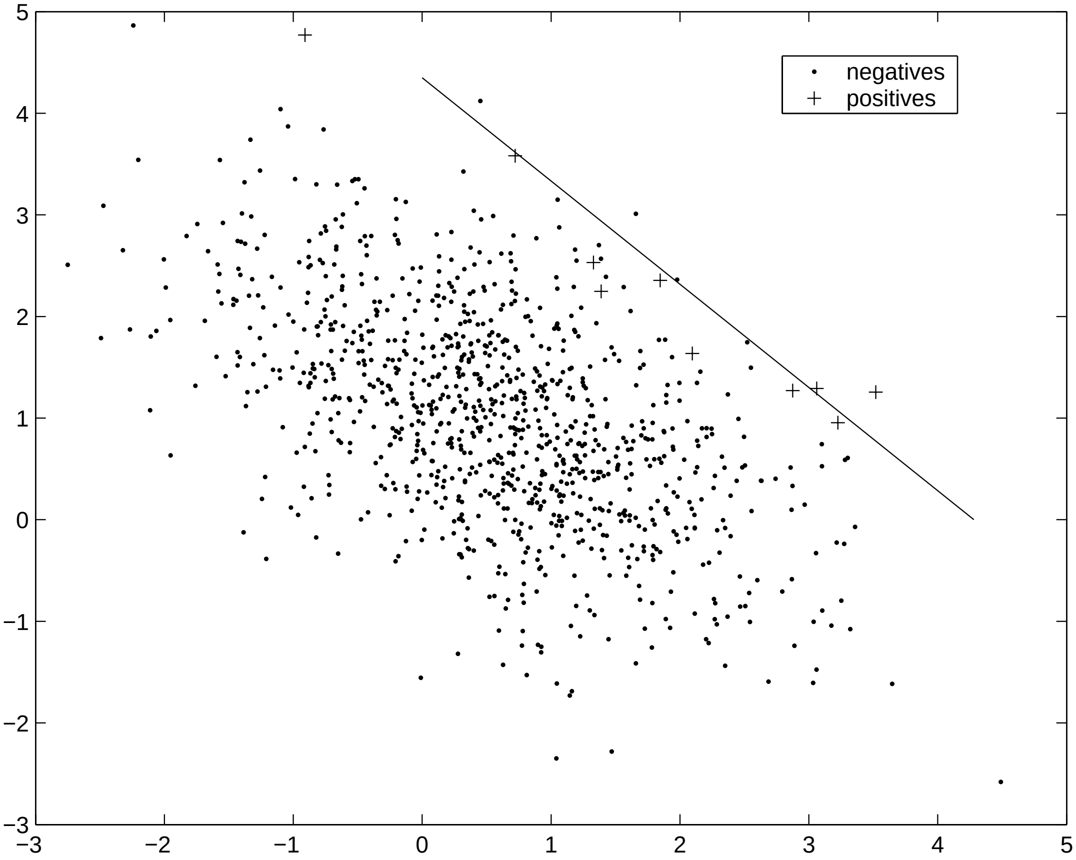

Figure 5.1: Toy classification problem solved by logistic regression

the decision boundary, by choosing a low threshold (cf. chapter 4.3), we can never reach a satisfying precision. In this chapter we investigate a method based on a different view of the data. We consider here that positive samples represent small islands among a sea of negatives, but we don't know how many islands there are nor their size. One possibility would be to use $k$-nearest neighbors ($k$NN), but unfortunately we have no idea what to expect for $k$, and a simple majority vote would not work because of the high number of negatives almost everywhere (including regions of relatively high concentrations of positives). We develop an algorithm based on hierarchical clustering that circumvents these problems.

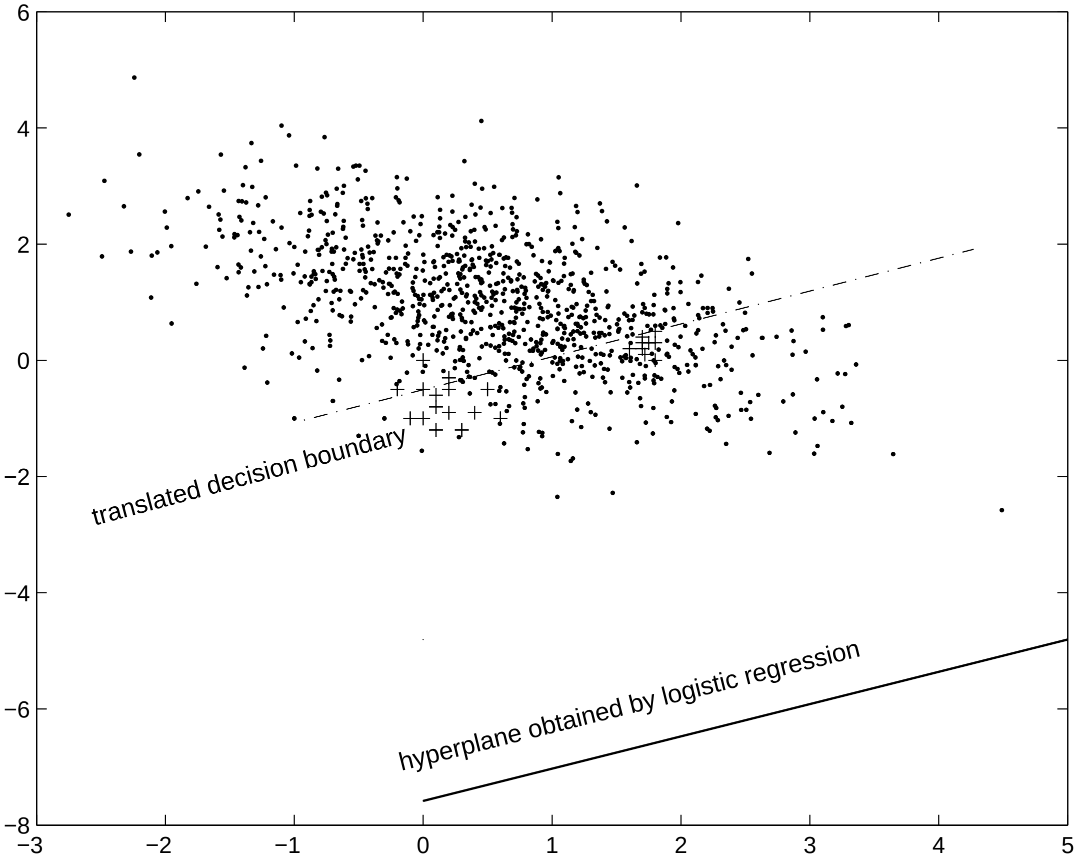

Figure 5.2: Toy classification problem not solved by logistic regression

Clustering has been used extensively in functional genomics [ABN+99, BDSY99, ESBB98, HMJ+00, SS00] and is probably what biologists use and trust most. A number of clustering algorithms have been applied to gene expression profiles, such as K-means, Kohonen self organizing maps and the graph-based algorithms Click [SS00] and CAST [BDSY99]. These algorithms generate clusters which are all assumed to be on the same level, thus they lack the ability to represent the relationships between genes and sub-clusters on different scales. This is important for our application because of the sparsity of positive examples which has the consequence that most, probably all, clusters have a very low proportion of positives.

Our method looks for regions in the data space of high concentrations of positives. All that is required is some notion of "distance" between all pairs of elements. In contrast, logistic regression does not work for the morphology dataset because, although the data is technically real valued, it is still too categorical for the fit to converge.

We based our algorithm on hierarchical clustering which had the advantage of being extensively used in the biology and bioinformatics worlds to find genes that share a common function [ESBB98] [AED$^{+}$00] or to group experimental conditions [ABN$^{+}$99], and therefore it will make results easy to interpret and to understand for biologists. Hierarchical clustering is widely used on gene expression data. The resulting dendrogram is usually displayed immediately beside the gene expression profiles heatmap from which it was derived. The leaves of the dendrogram are often labeled with gene names and/or biological processes in which these genes are involved. The method we develop here is based on this methodology, but extends it to an automated process. It also improves this methodology by using all of the available annotations for each gene whereas the common manual way only uses one annotation per leaf. Recall that a gene may be involved in several biological processes.

The idea is to cluster the data and look for clusters with a high proportion of positives, then label the unlabeled samples in such clusters if there are any. The hierarchical tree produced by hierarchical algorithms will allow us to find such positively enriched clusters, by concentrating on areas of the tree with a high density of positives. More precisely, rather than label a sample, the algorithm should produce a confidence level that the sample is positive. We will use heuristics and cross validation for producing such confidence levels.

The advantage of hierarchical clustering over a supervised method such as $k$-nearest neighbors, is that if we used $k$-NN in this situation with very few positive samples, most neighbors would be negatives for almost any region of the space unless $k$ is chosen very small, which would lead to overfitting the data and also would not give us any confidence measure for the prediction, whereas as we shall see, using hierarchical clustering will not only let us find positive enriched clusters but will also facilitate the production of a confidence level for each prediction.

This algorithm is not restricted to making predictions for the unlabeled genes but can also be used to relabel negative samples, *i.e.* make some predictions for negative samples that lie in positive enriched regions.

## 5.2 Hierarchical clustering

The hierarchical clustering algorithm requires a measure of dissimilarity between two disjoint groups of observations. These group dissimilarity measures are based on pairwise dissimilarities between observations of the two groups. The algorithm can easily be converted if one has a similarity measure for pairs of elements by transforming it into a dissimilarity measure. For example if one is using Pearson's correlation $r_{i,j}$ as a measure of similarity between elements $i$ and $j$, a dissimilarity measure is $d_{i,j} = 1 - r_{i,j}$. Other common dissimilarities for pairs of observations are the euclidean distance, the Manhattan or city block distance, the Mahalanobis distance and the Minkowski distance. There are three common group dissimilarity measures: in the *single linkage* case, the distance between two groups is taken to be the minimum distance between any two observations of the two groups.

$$D_{G,H} = \min(d_{k,l} \mid k \in G \ , \ l \in H) \tag{5.1}$$

In the *complete linkage* case, the distance between two groups is considered to be the maximum distance between any two elements of the groups.

$$D_{G,H} = \max(d_{k,l} \mid k \in G \ , \ l \in H) \tag{5.2}$$

The third most common group dissimilarity is the *average linkage* which as it's name indicates is the average of the dissimilarities between all pairs of elements from the two groups.

$$D_{G,H} = \frac{1}{|G||H|} \sum_{k \in G} \sum_{j \in H} d_{k,l} \tag{5.3}$$

where $|G|$ and $|H|$ denote the size of groups $G$ and $H$.

The algorithm produces a hierarchical representation in which the clusters at each level are obtained by merging two clusters at the next lower level. This representation can be organized as a tree, the root being the entire dataset and the leaves being the observations. For a dataset of $N$ observations, the hierarchy will have $N - 1$ groups of 2 elements or more (nodes in the tree). Another advantage of this method over clustering methods like $k$-means is that the user need not specify the number of clusters $k$ a priori.

There are two types of hierarchical clustering algorithms. In the agglomerative case, clusters at a lower level are grouped together to form one cluster at the next higher level. The algorithm works bottom-up, first joining the most similar observations, and then grouping the two most similar clusters, where a cluster can be a singleton or a group of observations, until all observations are grouped into one cluster: the root of the tree. In the divisive case, the algorithm works top-down, starting with all observations in one group and dividing each group into two smaller disjoint groups at the next lower level, until all groups are singletons. Each split is chosen so that the resulting two groups have the greatest between-group dissimilarity.

Each level of the hierarchy represents a particular grouping of the data, it is then up to the user to decide which level best represents a clustering. There are several techniques for helping the user choose the appropriate level at which to cut the tree, see [HTF01] for further details. This is a drawback of hierarchical clustering, although it will not matter for our particular application.

## 5.3 Details of our algorithm

We first build a hierarchical tree on all available labeled and unlabeled samples using hierarchical agglomerative clustering [HMS01] with average linkage and Pearson's correlation for the distance metric since it was shown that it is best suited for gene expression data. In constructing the tree, we ignore the labels on the data. In this way, we can include both labeled and unlabeled data in the tree, and more importantly, we can use the same tree for each biological process, thus saving on computing time, since the tree need only be built once. Thus, the construction of the tree can be viewed as a preprocessing step whose cost is amortized over all the biological processes. However, after the tree is constructed, it is not possible to add new unlabeled samples to the data.

We used the correlation coefficient between two samples as a measure of the distance between them rather than Euclidean distance. This is because the actual level of response of two gene probes is less important, and less trustworthy, than their profiles being correlated among a set of experiments. For example, the measured expression of a gene might be twice that of another gene in the same pathway because of experimental factors such as oligonucleotide probe quality (folding into a stable secondary structure, melting temperature etc).

Following the construction of the tree, we use it to build a classifier for each biological process. Recall that each such process provides a different set of labels for genes. Since the leaves of our tree represent genes, each leaf is assigned the label of the gene it represents. Leaves for unlabeled genes are labeled as negative, since it is likely that an unlabeled gene is not involved in any particular biological process. (We also flag such leaves, so as to remember that they are unlabeled). We can now look in the tree for regions of high concentrations of positive leaves, after which we assign labels to all the unlabeled genes that fall in such regions. These assignments represent our classifiers predictions.

To make these assignments, the algorithm computes a score $\sigma$ for each internal node in the tree, reflecting the concentration of positives at the leaves under the node.

$$\sigma = \frac{\text{\# of positives at leaves}}{\text{\# of leaves}} \times \left(1 - \alpha e^{-\text{\# of positives}}\right) \tag{5.4}$$

The first factor in this formula is the proportion of positive leaves under the node, it reflects the concentration of positives in the region of the data space in which the leaves are. The second factor tends to one when the number of positives raises, and tends to zero as the number of positives decreases. It gives more importance (higher score) to nodes with more positive leaves, *i.e.*‘ to larger regions of positive concentration, since we regard such regions to be more statistically significant. (We have used $\alpha = 0.5$ and haven't investigated tweaking this parameter nor using other functions for the second factor of this equation.) We then define the score of a leaf to be the maximum score of all it's ancestors (internal nodes). Since unlabeled samples are leaves in the tree, they automatically receive a score, which we use to classify them.

Before building decision rules, we use a technique similar to the cross validation of the previous section. At each iteration, we effectively remove a labeled sample by treating it as unlabeled. The scoring process described above is repeated each time. This provides a score for the labeled sample being treated as unlabeled. Each labeled leaf is scored in this way.

It is now easy to build a decision rule. We simply set a threshold, and a leaf is classified as positive if its score is above the threshold. To evaluate the rule, we apply it

```
Build hierarchical tree on all labeled and unlabeled samples.
For each GO biological process GO-BP[i] do {
  Label the leaves according to GO-BP[i].
  Label unlabeled samples as negatives.
  For each sample S[j] do {
    Relabel S[j] as negative.
    Compute the score of all internal nodes.
    Compute the score of S[j] as maximum score of all it's ancestors.
  }
  Find lowest threshold that achieves user-specified precision.
  Classify unlabeled samples using this threshold.
  Report predicted positives.
}
```

Figure 5.3: Pseudo-code of the algorithm

to labeled leaves, and compare each leaf's true label to its predicted label. A threshold that achieves a user-specified precision is then chosen. Finally, using this threshold, we use the decision rule to classify all the unlabeled data.

The pseudo-code for this algorithm is given in Figure 5.3. A toy example of how the tree is reused for each biological process is given in Figure 5.4. Our method is very fast, the whole process from building the tree to reporting predicted positives in all biological processes took a few seconds for each dataset on a Pentium IV 2GHz. This should be contrasted with the logistic regression methodology which required approximately a half hour for each dataset.

## 5.4 Results

Predicted positives were reported for all four datasets and assembled in tab delimited files. A prediction has four fields: a GO biological process, the gene systematic name, the difference between the score of the unlabeled leaf and the threshold used, and the

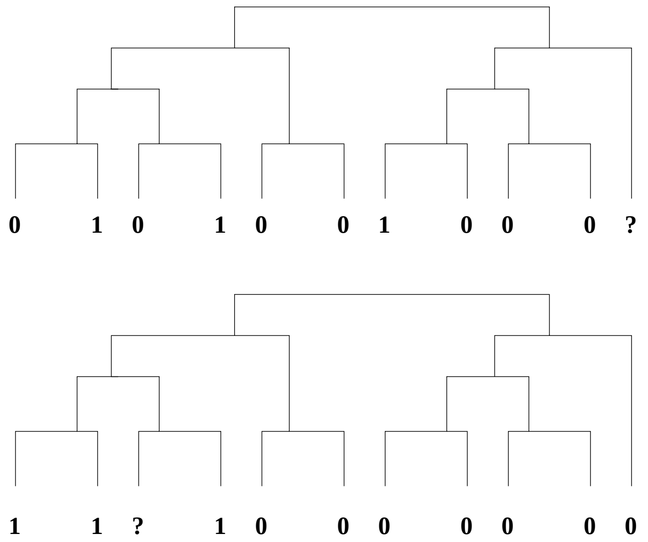

Figure 5.4: Toy hierarchical tree reused with labels from two biological processes

precision corresponding to that threshold. The precision and the difference between the score and the threshold represent the confidence we have in the prediction. Summaries of these predictions (except for the morphology dataset) are shown in Table 5.1-5.3. The summary for the morphology dataset is not shown, the number of confident predictions made were approximately the same as in the size and drugs datasets.

| Number of predictions for GO-BP: | known in GOBP | # pos in samples | precision .5 |
|---|---|---|---|
| transcription [GO:0006350] | 534 | 142 | 5 |
| transcription, DNA-dependent [GO:0006351] | 505 | 141 | 5 |
| cell proliferation [GO:0008283] | 571 | 116 | 1 |
| cell cycle [GO:0007049] | 494 | 99 | 1 |
| ribosome biogenesis and assembly [GO:0042254] | 186 | 85 | 5 |
| ribosome biogenesis [GO:0007046] | 151 | 79 | 2 |
| transcription from Pol I promoter [GO:0006360] | 149 | 76 | 3 |

Table 5.1: Summary of confident predictions made by our clustering method on the cell size distributions

Comparing the two methods for identical datasets (Table 4.2 vs. 5.3, Table 4.3 vs. 5.1 and Table 4.4 vs. 5.2), we observe that our method produces many more confident predictions, at precision levels 50% and 60% (even 75% with the drugs dataset), and for more biological processes, but not for the size data. In particular, our hierarchical

| Number of predictions for GO-BP: | known in GOBP | # pos in samples | precision .75 | precision .6 | precision .5 |
|---|---|---|---|---|---|
| transcription [GO:0006350] | 534 | 159 | 2 | 5 | 5 |
| transcription, DNA-dependent [GO:0006351] | 505 | 158 | 2 | 5 | 5 |
| RNA metabolism [GO:0016070] | 336 | 140 | | | 27 |
| RNA processing [GO:0006396] | 297 | 139 | | | 17 |
| DNA metabolism [GO:0006259] | 379 | 68 | | | 6 |
| mRNA processing [GO:0006397] | 124 | 60 | | | 4 |
| nuclear organization and biogenesis [GO:0006997] | 213 | 42 | | 3 | 3 |
| chromosome organization and biogenesis (sensu Eukarya) [GO:0007001] | 178 | 35 | | 3 | 3 |
| establishment and/or maintenance of chromatin architecture [GO:0006325] | 155 | 32 | | 3 | 3 |
| DNA packaging [GO:0006323] | 155 | 32 | | 3 | 3 |

Table 5.2: Summary of confident predictions made by our clustering method on the drugs dataset

method made prediction for 18 biological processes involving fewer than 20 positives in the samples whereas logistic regression produced none. Moreover our algorithm was able to make confident predictions for biological processes that had as few as five positive samples, which is the lower bound achievable by design of this algorithm since we constrained predictions to be based on ten positive samples and only report those with precision above 50%.

In Figure 4.1 we show the ROC curves of a couple of the classifiers used for making predictions, obtained by the method we developed. We clearly see that our method performs better than guessing the majority class, here it would mean to classify as negative every time, and achieves very high true positive rates at thresholds for which the false positive rates are still very low. For example, the classifier used for predicting genes involved in *glycerophospholipid biosynthesis* reaches a true positive rate of 100% for less than 2% false positive rate. These ROC curves are also much better than those of figure 4.1 obtained by logistic regression.

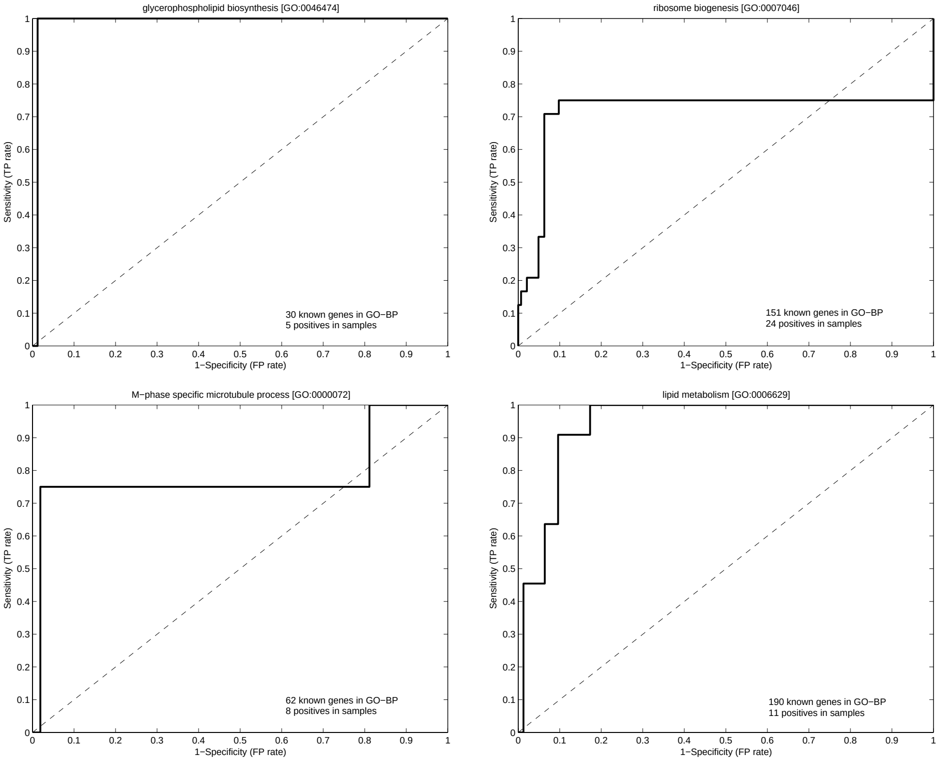

Figure 5.5: ROC curves for some of the classifiers we used for making predictions

| Number of predictions for GO-BP: | known in GOBP | # pos in samples | precision .6 | precision .5 |
|---|---|---|---|---|
| transcription [GO:0006350] | 534 | 39 | | 18 |
| transcription, DNA-dependent [GO:0006351] | 505 | 39 | | 18 |
| RNA metabolism [GO:0016070] | 336 | 34 | 9 | 11 |
| RNA processing [GO:0006396] | 297 | 33 | 11 | 11 |
| ribosome biogenesis and assembly [GO:0042254] | 186 | 26 | 19 | 21 |
| ribosome biogenesis [GO:0007046] | 151 | 24 | 12 | 19 |
| protein modification [GO:0006464] | 361 | 23 | | 3 |
| organelle organization and biogenesis [GO:0006996] | 550 | 22 | | 1 |
| macromolecule biosynthesis [GO:0009059] | 449 | 21 | | 9 |
| protein biosynthesis [GO:0006412] | 442 | 21 | | 9 |
| transcription from Pol I promoter [GO:0006360] | 149 | 20 | 11 | 11 |
| rRNA processing [GO:0006364] | 121 | 18 | 8 | 8 |
| catabolism [GO:0009056] | 276 | 16 | | 2 |
| cytoskeleton organization and biogenesis [GO:0007010] | 255 | 14 | | 2 |
| mRNA processing [GO:0006397] | 124 | 14 | | 4 |
| macromolecule catabolism [GO:0009057] | 176 | 12 | | 1 |
| lipid metabolism [GO:0006629] | 190 | 11 | | 1 |
| lipid biosynthesis [GO:0008610] | 111 | 11 | | 1 |
| RNA splicing [GO:0008380] | 112 | 10 | | 4 |
| mRNA splicing [GO:0006371] | 92 | 10 | | 4 |
| microtubule-based process [GO:0007017] | 94 | 8 | | 1 |
| microtubule cytoskeleton organization and biogenesis [GO:0000226] | 86 | 8 | | 1 |
| M-phase specific microtubule process [GO:0000072] | 62 | 8 | | 1 |
| membrane lipid metabolism [GO:0006643] | 85 | 6 | | 1 |
| membrane lipid biosynthesis [GO:0046467] | 62 | 6 | | 1 |
| phospholipid metabolism [GO:0006644] | 64 | 5 | | 1 |
| phospholipid biosynthesis [GO:0008654] | 48 | 5 | | 1 |
| glycerophospholipid metabolism [GO:0006650] | 34 | 5 | | 1 |
| glycerophospholipid biosynthesis [GO:0046474] | 30 | 5 | | 1 |

Table 5.3: Summary of confident predictions made by our clustering method on the gene expression data

# Chapter 6

# Conclusions and future work

We investigated several techniques for filling missing values in gene expression data, trying to minimize the normalized root mean squared error of our estimators. We were able to achieve good results, sometimes better than the current best public software, on simulated data of low dimension with the HLS and ILS algorithms.

We developed a method based on hierarchical clustering for labeled data to find regions in the data space of relatively high concentration of positives. This technique allows the analysis of biological processes involving very few genes. With this method, we were able to make confident predictions at precisions of 50% and above for biological processes for which our samples contained as few as 5 positives. The methodology developed here is not restricted to learning essential genes, but could be applied to any set of genes for which there is quantitative data. We compared our algorithm with logistic regression and found that either one can perform better depending on the biological process of interest, but that our algorithm produces more confident predictions for more biological processes, especially for those for which we have very few positive samples.

We used correlation as a measure of similarity between pairs of elements and average linkage to build the hierarchical tree. It would be interesting to investigate different distance metrics and especially other linkage strategies such as single linkage, which produces clusters that are typically less compact.

We focused on making predictions for unlabeled genes. However, it would be biologically interesting to report cases in which a gene's true label is negative but whose predicted label is a confident positive. This is because negative labels in our dataset are sometimes wrong. A more challenging task would be to use datasets concurrently for the intersecting samples and independently for disjoint sets of samples. Also finding methods for learning biological processes concurrently rather than independently is one of our future goals. We are thinking of using the Gene Ontology hierarchy to propagate up the hierarchy predictions made lower down, because if a gene is involved in a biological process, it is also involved processes above it in the hierarchy. This is not completely trivial because the hierarchy is not a tree and a biological process can have several parents. For examples vitamin biosynthesis is child of *biosynthesis* and *vitamin metabolism*. More interestingly, if a prediction is made in a biological process having children, we would like to find methods for making the prediction more specific by propagating it down the hierarchy as far as possible.

Another area of this thesis which could be developed is the tweaking parameter $\alpha$ in the second factor of equation 5.4. Other functions should also be tested in place of the whole second factor.

Finally, the iterative least squares regression for filling missing values should be analyzed in higher dimensions, and with a small number of samples to better simulate the type of datasets encountered with gene expression profiles.

# Bibliography

[ABN$^+$99] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Science*, 96(12):6745–6750, June 1999.

[AED$^+$00] Ash A. Alizadeh, Michael B. Eisen, R. Eric Davis, Chi Ma, Izidore S. Lossos, Andreas Rosenwald, Jennifer C. Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, John I. Powell, Liming Yang, Gerald E. Marti, Troy Moore, James Jr Hudson, Lisheng Lu, David B. Lewis, Robert Tibshirani, Gavin Sherlock, Wing C. Chan, Timothy C. Greiner, Dennis D. Weisenburger, James O. Armitage, Roger Warnke, Ronald Levy, Wyndham Wilson, Michael R. Grever, John C. Byrd, David Botstein, Patrick O. Brown, and Louis M. Staudt. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, February 2000.

[BCRC04] Joel S Bader, Amitabha Chaudhuri, Jonathan M Rothberg, and John Chant. Gaining confidence in high-throughput protein interaction networks. *Nature Biotechnology*, 22(1), January 2004.

[BDSY99] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal Of Computational Biology*, 6:281–297, 1999.

[BGL$^{+}$00] Michael P S Brown, William Noble Grundy, David Lin, Nellon Cristianini, Charles Walsh Sugnet, Terrence S Furey, Manuel Ares, and David Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceeding of the National Academy of Science*, 97(1):262–7, January 2000.

[BK97] C Burge and S Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.

[CWB$^{+}$04] K R Christie, S Weng, R Balakrishnan, M C Costanzo, K Dolinski, S S Dwight, S R Engel, B Feierbach, D G Fisk, J E Hirschman, E L Hong, L Issel-Tarver, R Nash, A Sethuraman, B Starr, C L Theesfeld, R Andrada, G Binkley, Q Dong, C Lane, M Schroeder, D Botstein, and J M Cherry. Saccharomyces genome database (sgd) provides tools to identify and analyze sequences from saccharomyces cerevisiae and related sequences from other organisms. *Nucleic Acids Research*, 32:D311–D314, 2004.

[DHD$^{+}$02] Selina S. Dwight, Midori A. Harris, Kara Dolinski, Catherine A. Ball, Gail Binkley, Karen R. Christie, Dianna G. Fisk, Laurie Issel-Tarver, Mark Schroeder, Gavin Sherlock, Anand Sethuraman, Shuai Weng, David Botstein, and J. Michael Cherry. Saccharomyces genome database (sgd) provides secondary gene annotation using the gene ontology (go). *Nucleic Acids Research*, 30(1):69–72, 2002.

[ESBB98] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Science*, 95(25):14863–14868, December 1998.

[FCD$^{+}$00] T S Furey, N Cristianini, N Duffy, D W Bednarski, M Schummer, and D Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–14, October 2000.

[GEO] http://www.ncbi.nlm.nih.gov/geo/.

[GPAH97] E. Gari, L. Piedrafita, M. Aldea, and E. Herrero. A set of vectors with a tetracycline-regulatable promoter system for modulated gene exression in *saccharomyces cerevisiae*. *Yeast*, 13:837–848, 1997.

[HMJ+00] Timothy R. Hughes, Matthew J. Marton, Allan R. Jones, Christopher J. Roberts, Roland Stoughton, Christopher D. Armour, Holly A. Bennett, Ernest Coffey, Hongyue Dai, Yudong D. He, Matthew J. Kidd, Amy M. King, Michael R. Meyer, David Slade, Pek Y. Lum, Sergey B. Stepaniants, Daniel D. Shoemaker, Daniel Gachotte, Kalpana Chakraburtty, Julian Simon, Martin Bard, and Stephen H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, July 2000.

[HMJ+01] T R Hughes, M Mao, A R Jones, J Burchard, M J Marton, K W Shannon, S M Lefkowitz, M Ziman, J M Schelter, M R Meyer, S Kobayashi, C Davis, H Dai, Y D He, S B Stephaniants, G Cavet, W L Walker, A Westand, E Coffey, D D Shoemaker, R Stoughton, A P Blanchard, S H Friend, and P S Linsley. Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer. *Nature Biotechnology*, 19(4):342–347, April 2001.

[HMS01] D Hand, H Mannila, and P Smyth. *Principles of Datamining*. MIT Press, 2001.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[Hug02] T R Hughes. Yeast and drug discovery. *Functional and Integrative Genomics*, 2(4-5):199–211, September 2002.

[KWR$^+$01] Javed Khan, Jun S. Wei, Markus Ringner, et. al, and Paul S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7(6):673–679, 2001.

[LL03] Y Lee and C K Lee. Classification of multiple cancer types by multicategory support vector machines using gene expression data. *Bioinformatics*, 19(9):1132–9, June 2003.

[MDH$^+$04] Sanie Mnaimneh, Armaity P. Davierwala, Jennifer Haynes, Jason Moffat, Wen-Tao Peng, Wen Zhang, Xueqi Yang1, Jeff Pootoolal, Gordon Chua, Andres Lopez, Miles Trochesset, Darcy Morse, Nevan J. Krogan, Shawna L. Hiley, Zhijian Li, Quaid Morris, Jrg Grigul, Nicholas Mitsakakis, Christopher J. Roberts, Jack F. Greenblatt, Charles Boone, Chris A. Kaiser, Brenda J. Andrews, and Timothy R. Hughes. Exploration of essential gene functions via titratable promoter alleles. *Cell*, 118(1):31–44, July 9th 2004.

[MNA] http://hugheslab.med.utoronto.ca/Mnaimneh.

[NR02] D V Nguyen and D M Rocke. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, 18(1):39–50, Jan 2002.

[OST$^+$03] Shigeyuki Oba, M. Sato, Ichiro Takemasa, Morito Monden, K. Matsubra, and Shin Ishii. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

[RSA00] S Raychaudhuri, J M Stuart, and R B Altman. Principal components analysis to summarize microarray experiments: Application to sporulation time series. *Proceedings of the Pacific Symposium on Biocomputing*, 5:452–463, 2000.

[RTR$^+$01] S Ramaswamy, P Tamayo, R Rifkin, S Mukherjee, C H Yeang, M Angelo, C Ladd, M Reich, E Latulippe, J P Mesirov, T Poggio, W Gerald, M Loda, E S Lander, and T R Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of National Academy of Science*, 98(26):15149–54, December 2001.

[SS00] R. Sharan and R. Shamir. Click: A clustering algorithm for gene expression analysis. In *ISMB*, 2000.

[SSDB95] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470, October 1995.

[TCS$^+$01] Olga Troyanskaya, Michael Cantor, Gavin Serlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Bostein, and Russ B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, June 2001.

[GO04] Gene Ontology Consortium. The gene ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32:D258–D261, 2004.

[WBD$^+$01] Mike West, Carrie Blanchette, Holly Dressman, Erich Huang, Seiichi Ishida, Rainer Spang, Harry Zuzan, John A. Olson, Jeffrey R. Marks, and Joseph R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Science*, 98(20):11462–11467, September 2001.