View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by KU ScholarWorks

A MACHINE LEARNING APPROACH TO QUERY TIME-SERIES MICROARRAY DATA SETS FOR FUNCTIONALLY RELATED GENES USING HIDDEN MARKOV MODELS

BY

Alexander Senf

Submitted to the graduate degree program in Computer Science
and the Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy.

Xue-wen Chen, PhD _____
Chairperson

Jun Huan, PhD _____*

James Miller, PhD _____*

Arvin Agah, PhD _____*

Ilya Vakser, PhD _____*
Committee members*

Date defended: _____

The Dissertation Committee for Alexander Senf certifies
that this is the approved version of the following dissertation:

A MACHINE LEARNING APPROACH TO QUERY TIME-SERIES MICROARRAY DATA SETS FOR FUNCTIONALLY RELATED GENES USING HIDDEN MARKOV MODELS

Committee:

Xue-wen Chen, PhD _____*
Chairperson*

_____

_____

_____

_____

Date approved:_____

To Andrea

ABSTRACT

A Machine Learning Approach to Query Time-Series Microarray Data Sets for Functionally Related Genes using Hidden Markov Models

Alexander J. Senf

Chairperson: Xue-wen Chen

Microarray technology captures the rate of expression of genes under varying experimental conditions. Genes encode the information necessary to build proteins; proteins used by cellular functions exhibit higher rates of expression for the associated genes. If multiple proteins are required for a particular function then their genes show a pattern of coexpression during time periods when the function is active within a cell. Cellular functions are generally complex and require groups of genes to cooperate; these groups of genes are called functional modules. Modular organization of genetic functions has been evident since 1999. Detecting functionally related genes in a genome and detecting all genes belonging to particular functional modules are current research topics in this field.

The number of microarray gene expression datasets available in public repositories increases rapidly, and advances in technology have now made it feasible to routinely perform whole-genome studies where the behavior of every gene in a genome is captured. This promises a wealth of biological and medical information, but making

the amount of data accessible to researchers requires intelligent and efficient computational algorithms. Researchers working on specific cellular functions would benefit from this data if it was possible to quickly extract information useful to their area of research.

This dissertation develops a machine learning algorithm that allows one or multiple microarray data sets to be queried with a set of known and functionally related input genes in order to detect additional genes participating in the same or closely related functions. The focus is on time-series microarray datasets where gene expression values are obtained from the same experiment over a period of time from a series of sequential measurements. A feature selection algorithm selects relevant time steps where the provided input genes exhibit correlated expression behavior. Time steps are the columns in microarray data sets, rows list individual genes. A specific linear Hidden Markov Model (HMM) is then constructed to contain one hidden state for each of the selected experiments and is trained using the expression values of the input genes from the microarray.

Given the trained HMM the probability that a sequence of gene expression values was generated by that particular HMM can be calculated. This allows for the assignment of a probability score for each gene in the microarray. High-scoring genes are included in the result set (of genes with functional similarities to the input genes.) P-values can be calculated by repeating this algorithm to train multiple individual HMMs using randomly selected genes as input genes and calculating a Parzen Density Function (PDF) from the probability scores of all HMMs for each gene.

A feedback loop uses the result generated from one algorithm run as input set for another iteration of the algorithm. This iterated HMM algorithm allows for the characterization of functional modules from very small input sets and for weak similarity signals.

This algorithm also allows for the integration of multiple microarray data sets; two approaches are studied: Meta-Analysis (combination of the results from individual data set runs) and the extension of the linear HMM across multiple individual data sets. Results indicate that Meta-Analysis works best for integration of closely related microarrays and a spanning HMM works best for the integration of multiple heterogeneous datasets.

The performance of this approach is demonstrated relative to the published literature on a number of widely used synthetic data sets. Biological application is verified by analyzing biological data sets of the Fruit Fly *D. Melanogaster* and Baker's Yeast *S. Cerevisiae*. The algorithm developed in this dissertation is better able to detect functionally related genes in common data sets than currently available algorithms in the published literature.

Table of Contents

List of Algorithms and Tables

List of Figures

List of Acronyms

ATP ..... Adenosine Triphosphate

BioGRID ..... Biological General Repository for Interaction Datasets

BN ..... Bayesian Network

ChIP ..... Chromatin Immunoprecipitation

CPU ..... Central Processing Unit

CTWC ..... Coupled Two-Way Clustering

CUDA ..... Compute Unified Device Architecture

DNA ..... Deoxyribonucleic Acid

EBI ..... European Bioinformatics Institute

EM ..... Expectation Maximization

FBA ..... Forward Backward Algorithm

FFR ..... Functional Feature Reduction

GEMS ..... Gene Expression Mining Server

GEO ..... Gene Expression Omnibus

GO ..... Gene Ontology

GPGPU ..... General-Purpose Graphics Processing Unit

GR ..... Gene Recommender

GRAM ..... Genetic Regulatory Modules

HMM ..... Hidden Markov Model

iHMM ..... Iterative Hidden Markov Model

ISA ..... Iterative Signature Algorithm

JAHMM ..... Java Hidden Markov Model

KEGG ..... Kyoto Encyclopedia on Genes and Genomes

KNN ..... K-Nearest Neighbor

MEFIT ..... Microarray Experiment Functional Integration Technology

MLP ..... Multi Layer Perceptron

mRNA ..... messenger RNA (Ribonucleic Acid)

NCBI ..... National Center for Biotechnology Information

NIH ..... National Institutes of Health

PDF ..... Parzen Density Function

QDB ..... Query-Driven Biclustering

RMSD ..... Root Mean Squared Deviation (Distance)

ROC ..... Receiver-Operator Characteristic

SA ..... Signature Algorithm

SAM ..... Sequence Alignment and Modeling

SAMBA ..... Statistical-Algorithmic Method for Bicluster Analysis

SGD ..... Saccharomyces Genome Database

SNN ..... Supervised Neural Network

SOM ..... Self Organizing Map

SPC ..... Superparamagnetic Clustering

SSL ..... Synthetic Sick or Lethal

SVD ..... Singular Value Decomposition

SVM ..... Support Vector Machine

Acknowledgements

I would like to thank my advisor, Xue-wen Chen, for his direction, discussion, comments and encouragement to always perform our best work.

I would also like to thank my family, and especially my wife, Andrea, for their continued support and understanding during this long and involved process.

Chapter 1 Introduction

To date close to 900 complete genomes of over 180 species have been sequenced (Liolios, Chen et al. 2010). A particular milestone was the sequencing of a complete human genome, generally seen as the beginning of the post-genomic era (Venter, Adams et al. 2001). Genomic data repositories, such as NCBI (Barrett, Troup et al. 2009) contain an ever increasing wealth of data, including DNA sequences, known and predicted genes, and a host of other information from multiple species. With a wealth of genomic sequence data now available from multiple species, more effort in molecular and computational biology has shifted to better understand the functions of the genes found in these genomes in a cellular context.

Cellular functions are carried out by proteins. Genes encode the information necessary to build proteins, so functionally related proteins can be identified by correlated higher rates of expression of the respective genes (coexpression), which is the underlying assumption for much of the work that is done in this field. Correlation between coexpression and functional relationships has been demonstrated in a number of studies (Eisen, Spellman et al. 1998; Ge, Liu et al. 2001; Jansen, Greenbaum et al. 2002; Kemmeren, van Berkum et al. 2002; Lee, Hsu et al. 2004). This has led to a variety of clustering algorithms aimed at detecting functionally related genes at the genetic level based on gene expression similarities (Eisen, Spellman et al. 1998; Tamayo, Slonim et al. 1999; Tavazoie, Hughes et al. 1999; Getz, Levine et al. 2000; Herrero and Dopazo 2002; Grotkjaer, Winther et al. 2006).

Functionally related genes can thus often be identified by similarities in gene expression patterns during time periods when the function is active in a cell.

The most prevalent technology in the study of gene behavior at a large scale is the gene expression microarray. Microarrays rapidly capture the abundance of gene products (mRNA) for thousands of genes and multiple experiments in the same data set. mRNA abundance under each experimental condition indicates the level of activity of a gene in response to the conditions set by the experiment. mRNA is a copy of a section of the DNA code that is passed outside of the cell nucleus and is used to assemble the amino acid chain which then folds into a protein; the higher the demand for that particular protein is, the more copies of a gene's code are produced.

Gene expression experiments are becoming more common and the number of microarray gene expression data sets available in public repositories increases rapidly. Advances in technology have now made it feasible to routinely perform whole-genome microarray studies where the behavior of every gene in a genome is captured. This increased availability of data promises a wealth of biological and medical information, but the increasing amount of data available also requires efficient computational algorithms for timely analysis. A new challenge has emerged to analyze and potentially annotate this massive amount of data, to develop intelligent computational tools to keep up with the pace at which new data is deposited to the repositories, and to make the information available to researchers to extract information pertaining to their respective interests.

1.0.1 Modularity

Coexpression analysis has shown that cellular functions are generally complex and require groups of genes to cooperate. Taken one step further it has been shown that functionality within a cell can be organized in terms of functional arrangements of genes, where groups of genes participating in the same function are called functional modules. Modular organization of genetic functions has been evident since 1999 (Hartwell, Hopfield et al. 1999; Ravasz, Somera et al. 2002; Ravasz and Barabasi 2003). Detecting these functional modules, learning how functions are carried out in detail and how individual functional modules relate to each other, are current research topics in this field (Barabasi and Oltvai 2004; Wu, Su et al. 2005).

Modularity is recognized as a common occurrence in all complex systems, and functional modules form a basis for the recovery of higher-level functional interaction networks. Complex interactions of cellular functions can be viewed as interaction networks of simpler subunits and functional modules. (Kholodenko, Kiyatkin et al. 2002; Ravasz, Somera et al. 2002; Friedman 2004; Petti and Church 2005). Understanding individual functional modules and interactions between modules will increase understanding of the inner workings of cells, and help in understanding where diseases such as cancer deviate from normal cellular operations.

The biological goal of detecting functional modules in microarray data sets is often formulated in terms of a bi-clustering task. Proteins that cooperatively perform the

same function in a cell, and experimental conditions under which the cellular function is performed, can be identified by the corresponding genes and experiments in a microarray data set. Various algorithms have been developed to perform this task. Some algorithms perform global clustering to identify potential functional modules in an entire data set (Herrero and Dopazo 2002; Grotkjaer, Winther et al. 2006). Global clustering is particularly useful when first analyzing new microarray sets to identify areas and genes of interest and in cases where there is little prior knowledge.

Localized clustering often consists of finding complete clusters based on a few known cluster members. The entire data set is still analyzed, but the goal is to find new information relating to the local cluster identified by the input gene set. This is a biologically relevant task where the data set is queried using functionally related (input, seed) genes as input in the study of specific cellular processes. Localized clustering algorithms are typically machine learning algorithms where the few known genes are used to train the algorithm to detect functionally related genes. The algorithm presented in this dissertation falls into the category of localized clustering.

1.1 The Problem

One aspect of interest for genetic data analysis is the function of genes in a cellular context. Complex cellular functions, such as the cell cycle, metabolism, or stimulus response and signaling are known to be carried out by multiple cooperating proteins: the functional modules (Hartwell, Hopfield et al. 1999; Kitano 2002; Ravasz, Somera

et al. 2002; Ravasz and Barabasi 2003; Spirin and Mirny 2003). The information necessary to build proteins is stored in the DNA in individual genes. Each gene carries information to build at least one protein (dependent on the organism), so functional modules can also be identified by correlated rates of expression of the respective genes (coexpression).

The goal for this dissertation research is to develop a novel algorithm to perform knowledge-based biclustering in order to analyze time-series microarray data sets with the primary purpose of detecting and verifying genes participating in the same cellular functions as a given set of input genes. This represents a localized clustering approach specifically on time-series data sets, where the clustering criteria are derived from a given set of input genes. The results of this algorithm lead to a better understanding of cellular functionality, and allow for the assignment of functional annotations to genes with few or no current annotations. The resulting groups of related genes produced by the algorithm may also serve as basis for the recovery of regulatory interaction networks, networks of interacting functional modules, to generate higher-level abstract views of the processes in a cell.

The machine learning approach used is related, in principal, to protein remote homology detection. It is capable of detecting weak similarity signals in data sets where other current algorithms produce no significant results, or where results degrade faster in the presence of high levels of noise. The open nature of the machine learning algorithm also allows for the study multiple heterogeneous data sets, or for gene lists based on different sources of similarity than cellular pathways, such as

functional annotations from Gene Ontology (GO) (Blake and Harris 2008), disease genes from Homophila (Reiter, Potocki et al. 2001), or localization in the cell, etc. This research will improve understanding of biological functionality and diseases, of the genes involved in these pathways, and it will enable the annotation of genes without prior functional annotations.

1.2 Contributions

The research for this dissertation has led to contributions towards several aspects if interest for the study of time-series microarray data sets. The next 4 sections in this chapter introduce an overview of the contributions; details are provided in the subsequent 3 chapters.

1.2.1 Feature Selection

One aspect of microarray data analysis is the notion that not all cellular functions are active at all times. Experimental parameters are usually set to study certain aspects of cellular functionality and behavior, but many other cellular functions continue to be carried out. Many of these cellular functions can be detected in the experimental data sets, but they may not all be active during the entire time period of the experiment. Also, not all cellular functions are even detectable in all microarray data sets. In extreme cases the feature selection step may suggest that a certain set of genes shows

no significantly correlated behavior between the input genes in a given microarray, in which case the machine learning step is unlikely to produce reliable results.

Furthermore, due to improvements in technology many microarray data sets are so large that it becomes infeasible to include all experiments (time steps) in the machine learning algorithm. Microarray data sets are represented as matrixes where rows represent genes or sections of genes, and columns represent varying experimental conditions. In time-series microarray data sets the experimental conditions in the columns refer to subsequent times during the same experiment when cell samples are collected. The terms *features*, *experimental conditions*, and *time steps* are used interchangeable in this context.

Two feature related selection algorithms are presented in this dissertation. They are called Functional Feature Reduction (FFR) algorithms, because their purpose is to reduce the number of features in the data set based on a set of input genes with shared cellular functionality. The set of features produced by the algorithm is then referred to as the Functional Features (FF).

One FFR algorithm globally selects a good set of *n* features from a data set, where *n* is a parameter selected prior to the algorithm run. This algorithm can be run even if the final correlation is very low. Details on this algorithm are provided in section 3.1.2.

A second FFR algorithm continuously selects features over a small sliding window, until the correlation for the current window falls below a threshold. The total

number of features includes the start window until the sliding window where the local correlation threshold falls below a threshold. The number of features depends of the quality of the data set and the seed genes. This window-based algorithm is particularly helpful for the combined analysis of multiple heterogeneous data sets of varying quality. Details on this algorithm are provided in section 5.1.3.

1.2.2 Linear Hidden Markov Model

The linear Hidden Markov Model (HMM) is the central component of the machine learning algorithm developed for this dissertation. It represents the general parameters within which genes of the same function operate. It is the combined profile that fits a certain set of input genes.

The HMM is trained with the gene expression values of the input genes over the functional features in a microarray data set. The linear HMM has as many hidden states as there are functional features to mirror the sequential nature of events captured in a time-series microarray data set. The trained HMM then represents the essence of the expected expression behavior for the function represented by the set of input genes. It is a constrained set of likely gene expression values based on the expression values of the input genes in the data set.

After the training step individual gene scores can be obtained from the HMM by calculating the probability that a given series of expression values (belonging to the

functional features of a gene) was generated by the trained HMM. The trained HMM represents the combined expression profile of a set of input genes.

1.2.3 Iterations for Hidden Markov Models

The combined profile of expression values for a set of functionally related input genes can be improved by feeding results back to the beginning of the algorithm and training a new generation of the Hidden Markov Model. This allows the profile to better incorporate expression profiles that were previously not contributing to HMM training, and to potentially exclude genes not fitting the dominant behavior found within the input gene group.

The underlying assumption here is that a set of input genes does not yet represent all genes in the data set participating in a cellular function. This means that there are genes whose behavior over the Functional Features will improve the HMM profile, producing better gene scores in the end. This approach is particularly helpful for small seed gene sets. Benefits of multiple iterations are demonstrated by using seed gene sets as small as a single gene in chapter 4.

1.2.4 Dynamic Data Set Combinations using linear HMMs

Increasing the amount of data available for analysis likely increases the power and performance of the algorithm. Due to the already large and still growing number of

microarray data sets deposited in public repositories, there are multiple microarray data sets available for almost every organism. Combining the data from multiple individual data sets has the potential to increase the power of the analysis algorithm.

Two approaches to integrating multiple separate, heterogeneous data sets are examined in this dissertation. A Meta Analysis approach combines the results of multiple individual data sets runs. The combination based on a majority-voting scheme using the individual result sets is shown to be beneficial in many cases.

A second approach stretches a single linear Hidden Markov Model across multiple data sets, analyzing the combined data set in one run. In this case only a single result set is produced, but the gene scores produced contain scoring elements from all data sets included in the analysis. This approach is particularly well suited for the linear Hidden Markov Model because data sets do not need to be normalized before they can be used in a combined analysis run. Details on this approach are described in chapter 5.

Each approach has distinct advantages, depending on the number and quality of data sets available.

1.3 Outline

This chapter has introduced the basic problem studied for this dissertation, and the contributions made to the field. Chapter 2 introduces the background for microarray bi-clustering analysis. It also discusses related concepts used in this dissertation, and reviews relevant approaches for the same problem in the current literature.

Chapter 3 introduces the linear HMM and demonstrates its performance relative to one of the high-profile approaches reviewed in chapter 2. This forms the base line of the performance that can be expected from the algorithm developed in this dissertation.

Chapter 4 introduces the concept of multiple iterations and demonstrates the performance gains obtained by it. This step is thoroughly compared to the current literature and performance is evaluated specifically on data sets used in the literature by competing approaches to exclude data set bias and to provide a firm basis for comparison.

Chapter 5 deals with the dynamic (at-run time, dependent on the input gene set) combination of multiple heterogeneous microarray data sets for analysis. In this chapter the benefits of the window-based feature selection algorithm in combination with a linear HMM spanning multiple data sets are demonstrated.

Chapter 6 ends with a discussion and provides an outlook on future directions for this research and the algorithm developed for this dissertation.

Chapter 2 Background and Related Work

Microarray gene expression technology captures the abundance of gene products for thousands of genes and multiple experiments in the same data set. The abundance of gene products under each experimental condition indicates the level of activity of a gene, which points to genes that are responding to the conditions set by the experiment.

Functionally related genes have been shown to exhibit some measure of similarity in their expression profiles; the expression profile of a gene is the set of all expression values recorded in a microarray for that gene. Standard analysis approaches for detecting functional modules are to cluster these genes using hierarchical clustering techniques (Eisen, Spellman et al. 1998; Grotkjaer, Winther et al. 2006), k-means clustering (Tavazoie, Hughes et al. 1999), techniques such as Self Organizing Maps (SOM) (Tamayo, Slonim et al. 1999), or combinations of these techniques (Herrero and Dopazo 2002). While this works well for smaller microarray data sets it has some limitations, specifically with larger microarray data sets (Ihmels, Friedlander et al. 2002). All clustering algorithms require a measure of similarity (distance metric) between individual genes to be able to perform clustering. Most clustering techniques base their distance metric on the entire range of experimental conditions, when typically only a subset of conditions is responsible for the functional similarity; and most standard clustering techniques do not take into consideration that genes can participate in multiple modules, which nearly every gene does.

Coupled two-way clustering (CTWC) (Getz, Levine et al. 2000) introduced the notion of bi-clustering, that functional clusters contain subsets of genes and experiments. This adds a feature selection component to the overall algorithm, but it also increases the amount of information produced by the algorithm – a list of genes with similar function, as well as the experimental conditions where the common function is active. Most of the supervised learning algorithms perform some form of bi-clustering, e.g. by (Ihmels, Friedlander et al. 2002; Tanay, Sharan et al. 2004; Wu, Su et al. 2005), including the algorithm developed in this dissertation.

Similar localized clustering work is also performed at the level of proteins, using protein-protein interaction data (Snel, Bork et al. 2002; Spirin and Mirny 2003; Pereira-Leal, Enright et al. 2004; Dittrich, Klau et al. 2008). Most algorithms in this category are graph algorithms operating on known protein interaction networks. Several promising approaches have also been undertaken in combining both protein-protein interaction data and gene microarray expression data to derive functional modules at both levels (Tornow and Mewes 2003). Other approaches integrate yet more data, such as growth phenotype data, transcription factor binding sites (Tanay, Sharan et al. 2004) or physical interactions, known protein functions, and network topology characteristics (Wong, Zhang et al. 2004) across multiple diverse data sources.

2.1 Modularity at the Cellular Level

2.1.1 Background

There is a good amount of research analyzing the functionality of genes, not all of them based on microarray gene expression data sets. This review presents some background on the functional analysis of genes to better place the research in the landscape of current activities. Research relevant to this dissertation is then presented in more detail.

Modular organization of biological processes at the molecular level has been observed since 1999 (Hartwell, Hopfield et al. 1999). Hartwell et al. makes an interesting comparison to computer logic with reference to the biological "logic circuits" within the cell. This approach is also suggested by (Yuh, Bolouri et al. 1998) and is later picked up by several studies that interpret biological activity in terms of logic gates (Buchler, Gerland et al. 2003; Istrail and Davidson 2005; Hermsen, Tans et al. 2006). The main idea of (Hartwell, Hopfield et al. 1999) is that the functions carried out in a cell are accomplished by multiple cooperating genes and proteins, and that these cooperating groups are organized in a modular fashion, called "functional modules." This forms the basis for the research pursued for this dissertation.

Modular organization within the cell exists at the level of proteins and at the level of genes. At the protein level (Snel, Bork et al. 2002; Spirin and Mirny 2003; Barabasi and Oltvai 2004; Pereira-Leal, Enright et al. 2004; Dittrich, Klau et al. 2008) groups of proteins interact closely to bring about certain cellular functions. Here

modules are primarily detected based on graph-theoretical algorithms working on protein-protein interaction networks. Proteins form the nodes of these networks and edges indicate evidence that two proteins interact. Functional modules at this level are characterized by a very high number of interactions between proteins within a module compared to a lower number of interactions outside of the module. Modules are detected based on the density of the interaction network.

At the level of gene expression and regulation algorithms primarily use microarray gene expression data sets where the 'activity' (rate of expression) of genes is measured for multiple genes under multiple experimental conditions. In these data sets the rates of gene expression of genes participating in the same cellular function are related (e.g. (Jin, Rabinovich et al. 2006; Li and Zhan 2008). There are several approaches to module detection at this level, and there are several definitions of what constitutes a module as well. The next section explores the usage of this term in literature.

2.1.2 Definition of Terms: Modules

A common definition takes a module at the genetic level to contain all genes regulated by the same set of transcription factors, or the same *cis*-regulatory elements (transcription factors bind to *cis*-regulatory elements); similar *cis*-regulatory elements indicate similar function (Bansal, Belcastro et al. 2007). These modules are often called transcriptional modules or gene regulatory networks (Hughes, Estep et al.

2000; Jin, Rabinovich et al. 2006; Yan, Mehan et al. 2007) etc. in the literature. These approaches assume that similarities in the levels of gene expressions are related to the same transcription factors, because the activity of transcription factors directly influences the level of activity for a gene.

- (Jin, Rabinovich et al. 2006) uses prior knowledge in the form of experimentally determined transcription factor binding sites and learns weight matrices to find other binding sites in the genome. This approach uses chromatin immunoprecipitation microarray data (ChIP-chip) to detect protein-DNA interactions of the transcription factors.

- (Yan, Mehan et al. 2007) uses graph-based data mining on multiple microarray gene expression data sets to find clusters of co-expressed genes. The results obtained from the algorithm are verified by ChIP-chip data to demonstrate that the modules found are transcription regulatory modules. These modules are defined as sets of genes regulated by a common set of transcription factors.

- (Hughes, Estep et al. 2000) uses a Gibbs sampling algorithm ("AlignACE") to detect sequence motifs found in upstream regions of apparently co-regulated genes. Analyzing the entire genome for these motifs finds genes that tend to have similar functions.

However, *cis*-regulatory regions in the genome tend to be quite complex (Hermsen, Tans et al. 2006). One interesting approach that deals with this complexity interprets

the activity of transcription factors analogous to the work of logic gates (Buchler, Gerland et al. 2003; Istrail and Davidson 2005; Hermsen, Tans et al. 2006). This takes into account that multiple transcription factors tend to work on the same *cis*-regulatory region with some transcription factors promoting gene expression, some repressing it, and some genes being activated only if multiple transcription factors are present.

As a representative of this approach, (Hermsen, Tans et al. 2006) treats *cis*-regulatory regions as logic components that integrate multiple cellular signals. This has revealed mechanisms of interplay between transcription factors, allowing for a simpler view of the architecture of complex *cis*-regulatory regions. Modules, in this approach, are taken to be the genes activated by the same signals.

All these approaches treat modules as the set of genes regulated by the same *cis*-regulatory elements. A different, also commonly used definition of a module at the genetic level is to view all genes with the same function (e.g. based on GO term annotations) (Yi, Sze et al. 2007) or belonging to the same pathway (e.g. in KEGG) to be a functional module (Hirose, Yoshida et al. 2008). This does not assume that genes with the same function or the same gene expression profiles are regulated by the same transcription factors. This is the definition used in this research proposal. Relevant research using this definition of a functional module is described in more detail the following sections.

- (Hirose, Yoshida et al. 2008) defines transcriptional modules as sets of genes sharing a common function or involved in the same pathway. Modules are detected from time-series microarray gene expression data sets based on a state space model. This allows for the detection of individual modules ad well as networks of modules. This is the definition also used in this research proposal.

- (Pang, Lin et al. 2006) points out that pathway analysis (analyzing microarray data sets based on genes sets belonging to a pathway, such as KEGG) provides much relevant information to biological researchers. (Pang, Lin et al. 2006) uses Random Forests to rank order pathways from external sources, such as KEGG, in microarray data sets. The results allow for classification of disease phenotypes.

In many cases these two definitions of what constitutes a functional module overlap, as it is often assumed that the functional behavior of genes is directly related to similarities in the *cis*-regulatory element, that the same transcription factors activate genes with a similar function. This dissertation does not make this implicit assumption; this information is also not directly available in microarray gene expression data sets, but comes from external data sources.

The general definition of what constitutes a functional module in this dissertation is taken to be a set of functionally related genes and the experimental conditions where cooperation between these genes can be detected. The source for input gene

sets can be taken from genes with common GO term annotation, genes in the same KEGG pathway, or simply come from the literature as genes of interest to individual researchers.

2.1.3 Microarray Data Sources

There are several sources of data used in the detection of functional modules. DNA sequence is used to find *cis*-regulatory regions and transcription factor binding sites based on sequence similarity. ChIP-chip data sets capture protein-DNA interaction data used for a similar purpose, finding regions of the DNA where transcription factors bind.

Microarray gene expression data captures the abundance of gene products for thousands of genes and multiple experiments in the same data set. The abundance of gene products under each experimental condition indicates the level of activity of a gene, which in turn indicates which genes are responding to the conditions set by the experiment. If experimental conditions remain constant and gene expression levels are sampled at multiple time intervals during the experiment, the resulting microarray is a time series microarray. Time series microarray data is very useful in observing the progression of complex biological functions over time.

Several approaches attempt to combine multiple sources of available data, including gene expression microarrays, protein-protein interaction data, transcription

factor binding sites, etc. These approaches (e.g. (Bar-Joseph, Gerber et al. 2003; Tanay, Sharan et al. 2004; Wong, Zhang et al. 2004)) search for groups of genes that simultaneously share close similarities in several data sets in an attempt to reduce false positives and to compensate for weaknesses (e.g. noise) in any one data set. This presents a possible future direction of the research pursued here.

- (Tanay, Sharan et al. 2004) presents an unsupervised algorithm that integrates gene expression microarray data sets, protein interaction networks, growth phenotype data, and transcription factor binding sites in an analysis framework named SAMBA (Statistical-Algorithmic Method for Bicluster Analysis). This algorithm selects genes that show statistically significant correlated behavior across multiple data sources on yeast data. The results show a very interesting hierarchical set of functional modules that allow for the generation of a network of clusters, presenting a high-level view of abstraction of the processes in a cell.

- (Wong, Zhang et al. 2004) uses probabilistic decision trees over multiple data sources to detect synthetic sick or lethal (SSL) genetic interactions. Data sources used are localization information, mRNA expression, physical interaction, protein functions, and characteristics of network topology, for a total of 122 individual characteristics extracted from these sources.

- (Bar-Joseph, Gerber et al. 2003) describes the GRAM (Genetic Regulatory Modules) algorithm to discover regulatory networks of gene modules. This approach combines co-expression in microarray gene expression data sets with transcription factor binding site data. Functional modules detected by GRAM simultaneously are constrained by similarities in expression profiles and by a common set of transcription factors. A functional module is seen to be a set of co-expressed genes to which the same transcription factors bind. This combination of gene expression microarray data with transcription factor binding site data is used in several papers (e.g. (Li and Zhan 2008)).

In another interesting approach (Li and Zhan 2008) proposes an algorithm to detect transcriptional modules using microarray gene expression data to detect modules. Here the microarray is treated as a matrix. This matrix is decomposed in two stages: first non-linear independent components are extracted from the original gene expression matrix, and then this matrix is approximated by the product of a sparse matrix and a low-rank matrix. Modules are extracted from the final set of matrixes. In later work (Li and Zhan 2008) additional information in the form of transcription factor binding data is added to this algorithm, similar to (Bar-Joseph, Gerber et al. 2003).

2.1.4 Supervision

Microarray analysis can be classified primarily into two categories: unsupervised and supervised. Unsupervised clustering algorithms that look at the microarray without any prior information and attempt to detect clusters based on a similarity measure, supervised machine learning algorithms where the classification function is learned or constrained, at least in part, based on prior knowledge.

All common clustering algorithms may be applied to microarray data sets, such as singular value decomposition (SVD) (Alter, Brown et al. 2000), self-organizing maps (SOM) (Kohonen 1987; Tamayo, Slonim et al. 1999), K-Means clustering (Tavazoie, Hughes et al. 1999), superparamagnetic clustering (SPC) (Blatt, Wiseman et al. 1996), etc. These algorithms cluster genes into individual groups based on a distance metric, typically the similarity of gene expression profiles between genes.

(Eisen, Spellman et al. 1998) is a notable work introducing the still-used Eisen graph that hierarchically clusters and arranges genes based on similarities of the expression profiles. This approach hierarchically clusters the entire microarray without the need for prior information or knowledge.

Early classification approaches using these techniques had the drawback of classifying genes exclusively into one cluster, while it is much more common that genes participate in multiple functions, and thus need to be part of multiple clusters. Later approaches take this into account to produce overlapping clusters.

Supervised algorithms start with prior knowledge that is used to direct the way clusters are generated. Prior knowledge comes in the form of a set of genes with known or desired similarities, or many other sources, even outside of the microarray data set itself. Many of the algorithms already mentioned are supervised algorithms, e.g. (Hughes, Estep et al. 2000; Yi, Sze et al. 2007).

(Brown, Grundy et al. 2000) uses Support Vector Machines (SVM) on yeast data to identify genes with common function. The SVMs are trained from genes with the function that is searched. One of the algorithms (Mateos, Dopazo et al. 2002) related to this dissertation research is closely based on this paper.

(Pena, Bjorkegren et al. 2005) is another example that uses prior knowledge in the form of seed genes to build a Bayesian network (BN) over gene expression microarray data to find genes closely related to the seed genes. This algorithm, AlgorithmGPC, uses seed genes belonging to the iron homeostasis pathway in yeast to prove its performance. The idea of seed genes that represent genes with a common function or as part of the same pathway is also used in this dissertation research.

This dissertation uses prior knowledge in the form of gene lists belonging to the same pathway (KEGG), to the same functional category (GO), or assembled by a researcher for a specific experiment to train a machine learning algorithm, and can thus be seen as a supervised algorithm.

2.1.5 Bi-clustering, Two-Way Clustering

Biclustering performs a selection of subsets on both axes of microarray data sets, selecting subsets of genes as well as subsets of experimental conditions. (Getz, Levine et al. 2000) realized that only a subset of experimental conditions participates in any cellular process of interest, rendering the remaining conditions merely as noise.

The coupled two-way clustering (CTWC) algorithm described in (Getz, Levine et al. 2000) is an iterative unsupervised algorithm that clusters subsets of genes and subsets of conditions alernatingly to produce small pairs of genes and conditions in the microarray corresponding to biologically relevant information. CTWC refers to an architecture than can be used with any reasonable clustering technique, including the algorithms described in section 2.1.7. Results in (Getz, Levine et al. 2000) are prepared using superparamagnetic clustering (SPC).

(Wu and Kasif 2005) introduce the Gene Expression Mining Server (GEMS) as a web server that allows users to upload both a gene expression microarray data set and limitations upon which GEMS performs biclustering using a Gibbs sampling paradigm. Unlike with other algorithms the clusters found are limited by a certain distance measure to find localized bi-clusters in the microarray data set. This is an unsupervised clustering approach.

This idea of selecting subsets of experimental conditions along with subsets of genes is picked up by many of the algorithms described in this review (e.g.

(Bergmann, Ihmels et al. 2003; Tanay, Sharan et al. 2004; Dhollander, Sheng et al. 2007). The algorithm in this dissertation also follows this lead.

2.1.6 Combined Microarray Data Set Analysis

The amount of information increases the more data one has available. The bigger the data sets used for the analysis algorithms developed in this dissertation, the more likely the results will be useful. Two fundamental approaches have been pursued. The first way is to increase the amount of information available to individual runs of the analysis algorithm. This is achieved by combining multiple microarray gene expression data sets into one larger data set prior to analysis. There are several published methods on the combination of microarray data sets, such as:

- MEFIT is a system to perform functional analysis based on integrated microarray data sets. The publication (Huttenhower, Hibbs et al. 2006) describes a Bayesian Network based integration method that is validated by higher performance in functional analysis later in the same publication.

- Integration methods based on a set of input genes. This approach ensures that correlation between a set of input genes is preserved, as much as possible, across multiple microarrays. Values are scales in such a way as to preserve this similarity.

One of the fundamental problems with data set integration is in the way microarrays are generated. Each microarray is produced under different experimental parameters, which makes a direct combination of raw data sets difficult. Data sets must first be normalized and pre-processed to remove differences introduced by the experimental parameters.

The second approach integrates the results obtained from analysis runs on individual data sets in an ensemble approach of existing methods. Ensemble methods have been shown to work well in machine learning applications, because they avoid the impact of weaknesses of any one combination method. The end result of these methods will be one result data set based on information from multiple microarray data sets.

This dissertation examines both approaches; Meta-Analysis is used to produce high-quality results in chapter 4 and data set combination is used in chapter 5 to dynamically combine a large number of microarray data sets in a single analysis run.

2.1.7 Relevant Research to this Dissertation

This dissertation specifically uses prior knowledge in the form of genes known to be participating in the same function. The goal is to train the machine learning algorithm, a linear profile Hidden Markov Model (HMM) using the selected input genes. The algorithm then used the trained model to detect similar genes from the remaining genes in the microarray data set. Very few papers in the published literature use this

semi-supervised approach to detect functionally related genes or to bi-cluster microarray data sets. The following paragraphs introduce a survey of the literature for comparable approaches.

Bayesian Network Biclustering based on Query Genes

(Dhollander, Sheng et al. 2007) uses a Bayesian query-driven biclustering (QDB) approach that allows a data set to be queried using a set of seed genes with known similarities as input. The output is a set of genes with the same functional characteristics and a set of experimental conditions where these genes are related.

This algorithm gives a biological researcher the opportunity to "query" a microarray data set based on a given set of genes with known function or common biological pathway. The expression patterns of these genes are used to restrict the search space, assuming that the clusters conform to similar expression patterns. This prior knowledge is used to initialize the prior probabilities of the Bayesian network (BN).

(Iterative) Signature Algorithm

(Bergmann, Ihmels et al. 2003) develops the (Iterative) Signature Algorithm, which is a continuation of the Signature Algorithm introduced in (Ihmels, Friedlander et al. 2002). This algorithm works on gene expression microarray data sets to extract

functional modules consisting of subsets of genes and experimental conditions, using a set of seed genes as input. The seed genes are a set of genes known to have a similar function or part of the same pathway.

This algorithm first selects a suitable subset of experimental conditions where the set of seed genes show statistically significant rates of gene expression activity compared to the rest of the microarray. Genes are then scored over just the subset of conditions and genes with statistically significant scores are added to the result set. Similar to (Dhollander, Sheng et al. 2007), functional modules are defined to be subsets of both experimental conditions and genes, providing information not only about genes related to a function or pathway, but also about the conditions where this function or pathway is active.

This algorithm is similar to this dissertation in its architecture of first selecting a subset of experimental conditions and then a subset of genes, based on the reduced condition set. However, the machine learning algorithms used are different.

Gene Recommender

The Gene Recommender (GR) algorithm (Owen, Stuart et al. 2003) specifically attempts to discover genes with similar function to a set of query genes. Results of this algorithm rank genes according to how strongly they correlate with the query (seed) genes.

The query genes are genes with a common function of interest; a group of query genes is referred to as “cassette” in this paper. High-ranking result genes are genes with a high likelihood of the same function as the cassette genes; results are likened to the results of a web query. Similar to other biclustering approaches, this algorithm first selects a subset of experimental conditions from the microarray data set. The architecture of this algorithm is similar to the Signature Algorithm.

To determine if a set of input genes is actually co-regulated, correlation is compared to a random group of genes of the same size. Useful results can only be expected if the input genes show an above-random level of correlation.

The condition (experiment) score is calculated as a normalized average score over the non-missing values of the input genes. Informative conditions are those with scores far from zero, which is a preference given to high gene expression values. Once conditions have been selected, genes are only scored over the subset of conditions. A gene score, *S*, is calculated as the mean over the condition subset for that gene. The actual significance score, *Z*, produced for each gene is calculated as the quotient of that score over a constant derived from the all seed genes. Genes in the microarray data set are rank ordered according to the score *Z*, and all genes above a threshold are included in the result set.

Gene Recommender and the Signature Algorithm are the most closely related algorithmic architectures to this dissertation.

Supervised Neural Network Approach

(Mateos, Dopazo et al. 2002) proposes the use of Supervised Neural Networks (SNN) for the detection of gene functions. Neural networks (MLP, Multi Layer Perceptrons) are trained on functional classes of genes, where each class contains genes of a common pathway or with the same biological function, such as the TCA cycle.

Each input neuron corresponds to one experimental condition in the microarray gene expression data set; in this case there are 79 input neurons. Heuristics and tests determined the use of 8 hidden neurons. This network is then trained with each of the functional classes to be detected. The neural network contains as many output neurons as there are classes to be distinguished. The results of this approach compare favorably with (Brown, Grundy et al. 2000) upon which this paper was inspired.

An interesting aspect of this study is the exploration of the limitations of learning functional classes from gene expression microarray data sets. While this is limited to the results of one data set and doesn't claim to be true for the general case and other data sets, it found that the "learnability" is related to the structure of the catalog of the biological function, to the heterogeneity of the expression profile within the class, and to the size of the class.

2.2 Challenges Addressed

2.2.1 Gene Set Functions

One challenge for any machine learning algorithm is related to the kind of prior knowledge used. The class of algorithms to which this dissertation contributes uses a set of input genes used to train models of machine learning algorithms. These genes represent knowledge of common gene function or pathway involvement. Each microarray data set contains the levels of gene products for certain experimental conditions. Prior work has established a need to inspect a given set of genes with respect to the microarray data set used, and to select a subset of experimental conditions, because gene functions are typically not active over the entire range of experiments. This observation has introduced the biclustering approach.

A particular set of genes may not be correlated at all in a given data set, or that only a subset of the set of seed genes shows significantly correlated behavior in the seed gene set, with other genes only adding noise to the model to be trained. If genes show no correlated or coexpressed behavior, then running a machine learning algorithm on that data set is not beneficial. The more interesting case is when there are correlated subsets in the input gene data, and there are two basic solutions in this case: one solution is to add a further preprocessing step to identify subgroups within the input set and divide the seed gene set into subgroups. This approach is shown to work well in section 3.2.3. This should be the approach taken for large input gene sets.

The second solution lies in the repetition of the algorithm with iterative refinement of the input set. This approach is explained in detail in chapter 4.

The actual biological function of a set of gene bears some closer inspection. The vocabulary predominantly used to describe the function is the Gene Ontology (GO) (Ashburner, Ball et al. 2000). GO ascribes annotation terms for genes belonging to three categories: biological process, molecular function, and cellular component. Determining if a set of genes has a common biological function can done by calculating a p-value that a certain biological process term occurs repeatedly in that set (Bauer, Grossmann et al. 2008). This approach is used in GR, ISA, QDB, and in this dissertation. Functional modules then refer to related (functionally and expression) gene sets.

An alternate approach looks at whole sequences of events that make up a complex biological function (biological pathways) or protein complexes that combine to form a part of molecular machinery within a cell. This data is available in the KEGG database (Kanehisa and Goto 2000). Genes in the same biological pathway or protein complex will usually be annotated with closely related GO terms, but genes within the same pathway do not necessarily exhibit correlated gene expression values. KEGG additionally provides pathway maps to describe the temporal and causal interaction between related genes.

2.2.2 Handling Missing Data

Experimental microarray data sets often contain missing values stemming from errors or contaminations in the experiment that render parts of the results unusable. However, statistical analysis algorithms work best with complete data sets. While missing data can be modeled in the statistical tests, more often the experimental data sets are pre-processed to eliminate gaps as much as possible.

The options are to have the HMM model missing values, or to pre-process the data set to exclude and impute missing values. Literature review did not indicate a clear advantage for either method. In the end the approach taken in this dissertation is to pre-process the data sets used. Pre-processing the data set ensures that the machine learning algorithm can assume a complete data set and that as many data items (genes) as reasonably possible are included in this set.

The algorithm in this dissertation imputes missing values using a KNN-based method (Troyanskaya, Cantor et al. 2001). Missing values are estimated to be the average of the five closest genes, as measured by root mean squared distance (RMSD) in the microarray. Once the data set is complete it is passed on to the input gene pre-processing algorithm. This method provides a reasonable improvement in the number of usable genes from the microarray, compared to the alternative of removing all genes with any missing data. KNN-impute is also used by other algorithms working on microarray data sets, such as the microarray integration method described in (Huttenhower, Hibbs et al. 2006).

Other approaches offer slightly higher performance at the cost is increased complexity, such as a method of selecting features in microarrays proposed by (Schafer and Strimmer 2005). The purpose of this dissertation is not to advance the state of the art for missing value imputation methods. Instead existing and proven methods to increase the quality of the data sets are used with the analysis algorithms developed in this dissertation.

2.2.3 Data Set Integration

The problem with combining microarray data directly is that each data set is generated with different experimental variables, so the results are not directly comparable. Machine learning algorithms may pick up upon the different biases within each data set, or extreme values in one data set may dwarf contributions from other data sets.

Ordinarily it is essential that the individual data sets to be combined are first adjusted so that they are comparable. This task is most often performed by normalizing the data sets; among normalization schemes the Z-score normalization is an often-used first step. This scales expression values so that the average value is 0 and the standard deviation for the values of a gene is 1.

Beyond Z-score normalization data set integration is a branch of research in its own right, eg. (Huttenhower, Hibbs et al. 2006). This dissertation takes advantage of

the special topography of the linear HMM to enable use of varied microarray data sets without the need for prior normalization. Details are provided in chapter 5.

2.2.4 Computational Complexities

The analysis algorithm developed in this dissertation treats gene similarities not necessarily based on shared extreme (high or low) gene expression values in the microarray data set, but instead based on the similarity of behavior (expression values) of the genes at various experimental conditions. This similarity search works even with genes that do not share common conditions with extreme expression values. This approach is shown to work well in a wider variety of data sets compared to competing approaches in the current literature. The primary challenge of this approach is that of asserting the significance of a gene score produced in this environment. Unlike approaches looking at extreme values there is no absolute measurement in this case.

Computational complexity in the algorithm developed for this dissertation stems primarily from the calculation of P-values for each gene score. P-values are generated by calculating a probability density function for each gene based on gene scores produces by random gene input sets. The more scores are available, the better the P-value. However, each random score requires the same amount of time to generate as the original gene score; providing 400 random scores increases the run time of the algorithm 400-fold.

This problem is trivially parallelizable, and is perfectly suited for massively parallel hardware, such as GPGPU (General-Purpose Graphics Processing Unit) computing and implementation via OpenCL or CUDA; this is an emerging technology used in many compute-intensive applications (Manavski and Valle 2008; Trapnell and Schatz 2009; Davis, Pandey et al. 2010; Shi, Schmidt et al. 2010; Suchard, Wang et al. 2010). In this dissertation implementation is limited to multiple threads in a CPU, leaving higher-performance CUDA implementations for future work.

The second contributing factor to the complexity of the algorithm developed in this dissertation is the size of the linear HMM. Complexities for training and probability calculation are directly dependent on the number of states in the model. The algorithms (Baum-Welch, Forward-Backward FBA) used for these tasks do not offer much potential for parallel implementations. This challenge is addressed in this dissertation by performing feature selection to reduce the number of features selected from the data set and to generate a smaller model.

High computational requirements are an issue with many machine learning approaches involving potentially massive data sets. It presently remains a challenge to realize the inherent parallelism in many of the machine learning approaches to handle very large data sets efficiently.

2.2.5 Other Comments

In the post-genomic era with the availability of whole genomes there are still many genes without any functional annotation. Understanding the function of genes is an important step towards deciphering the processes and higher-level "programs" of cells. This is important basic research, but it also enables us to understand diseases such as Cancer much better, which work based on small, but important, alteration in these cellular programs.

Another challenge is the detection of very weak similarity "signals" in existing data sets. The algorithm developed in this dissertation is well-suited to detect similarity signals obscured by noise and promises to be able to detect weak signals in biological microarray data sets.

It is important to extend the search for gene functions to cover a wider range of genes, to derive annotations for more genes. Many current approaches do not look for the "weak" similarity signals that still produce useful results. And the three issues mentioned by(Mateos, Dopazo et al. 2002), (1) class size, (2) heterogeneity of classes, and (3) high degree of overlap, also remain challenges.

Chapter 3 Linear Hidden Markov Models

The heart of this dissertation research is a Hidden Markov Model based algorithm to select functionally related genes from microarrays based on expression patterns learned from a set of input genes (Senf and Chen 2009). Most of the existing algorithms for this task are limited by considering only genes (without selection of experimental conditions), or by focusing on genes with extreme (high or low) expression levels.

Functional similarities are not always characterized by consistent sections of extreme expression levels, however, and in application to biological data sets, including *D. Melanogaster* and *S. Cerevisiae*, this approach did not lead to biologically meaningful results. The algorithm developed in this dissertation is shown to perform well regardless of the nature of the similarity between functionally related genes.

3.1 Algorithm

The algorithm is designed to work with input genes based on common KEGG (Kyoto Encyclopedia of Genes and Genomes) pathway (Kanehisa and Goto 2000), common functional properties, common Gene Ontology (GO) annotations (Ashburner, Ball et al. 2000), or based on the literature. It proceeds in clearly defined steps:

1. A microarray data set is acquired from a public source and prepared using published data preprocessing and cleansing techniques to produce a complete expression matrix with no missing values. Missing value imputation increases the amount of information available to the analysis algorithm.

2. Input genes are selected from the literature or from publicly available databases. Genes with an average pairwise correlation coefficient above a certain threshold in the microarray are selected for HMM training. Microarrays are often generated to study gene responses under specific conditions, and not all sets of related input genes show significant similarities in all microarrays. This step filters out genes where the similarity relation is not captured well in the selected microarray.

3. A novel algorithm is used to select a relevant subset of experiments from the microarray data matrix that best represent the input genes. Similarities in gene behavior generally do not extend to all experiments in a microarray. This step filters out experiments where there is low similarity between the selected genes. All experiments not in the subset are discarded from the expression matrix. This step filters out experimental conditions where the input genes are less likely to cooperate.

4. A profile Hidden Markov Model (HMM) is trained on the input data set. Additional sets of profile HMMs are trained from random sets of genes of the same size as the input gene set to enable an estimation of the significance of

HMM scores. These scores are used to find new genes related to the input genes.

5. All genes in the microarray are scored with each trained HMM. The random-gene-trained HMM scores are used to estimate the Parzen density function (PDF) (Parzen 1962). New genes from the microarray are added if the score from the input-gene-trained HMM is statistically significant, given the PDF for that gene. This step filters out genes not related to the same source of similarity selected for the input set. Generating the PDF is further discussed in section 3.1.4.

A graphical outline of the algorithm is presented in Fig. 1. The following sections describe each step of the algorithm in more detail.

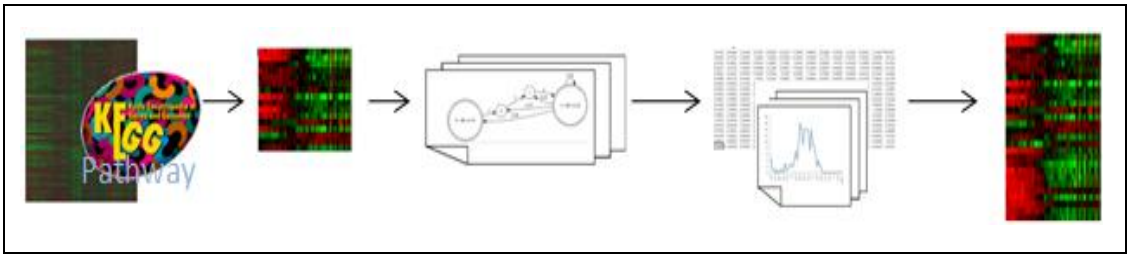

Fig. 1: Algorithm outline: given a microarray and one pathway from the KEGG database, Functional Features (subset of features where pathway genes show very high correlation) are calculated. This input set, along with multiple random-generated input sets with the same number of genes, is used to train HMMs, one HMM per input set. All genes in the microarray are scored with all HMMs and the PDF for each gene is derived. Genes with significant scores from the input-gene-trained HMM are included in the result set.

3.1.1 Microarray data sets and data preparation

While much work still remains to be done, microarray technology has already proven to be useful in two areas of interest. First, finding genes with similar rates of expression over multiple experiments allows for the characterization of functional modules comprised of genes/proteins cooperatively necessary to perform complex cellular functions (Bergmann, Ihmels et al. 2003; Dhollander, Sheng et al. 2007; Li, Ma et al. 2009). Second, differential analysis allows or the comparison of the same types of experiments run on samples taken from normal vs. diseased tissue to detect genes which behave differently (Baggerly, Coombes et al. 2001; Park, Yi et al. 2003; Yang, Zou et al. 2009). Both approaches provide information useful in areas like the development of medicine targeting specific genes of interest.

There are several databases publishing microarray data sets. Prominent among them are The European Bioinformatics Institute's (EBI) ArrayExpress (Parkinson, Kapushesky et al. 2007; Parkinson, Kapushesky et al. 2009) and the National Center for Biotechnology Information's (NCBI) Gene Expression Omnibus (GEO) data base (Edgar, Domrachev et al. 2002; Barrett, Troup et al. 2009). The dataset used in this chapter is selected from the GEO data base, accession number GDS191, the Life Cycle of Drosophila (Arbeitman, Furlong et al. 2002).

After pre-processing the data set to remove and impute missing gene expression values a subset of genes is selected with an average absolute pairwise correlation coefficient above a certain threshold. The commonly used Pearson correlation

coefficient is used to calculate the similarity between two gene expression profiles. The input for this algorithm is a set of genes known to be involved in the same pathway or cellular function. For the analysis performed in this chapter a list of input genes list from the KEGG pathway data base is used.

3.1.2 Functional Feature Reduction

Cellular functions do not typically span the entire set of experiments covered by a microarray. The first step in the analysis then is to detect a reasonable subset of experiments in which the input genes show the greatest level of similarity, as measured by the average absolute pairwise correlation coefficients. This algorithm chooses *n* Functional Features from a microarray with *m* experiments where genes are highly correlated, as outlined in Alg. 1.

Alg. 1 converges when the set of experimental conditions 1 – *n* remains unchanged after one loop iteration. A number of $n = 60$ experiments is chosen initially to represent the Functional Features in the microarray, reducing the dimensionality of the data set to those experiments where the average pairwise correlation of the input genes is largest. The set of input genes, now each 60 elements long, is then used to train a Hidden Markov Model. The actual choice of number of features *n* is less important, because the HMM model used in the next step can mask areas of low similarity between the input genes to a certain degree.

Algorithm 1

```
1:  Repeat until convergence
2:    Select experimental conditions 1 – n
3:    for i = 1 … n
4:      calculate average pairwise correlation, c, over 1 … n
5:      for j = n+1 … m
6:        exchange experimental features i and j
7:        calculate average absolute pairwise correlation, d
8:        if d < c then
9:          reverse, exchange features i and j again
10:       Else
11:         keep features exchanged
12:       end if
13:     end for
14:   end for
```

3.1.3 Functional Learning using Hidden Markov Models

Hidden Markov Models are machine learning tools for modeling hidden (unobservable) generative processes from observable events (Baum, Petrie et al. 1970). The hidden process in an HMM is assumed to be a 1^{st} order Markov Chain (Markov assumption). Parameters are learned from given sequences of observable events. A trained HMM can generate observation sequences similar to the training data, and for a given sequence the probability that this sequence was generated by the HMM can be calculated. HMMs are a widely used machine learning tools especially in areas of gene and protein sequence alignments, e.g. (Karplus, Barrett et al. 1999).

HMMs are essentially finite state machines that produce output symbols according to an emission probability distribution B at each state ($B=\{b_i\}$, $b_i=P(O_t=v_k|X_t=a_{it})$). In the most general case, each state may transition to any of the other states in the HMM,

according to a state transition probability A ($A=\{a_{ij}\}$, $a_{ij}=P(X_{t+1}=a_j|X_t=a_i)$). Due to the Markov assumption each state transition a_{ij} only depends on the current state a_i. Each state may be chosen as the starting state according to the initial state distribution π (π, $\pi_i = P(X_0 = a_i)$). The formal definition of a HMM λ is the set of all parameters:

$$\lambda = (A, B, \pi) \tag{1}$$

The set of all observed events is the state alphabet set of the HMM and is denoted by *S*; *N* is used to refer to the number of states in the HMM. The state alphabet set is a subset of the observation alphabet, *V*, of all events that could possibly be observed. (Observed events are also referred to as symbols, *S*, from the alphabet):

$$S = (s_1, s_2, \ldots, s_N) \tag{2}$$

$$V = (v_1, v_2, \ldots, v_M) \tag{3}$$

M denotes the number of discrete observable events. In the most general HMM model the length of an observation sequence, *T*, can be shorter or longer than the number of states. An actual series of observations is denoted by *O*:

$$O = (o_1, o_2, \ldots, o_T) \tag{4}$$

HMMs are usually represented by a directed acyclic graph (DAG). An example of a general HMM with three states is given in Fig. 2. At each state, *i*, a symbol from the state alphabet is generated with a probability b_i:

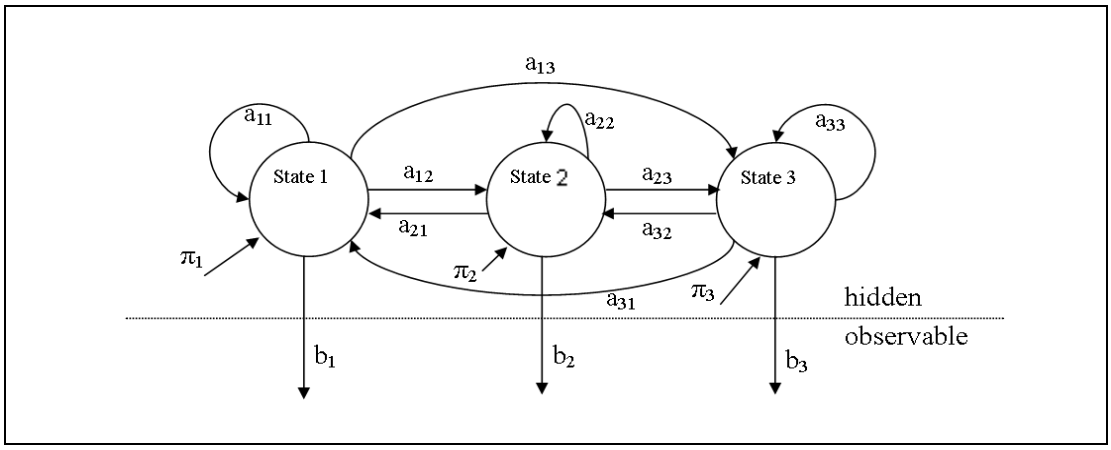

Fig. 2: a three-state HMM. Each variable represents a parameter to be learned. The only visible events are the emissions produced by of the model. Training adjusts al parameters to maximize the probability that the model will produce the sequences of observations used to for training.

Sequential applications of HMM often use profile HMMs (Eddy, 1998), special-purpose models with additional restrictions. Profile HMMs are very popular in remote homology detection (HMMER (Wistrand and Sonnhammer 2005), SAM (Karplus, Barrett et al. 1999), etc.), sequence alignments, and other sequential applications such as speech or handwriting recognition (Rabiner 1989).

Profile HMMs have at least a matching number of states and observation sequence events ($N \geq T$). The starting state is always the first node ($\pi_1 = 1$, $\pi_2, \ldots, \pi_N = 0$), and the state transition probability is set to 1 for the next step in the sequence ($a_{ij} = 1$, $j = i+1$; $a_{ij} = 0$, $j \neq i+1$). Each step may have multiple states to better model domain-specific occurrences such as deletions or insertions in protein sequence alignments, which would provide three states per step. If each step contains multiple states, then the sum of probabilities between nodes for increasing steps is 1.

The application of interest in this dissertation is to find genes that are functionally related based on similar expression patterns in a microarray expression data set. This application is quite similar to speech recognition (Rabiner 1989) and sequence alignment applications (Karplus, Barrett et al. 1999; Wistrand and Sonnhammer 2005), and the model setup used by the algorithm developed in this dissertation is similar to profile HMMs, with one state representing one event in the sequence, and the length of the observation sequence being identical to the number of HMM states. The model developed in this chapter deviates from profile HMMs in the number of restrictions. It allows state transitions between all nodes if the training data strongly indicates an advantage in this. After the training step this HMM model tends to converge to a true profile HMM for the majority of nodes; exceptions generally have very low state transition probabilities.

Fig. 3 shows an example of a trained HMM for observation sequences of length n=4 produced by the training algorithm developed in this chapter. Analysis results presented throughout this chapter was performed with much longer sequences, for example using a series of n=60 features. The algorithm implements HMM functionality using the freely available Java HMM implementation JAHMM (Francois). Graph visualizations are generated using the freely available package GraphViz (Gansner and North 2000).

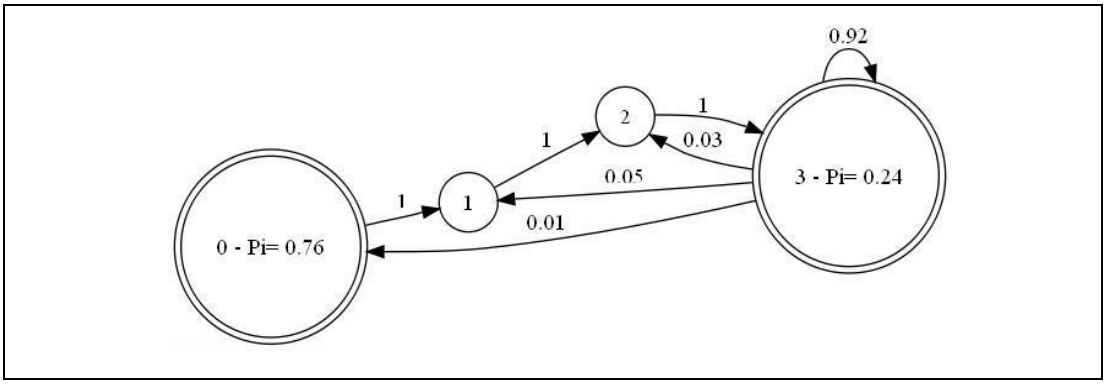

Fig. 3: An example of the structure of a trained HMM for an observation sequence of length T=4. Double circles indicate starting states with π_i>0. Transition probabilities, left to right: 1, right to left: very low. Emission probabilities, B, are omitted in this graph. All states emit symbols from the observation alphabet according to learned emission probabilities. Emission probabilities learned correspond to gene expression levels at each of the 4 experiments covered by this model.

Training an HMM is performed using the Baum-Welch algorithm (Baum, Petrie et al. 1970). This algorithm needs a fully specified initial HMM model, even if it is initialized using just random values. The algorithm initializes each HMM based on statistical observations in the input gene set. The number of nodes is set to the number of experiments selected after Functional Feature Reduction (FFR). Initial state probabilities are set to $\pi = \{\pi_1=0.9, \pi_2=0.1/(N\text{-}1),\ldots, \pi_N=0.1/(N\text{-}1)\}$, placing a heavy bias on the first node. State transition probabilities are initialized similarly, using $a_{ij} = 0.9$ for $j = i+1$ and $a_{ij} = 0.1/(N\text{-}1)$ for $j \neq i+1$, placing a heavy bias on always proceeding to the next state. Emission probabilities are initialized using an even distribution of all observations.

These parameter settings provide an early approximation of the structure of the final trained HMM, without locking any parameters firmly into place. If the training data demands changes to the basic profile HMM model then it can be accommodated.

Otherwise it converges to a basic profile HMM model. Fig. 3 demonstrates how state transition probabilities converged to 1 for all states a_{ij} where $j = i+1$, but also left a possibility to transition from state 4 to other states; and while initial state probabilities for π actually decreased for state 1, it increased for state 4. In general π_1 is expected to converge to 1 after training.

Given this model, the Baum-Welch algorithm modifies the parameters (A,B,π) to increase the likelihood $P(O|\lambda)$ that this HMM generates the provided observation sequences. There is no known algorithm to globally maximize the parameters, so Baum-Welch locally optimizes λ in one step and is repeated multiple times, until the HMM reaches sufficient probabilities and parameters stabilize. The Baum-Welch algorithm has a forward pass and a backward pass, which are like the E and M steps in EM (Expectation Maximization) algorithms and are guaranteed to converge to a local maximum (Dempster, Laird et al. 1977).

Let $Q = \{q_1, q_2, \ldots, q_N\}$ denote the states of the HMM and $I = \{I_1, I_2, \ldots, I_N\}$ the underlying state sequence; in a true Profile HMM I is expected to converge to $I = \{1, 2, \ldots, N\}$. Further, let λ^g denote parameter values from the previous EM iteration, and $O_{1:N}$ the entire observation sequence where the number of observations is identical to the number of nodes, N. Q can be viewed as the hidden variables and defined $Q(\lambda,\lambda^g)$ as the auxiliary functions to be maximized:

$$Q(\lambda, \lambda^g) = E[\log\, p(O_{1:N}, Q_{1:N} \mid \lambda) \mid O_{1:T}, \lambda^g] \qquad (5)$$

$$= \sum_{q_{1:N}} [\log p(o_{1:N}, q_{1:N} \mid \lambda)] \mid p(o_{1:T}, q_{1:N} \mid \lambda^g) \tag{6}$$

In each iteration of the Baum-Welch EM algorithm parameters λ are updated by maximizing Q with respect to λ:

$$\lambda^i = \arg\max_{\lambda} Q(\lambda, \lambda^i) \tag{7}$$

$$i = i + 1$$

This re-estimation procedure is repeated iteratively until convergence is reached and the change in parameter estimates λ is very small between successive EM iterations. The final and fully trained HMM model is then called the maximum likelihood estimate of that HMM.

3.1.4 Random HMM Generation and Test

The fully trained HMM is used to score every gene in the microarray, using only the Functional Features of the data set identified in the previous step. The score is the probability that an observation sequence was generated by the HMM. A high probability score indicates a high likelihood that the gene with the observation sequence was generated by the HMM. The goal is to find genes that score very high given the trained HMM.

To estimate the significance of obtained probability scores, a set of HMMs is generated and trained from a large number of randomly selected groups of genes from

the microarray, each group contains the same number of genes as the input gene set. Each gene in the microarray is then also scored using all random-generated HMMs. The resulting scores are used to estimate the Parzen density distribution function (Parzen 1962) for that gene. If the probability score from the input gene-trained HMM is significant with respect to the PDF of the random-generated HMM scores then the gene is included in the functional module set.

A Parzen density function (PDF) is a nonparametric kernel density estimation. The density function, $\hat{p}(x)$ is calculated as

$$\hat{p}(x) = \frac{1}{V}\sum_{i=1}^{n} K_i(x_i) \tag{8}$$

It is the sum of kernel functions $K_i(x)$ around x_i. The scaling factor V ensures that the total area under the function is 1. While K can be any kernel function, most commonly a Gaussian kernel centered on the data points is used, such that $K_i(x) = G(x;x_i,\Sigma)$. The general Gaussian kernel $G = G(x;\mu,\Sigma)$ is given in Eq. 9:

$$G(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{9}$$

Here d refers to the dimensionality of the data; μ is the center of the kernel, Σ and the covariance matrix. Parzen density functions have the advantage of not relying on a-priori assumptions on the distribution of HMM scores, but are derived directly from the data itself.

Significance estimates are calculated as the *p*-value of the original HMM score, given the scores of all random-set HMMs. The probability distribution for the random-set HMM scores is derived using Parzen density function. One-tailed probability thresholds for *p* are set very low to exclude "noisy" data, i.e. high-scoring genes that are not related to the input gene group.

3.1.5 Statistical Analysis

Functional annotations of the set of result genes are compared to annotations of the input gene set to estimate the confidence that new genes really are related to the input genes. This is not trivial; each gene has multiple different functional annotations in Gene Ontology (GO), because each gene participates in multiple cellular functions. Even if an input gene data set is chosen based on a specific functional similarity, other biological functions may also be overrepresented in the same data set. A machine learning algorithm given this input set may pick up new genes related to any of the original functions, not just the intended one. It is important to account for this in the final analysis of the result set.

GO term annotations list three categories for each gene: biological process, cellular component, and molecular function. The modules detected in this chapter are parts of the complex biological machinery of the cell, so of primary interest is whether genes are involved in related biological processes. However, Nothing in the algorithm itself limits its application to the other two categories.

For the result analysis in this chapter the list of GO term annotatons for new genes in the result set is first enlarged by including all direct parent and child terms in the GO term hierarchy. Then all annotations from gene homologues and orthologs listed in the FlyMine database (Lyne, Smith et al. 2007) are added, and annotations from interacting proteins from the BioGRID database (Stark, Breitkreutz et al. 2006) are also added. The similarity between input genes and the set of new genes is calculated as overlap between both GO term annotation lists, and by comparison of statistically overrepresented GO terms (given only the genes in the microarray) in both sets. The goal is to find high match rates in both categories.

Overrepresentation analysis is performed using the freely available Ontologizer software (Grossmann, Bauer et al. 2007; Bauer, Grossmann et al. 2008). This software package allows the calculation of overrepresented GO terms based on restricted input data sets, which allows for a calculation relative to the genes in the microarray data set used in each experiment. Calculations are carried out using the term-for-term mode and Westfall-Young single step calculation with 500 samples. Any result visualizations are performed using GraphViz.

If genes in the result set do not have any GO term annotations they are assigned a function based on the closest or most dominant category of the combined group, and are assigned to the cellular pathway of the input gene set. The higher the number of matches between the input gene set and the new genes in the result set, the higher the confidence of this assignment.

Due to the nature of this algorithm there remains a small amount of randomness in the results. Each algorithm run may produce slightly different sets of results; however, the overlap between result sets is very large. The difference stems from the use of randomized groups of genes used to estimate a PDF for each gene. Each run will produce a new set of random gene sets, affecting the PDF slightly.

3.2 Results

Tests for the HMM algorithm were performed on synthetic microarray data sets and on a biological data set. Two synthetic data sets were constructed: one by embedding a simulated pathway into a data set of randomized gene expression values. This pathway consists of gene expression values generated from an HMM previously trained on the cell cycle pathway in a biological data set. The second data set contains a pathway with aggregate gene expression values of the pathway genes above a certain threshold.

This first data set (HMM set) contains 100 experiments for 500 genes. The first 125 genes contain the simulated pathway, extending over 40 experiments. Gaussian noise, multiplied by a factor ranging from 0 to 4 was added to evaluate the ability of the algorithm to recover the correct set of Functional Features and pathway genes. The data set used is shown in Fig. 4b.

A second synthetic data set (SA set) was generated based on (Bergmann, Ihmels et al. 2003) to compare the performance of this algorithm to the Signature Algorithm. This data set, E^{cg}, was generated by setting gene expression values of all members of the pathway to 1, and all other values in the microarray to 0. The matrix E^{cg} was then multiplied by scaling factors s_g and s_c, $E^{cg}s_g s_c$, picked randomly from the uniform distribution over [0,1] for each gene and condition. Varying degrees of noise was then added to every element of the matrix. The data set generated for this study is shown in Fig. 4a.

The biological microarray used in this chapter is taken from GEO, a large repository hosted by the National Institutes of Health (NIH) (Edgar, Domrachev et al. 2002; Barrett, Troup et al. 2009). The data set accession is GDS 191, which is the Life Cycle of Drosophila (Arbeitman, Furlong et al. 2002). This microarray captures the life cycle of *D. Melanogaster* from conception through 30 days under normal environmental conditions. At each time interval during the experiment expression levels of genes from the entire organism were tested. This data set is first parsed to remove any data not related to the result (duplicate rows, tests, comparisons, etc), and columns and rows with an excessive amount of missing data are also removed (columns: ≥ 25% missing, rows: ≥ 33%), due to limitations in the ability to impute very large blocks of missing values. Any remaining missing values are imputed, resulting in a 2646 × 158 gene expression matrix with no missing values.

In sections 3.2.1 and 3.2.2 the algorithm is run with two biological input gene data sets: (1) the KEGG Cell Cycle reference pathway, and (2) the KEGG dme03012 pathway (Translation Factors).

3.2.1 Synthetic Data Set - Evaluation

Feature Reduction The algorithm uses the average absolute pairwise correlation coefficient of all possible pairs of input genes to select a subset of *n* Functional Features from the microarray data set. A synthetic microarray data set is used to evaluate the ability of this Functional Feature algorithm to select the correct features with respect to an increasing amount of noise.

Using the HMM Data Set the first 50 genes of the pathway were selected as input gene list. The purpose of this step in this algorithm is to select a good set of *n* features for the HMM training and evaluation algorithm.

The Functional Feature algorithm recovers 97.5% of the correct features in the data set at noise level 0 (no noise), 95% of features at noise level 1, 80% of features at noise level 3, and 60% at noise level 4 (pathway begins to blend into background noise). Even at noise level 4 the algorithm is still above purely random performance.

The advantage of the profile HMM algorithm is its ability to focus on features with higher amounts of information during the training process, and place less emphasis on features that provide less information.

Gene Selection Using the two synthetic data sets (HMM Set) and (SA Set), the performance of the HMM algorithm is evaluated in comparison to the Signature Algorithm and the effect of increasing noise on the ability to recover pathway genes. Fig. 4c and Fig. 4d show the respective results.

The HMM algorithm is tested on the HMM Data Set containing one pathway that includes 125 genes and extends over 40 conditions. This pathway was generated using an HMM previously trained on the biological data set and the cell cycle pathway. This pathway was embedded in a 500 × 100 microarray consisting of randomized expression values. To simulate noise, Gaussian noise multiplied by factors 0 – 4 is added to the entire microarray. This algorithm shows a good ability to recover pathway genes even in noisy data.

The Signature Algorithm does not work well with this synthetic microarray. By design, the Signature Algorithm detects groups of genes with higher-than-average aggregate expression values over the set of pathway genes. In the HMM Data Set the pathway genes are distinguished by high HMM scores instead; the expression values of the pathway genes are lower than many random genes.

To perform a valid comparison to the Signature Algorithm on synthetic data a 2600 × 100 microarray is generated using the same method described in (Bergmann, Ihmels et al. 2003). In this data set the pathway is characterized by genes with higher average expression values relative to the remaining genes. The pathway contains 250

genes and extends over 40 conditions. Gaussian noise is added to the entire microarray. This data set works with the strengths of the Signature Algorithm.

The analysis in Fig. 4d shows the HMM Algorithm to be competitive to the Signature Algorithm on this data set. This result is significant because it demonstrates the ability of the HMM-based machine learning algorithm to detect clusters in noisy data sets where the source of similarity is not primarily a correlation expression pattern between related genes, but shared extreme expression levels.

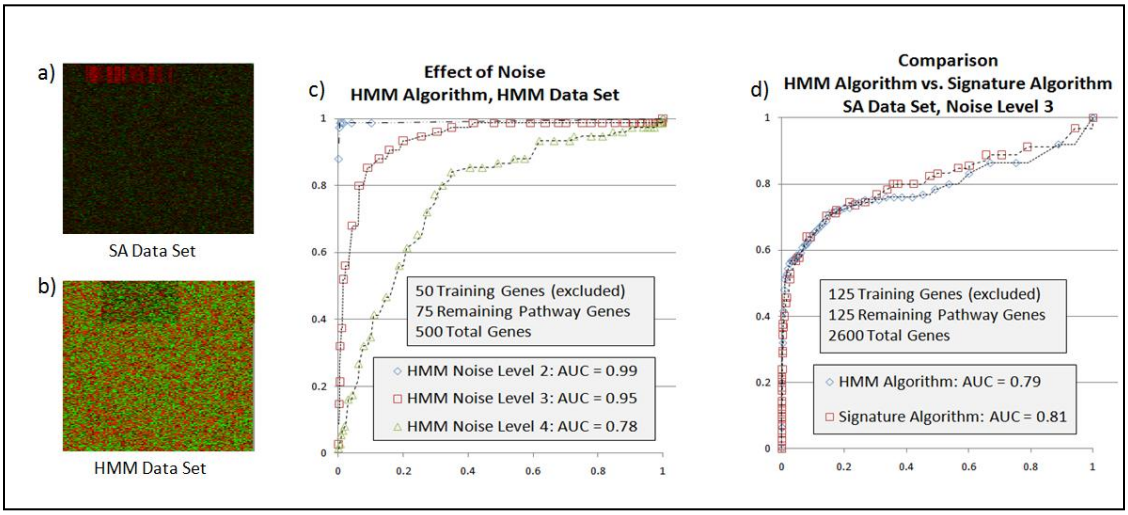

Figure 4: a) Synthetic Signature Algorithm (SA) Data Set, created according to (Bergmann, Ihmels et al. 2003) (see section 3.1 for details). The pathway contains 250 genes and extends over 40 experiments. The entire data set contains 2600 genes and 100 experiments. b) HMM Data Set is created by generating sequences of expression values from a previously trained HMM and embedding it in a matrix filled with random expression values. This pathway contains 125 genes and extends over 40 experiments. The entire data set contains 500 genes and 100 experiments. c) Shows ROC curves using the HMM Algorithm developed in this dissertation and the HMM Data Set. 50 genes were used to train the HMM, the ROC curve shows the performance finding the remaining 75 genes in data sets with increasing amounts of noise. Noise Level 2 contains some noise; Noise Level 4 leaves the pathway almost indistinguishable from background noise. d) Shows a comparison between the Signature Algorithm and the HMM Algorithm using the SA Data Set. In this test 125 randomly selected training genes were used to recover the remaining 125 pathway genes. The data set contains a medium level of noise (3). Both algorithms show comparable performance. The Signature Algorithm does not work well with the HMM Data Set because the expression values of the pathway are, on average, lower than the random expression values. The Signature Algorithm works by finding genes with aggregate expression values over all pathway genes above a threshold.

Run Time Evaluation This algorithm consists of three steps that tend to contribute differently to the total run time, depending on the size of the microarray data set, the size of the list of input genes, and the size of the Functional Feature set. Timing of these parts of the algorithm is tested using the HMM Data Set and a set of 50 input genes:

1. Functional Feature selection run time depends on number of features and the number of input genes. The run time from start of the program, including loading of all data sets, until the completion of the selection algorithm is 47 seconds with the synthetic HMM Data Set.

2. HMM training. This step is also dependent on the same factors – number of Functional Features and number of input genes. Additionally, this step trains multiple HMMs from random sets of input genes, which multiplies the time requirement by the number of random HMMs. This step is trivially parallel, however, which makes it convenient to use in a multi-CPU environment. Tests were performed using 400 random HMMs and were run using 4 Java threads on a quad-core Intel Q6600 CPU running at 3.0GHz. Run time for this part of the algorithm using the synthetic data set with 400 random HMMs is 7 minutes, 46 seconds.

3. Evaluation and Parzen Density Function (PDF). The run time of this step is dependent on the size of the microarray data set and the number of random HMMs used, because every gene in the microarray is scored with every HMM. This step is also trivially parallel and is performed using 4 Java threads. Run time for this step is 39 seconds, including disk output of the results.

The total run time using 50 input genes is 9 minutes, 12 seconds. Run times vary slightly between runs due to the random nature of this algorithm.

3.2.2 KEGG Cell Cycle Data Set (Derived)

There currently is no cell cycle pathway available for *D. Melanogaster* in KEGG, so the dataset used here is the set of all orthologs from the reference cell cycle pathway in KEGG, selecting only the genes present in the pre-processed GDS191 microarray data set. 18 genes matched and were taken as input genes for the algorithm. One gene with a low average pairwise correlation to the remaining set was removed, yielding an HMM training set of 17 genes.

800 additional HMMs were trained from random-generated groups of 17 genes to estimate the PDF used for each gene, and a low *p*-value of 0.005 is used as significance threshold for the inclusion of new genes, given the PDF. The algorithm produced 27 new genes with these settings, 23 of them with current GO term annotations.

Fig. 5a shows the similarity in expression patterns between the dominant features of the set of input genes and the new genes added. Functionally these new genes in the result data set are similar to the input gene data set.

Overrepresentation analysis shows that both the original set of genes and the combined result set have the same functional characteristic, which is a strong indication that the new genes added to this group are not random.

Analysis of the functional annotation shows that the majority of new genes with GO term (biological process) annotation are functionally related to the input gene data set. 83% of genes have annotations directly matching the set of expected annotations for the input genes. Four new genes have no GO Term annotation and based on these results genes with no current GO term annotations can be predicted to be annotated with the Cell Cycle biological process. These genes are: FBgn0033459, FBgn0033992, FBgn0030122, and FBgn0028506.

Of the remaining three genes with no direct match with the input gene annotations, two genes only have a single annotation. It is reasonable to add a cell cycle annotation to these genes as well: FBgn0030854 and FBgn0038306. This leaves just one gene with no matching annotation, FBgn0013269. Its pathway annotation is “Protein Folding.”

3.2.3 KEGG Translation Factors (dme03012) Data Set

The Translation Factor pathway genes taken from KEGG (KEGG Database) exhibit a low average pairwise correlation in the microarray data set used. Pre-processing the input data set to exclude genes with low average pairwise correlation produces only 10 matching genes between the microarray and the pathway. In these 10 genes there were two subgroups of 5 genes that exhibit a very high average pairwise correlation. This indicates that the pathway may contain more than just one function, and overrepresentation analysis confirms that each subgroup captures a different GO term annotation category from the original set of all pathway genes.

This is an interesting situation. To find genes associated with the Translation Factors pathway the algorithm was run twice – once for each of the subgroups. The results were combined again for the analysis step.

New genes are added in two independent runs of the algorithm, the result data set was combined for the final analysis. The number of random-group HMMs was lowered in this case to 400, and the inclusion *p*-value increased to 0.035 to account for the small size of the input set.

The two independent runs produce 6 and 19 new genes, respectively, for a total of 25 genes added to the original 10 genes. Fig. 5c shows the combined result set matrix. Of the 25 new genes there are 10 matches with expected GO terms from the input data set; one gene is annotated with a different pathway (‘Purine metabolism’) and one gene with a different GO term (‘protein amino acid glycosylation’), producing a

match rate of 88%. 8 genes have no prior GO term or pathway annotation and can be annotated with the translation factors pathway. The results are shown in Fig. 5b.

Putative annotations for the "translation factors" pathway are assigned to genes FBgn0001977, FBgn0042125, FBgn0037490, FBgn0035373, FBgn0036958, FBgn0034643, FBgn0036911, and FBgn0033794.

Overrepresentation analysis confirms that input genes and result data sets share common overrepresented GO terms.

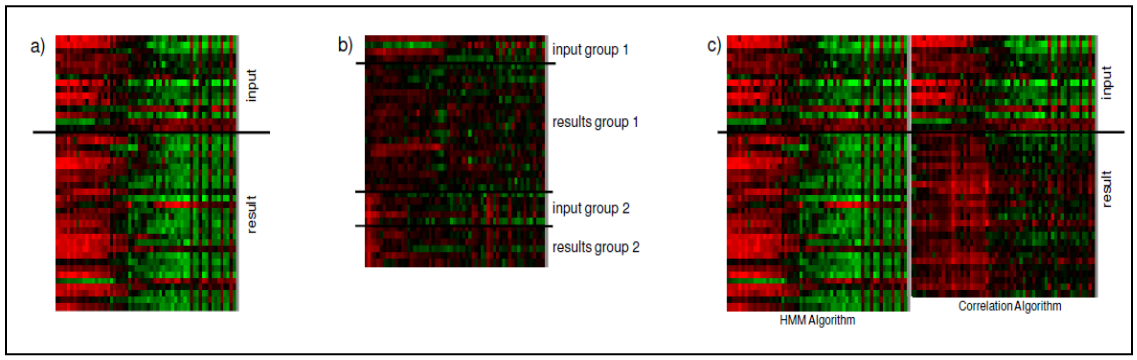

Fig. 5: a) Input and result genes for the Cell Cycle pathway. Result genes show a very similar expression behavior as the input genes. b) Two input gene groups, and result genes for Translation Factors pathway. In this case the pathway was split into two groups of genes, each producing their own result genes. Expression levels are generally low, but show similarities for each group. c) Comparison between HMM-based and correlation-based results for the Cell Cycle pathway. The result genes produced by the correlation-only algorithm exhibit visibly less consistent expression behavior to the input gene group. This is confirmed in the statistical analysis.

3.3 Discussion – Cell Cycle Data Comparisons

Why not just use correlation? The algorithm presented in this chapter uses machine learning tools to pick out genes whose expression profile is similar to a group of input

genes. It would be computationally much faster to simply pick out new genes based on correlation coefficients between new genes and the group of input genes.

The purely correlation-based algorithm was run on the same pre-processed Cell Cycle data set. The data set does have a very good pairwise correlation coefficient to begin with, so is it possible to pick out new genes based on correlation? Fig. 4c shows a comparison between HMM-based and correlation based runs.

It is possible to find genes that have a similar gene expression behavior in the microarray, although visually the HMM approach produces better agreement between the character of the input gene data and result genes. Functional analysis reveals that the resulting group of genes does not have the same functional characteristics in the overrepresentation analysis as the input genes. The new genes also do not have a significant number of expected GO term annotations.

The HMM-based approach is apparently more apt at recognizing the relevant expression values of the input genes. It is interesting to see how areas of greater variance in the expression values of input genes produce a more uniform expression level in the result genes, as seen in the right part of the result data set. Correlation by itself is limited in its ability to find relevant genes given a set of input genes.

Comparison to the Signature Algorithm The Signature Algorithm works with a similar overall architecture, but the details of the algorithm are implemented very differently. The gene expression matrix is normalized within each experimental

condition to zero mean and unit length for all expression values. Experimental conditions are selected where the sum over absolute expression values for the input genes are significant (above a threshold). The raw expression matrix is then normalized again with respect to each gene, weighing each condition according to its condition score, and genes with a significant absolute sum of expression values are added to the functional module. To reduce noise the algorithm is run multiple times with slightly modified input data sets: a fraction of the input set is replaced by random genes. Genes that occur in the majority of result sets are taken as final result of the algorithm run. This eliminates false positives based on potential noise in the input gene data set.

The Signature Algorithm has proven to run well with combined yeast microarray data sets (Ihmels, Friedlander et al. 2002). The SA is run with the same data sets used in the study of the fruit fly, the GDS191 microarray and the derived KEGG Cell Cycle pathway for *D. Melanogaster*. The set of input genes is the same set of 17 genes used with the algorithm.

While it was possible to generate results for individual data sets, the recurrence requirement did not turn out to have enough recurrent genes to be added to the result data set. In fact, the algorithm was run for 20 iterations with fixed parameter settings and a fraction of 25% of input genes replaced with randomly selected genes each time, producing widely varying result both in the number of experimental conditions as well as in the number of genes included in the functional modules.

This is an interesting result. Why does an algorithm that is so successful with the yeast data set produce no results? The Signature Algorithm works based on summation over ranges of genes and conditions. Analysis of the scores produced by the Signature Algorithm show that there is decent separation between experimental conditions of the input genes and the rest, enabling a selection of the Signature (subset of conditions where input genes score above a threshold).

The problem seems to be the scoring of all genes in the microarray data set. While it is possible to select genes above a certain threshold for any given set of input genes, it does not seem to be possible to generate result data sets with recurrent genes produced by slightly modified input data sets. The microarray used apparently does not contain features strong enough to be detected by the Signature Algorithm.

This suggests that the algorithm developed in this dissertation is able to pick out weaker signals from the data. The signal in this case is a certain gene expression profile that is related to some cellular function or pathway.

3.4 Conclusion

This chapter has introduced the linear Hidden Markov Model and demonstrated its performance next to a popular algorithm from the literature. This forms the baseline of the algorithm developed in this dissertation. From this stage the algorithm is enhanced to substantially improve its performance and widen the kinds of data sets

and input gene sets it can use. Chapter 4 adds a feedback cycle to the algorithm which enables it to use smaller input gene groups and to detect weaker modules. Chapter 5 adds the ability to query multiple microarray data sets at once.

Chapter 4 Iterative Hidden Markov Model

4.1 Multiple Iterations

Chapter 3 demonstrates the performance of the HMM algorithm to detect functional similarities based on similarities in the expression levels of genes in microarray data sets. One of the challenges with this approach is the quality set of input genes and the match between the functionality represented by the input genes and the microarray data set analyzed. Pre-processing the input set to divide it into correlation-based subgroups works; the results in chapter 3 have shown this on the Translation Factors data set. In this chapter a second solution to this problem is developed: iterating multiple times and using the results produced by one iteration as input set for the next iteration. This enables the algorithm to focus on genes adding to the distinct expression behavior of the majority of the input genes. This approach is also very helpful when only a very small set of input genes is available for initial HMM training.

The input to the algorithm again consists of a gene expression microarray data set and a set of seed genes with common cellular functionality. This chapter focuses on using sets from the published literature, which is similar in application to individual researchers investigating certain cellular functionality. Functional Feature Reduction proceeds in the same way as described in chapter 3 to reduce the number of

experiments. The machine learning algorithm selects a new group of genes from the entire microarray to resemble the functionality of the seed genes.

4.1.1. Multiple Data Set Meta Analysis

It is also possible to improve the performance of the algorithm by integrating the results obtained from multiple data sets to produce one improved final result set. The increasing number of available microarray data sets enables researchers to analyze various related microarrays with the same input genes. Genes that show close similarities in the result sets of multiple microarrays can be selected with a higher level of confidence compared to single microarray results. Conversely, the variety of microarrays allows the algorithm to pick up new functionally related genes that may have been missed by analysis of other data sets. In this analysis, if multiple similar data sets are used then a final result set of genes is selected based on a majority-voting scheme from all individual result sets.

The focus of this chapter is on the use of closely related microarray data sets. Using microarrays with widely different background and experimental goals is expected to also produce widely different sets of result genes. In this case the Meta Analysis approach can also be expected to not produce very good results. An approach to gainfully deal with widely differing microarrays is developed in chapter 5.

4.1.2 Feature and Gene Selection

Functional Feature Reduction in this chapter follows the same algorithm already presented in chapter 3: a fixed subset of *n* experiments is selected from the microarray by maximizing the average pairwise correlation coefficient of the seed genes. This eliminates experiments where the seed genes do not cooperate (i.e. noise) from the data and improves the performance of the algorithm. Genes are represented by the observed expression values in the microarray data set.

The HMM score of a gene is the probability that a series of observed intervals from the microarray could have been produced by the trained HMM. All genes in a microarray are scored and the list is sorted according to HMM score. This is where chapter 3 ends. In this chapter the top-scoring genes are either fed back to the training algorithm for the next iHMM iteration, or they are used as final output list of the algorithm. The cutoff for the list of genes included in the result is based on the rate of decline between successive gene scores in a sorted list of all gene scores. More detail on this step is provided in section 4.2.3.

4.1.2 Functional Learning using iHMMs

Hidden Markov Models The same theory regarding HMM training applies. Parameters are learned from sequences of observed events, referred to as observation sequences. Fig. 6 shows the application to microarray data. The model setup used in

this algorithm is very similar to the BLOCKS topology (Henikoff, Henikoff et al. 1995) with one state in the HMM representing one observation event in the sequence, and the length of the observation sequence being identical to the number of HMM states ($N = T$). The Functional Features from the previous step represent "similarity motifs" for a particular functional module in this model. The model developed in this dissertation deviates from the approach in (Krogh, Brown et al. 1994; Henikoff, Henikoff et al. 1995) in allowing state transitions between all nodes, if the training data strongly indicates an advantage in this. This better enables the algorithm to model noise in the data set. Fig. 6 shows the relation between the HMM topology and the gene expression data.

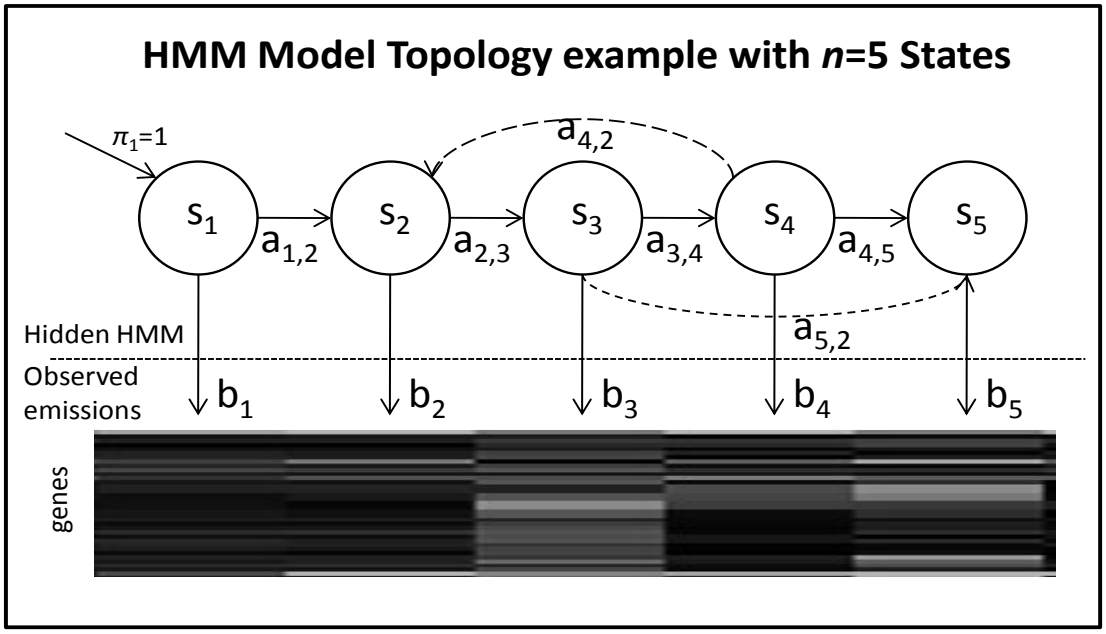

Fig. 6: A HMM with n=5 states models functionally related genes. There are as many HMM states as there are microarray experiments and state transitions $a_{i,i+1}$ tend towards 1 after the training. Other transitions are permitted, if the training data indicates an advantage. This HMM learns the "profile" of expression values of a group of seed genes. In this paper n tends to be much larger than 5.

Algorithm Iteration The trained HMM is used to score all genes in the microarray. The genes are then rank-ordered by the logarithm of the HMM probability score, as shown in Fig. 7. The graph of scores invariably exhibits tails at each end where HMM scores decrease more rapidly compared to the center of the graph. The genes scoring in the left tail are then used as a seed gene set to a new algorithm run.

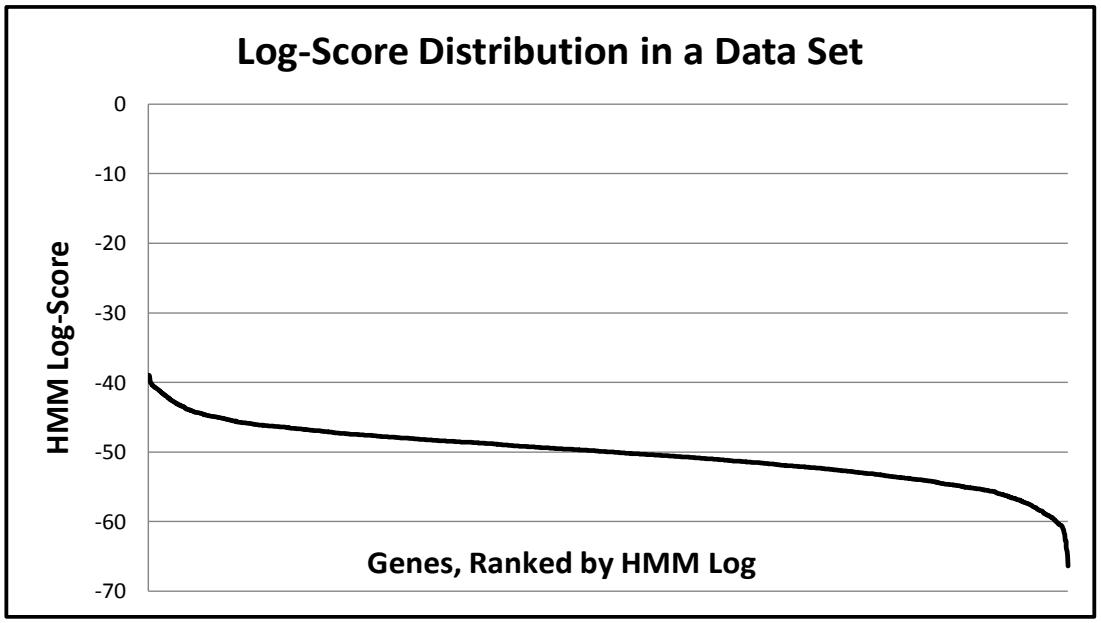

Fig. 7: Scores of all genes in a microarray sorted by HMM score. The tails at each end of the graph show rapidly decreasing HMM scores. The shape of this curve is used to select genes to be included in the next iteration of the algorithm.

Each algorithmic iteration changes the composition of the training data. The underlying assumption for this approach is that a functional module contains more genes than there are seed genes, so that the inclusion of additional genes provides more information, resulting in a better characterization of the expression profile by

the HMM. Genes to be included in subsequent runs are selected from the genes responsible for the left tail in Fig. 7, specifically from genes before the score curve flattens out, retaining only the highest-scoring genes. Subsequent algorithm runs train the HMM with the improved information provided by the new seed genes. Results provided by (Bergmann, Ihmels et al. 2003) and others indicate the potential of iterative refinement of results.

Two approaches are used for determining which genes are to be included in the results set after each iteration. The first approach picks all top-*n* highest-scoring genes, where *n* is a parameter selected relative to the size of the input gene set of the previous iteration. The second approach takes the shape of the curve in Fig. 7 into account, and selects genes up to a point where the slope between consecutive scores flattens out too much. Both approaches have shown to work; the results presented in this chapter predominantly use the top-*n* scoring gene-approach.

Meta Analysis If multiple comparable data sets are available then this iHMM algorithm is performed independently on each data set and the final result gene list is obtained by simple majority vote of all individual result gene lists. Section 3.1 demonstrates that this approach has the potential to significantly improve performance, especially in very noisy data sets. This approach does not perform well if the individual results are obtained from very different data sets.

4.1.3 Statistical Analysis

In synthetic data sets the list of true positives is known a-priori, rendering the result analysis trivial. In biological data sets functional annotations of the set of result genes are compared to annotations of the seed gene set to estimate the confidence that new genes really are related to the seed genes. This is not trivial, as described in section 3.1.5: even if a data set is chosen based on a specific functional similarity, other biological functions may also be overrepresented (enriched) in the same data set. A machine learning algorithm given this input set may pick up new genes related to any of the original functions, not just the intended one. It is important to account for this in the final analysis of the result set.

In the analysis the biological function of the seed gene set are characterized by the list of statistically overrepresented (enriched) GO term annotations of the set. The modules detected by this algorithm are parts of the complex biological machinery of the cell, so of primary interest is whether genes are involved in related biological processes. The same overrepresentation analysis is performed on the result set of genes. If both sets of genes exhibit the same overrepresented set of (biological process) GO terms then the result is deemed successful and the confidence that any additional genes added to the list of seed genes performs the same cellular function is very high.

If genes in the result set do not have any GO term annotations they are assigned a function based on the closest or most dominant category of the combined group, and

are assigned to the cellular pathway of the seed gene set. The higher the number of matches between the seed gene set and the new genes in the result set, the higher the confidence of this assignment.

4.2 Results

Tests are performed on three synthetic microarray data sets and validated biological application of this algorithm on a well-known Yeast Cell Cycle data set (Spellman, Sherlock et al. 1998) as well as a widely used *D. Melanogaster* data set (Arbeitman, Furlong et al. 2002).

One synthetic data set is constructed by embedding a simulated pathway generated from an existing trained HMM into a data set of randomized gene expression values. The embedded functional module consists of gene expression values generated from an HMM previously trained on the cell cycle pathway in a biological data set. This process was repeated ten times, generating ten different pathways from the same trained HMM. Gaussian noise was then added in 10 levels to the data set, producing a total of 100 synthetic data sets. Each matrix contains 80 experiments for 2,500 genes. The simulated module extends over a subset of 220 genes and 40 experiments. This data set allows us to study the effect of multiple HMM iterations and the Meta analysis approach with respect to increasing noise levels.

A second synthetic data set is taken from a recent bi-clustering benchmark paper (Prelic, Bleuler et al. 2006). This is a widely used data set to compare the general clustering performance of algorithms. This relatively small data set contains 10 non-overlapping modules. Each module contains 10 genes and extends over 5 experiments; the total microarray contains 100 genes and 50 conditions. Modules contain genes with the same random expression level inscribed in an array of random numbers. Gaussian noise in increasing levels is added uniformly to all genes and experiments. This data set is also used in other publications (e.g. (Dhollander, Sheng et al. 2007)) and provides a good baseline to compare the iHMM algorithm to existing clustering algorithms.

The third synthetic data set is quite similar to the set used by (Prelic, Bleuler et al. 2006) but is much larger in size. This data set contains 10 non-overlapping functional modules extending over 2000 genes and 100 conditions. The entire data set contains 20,000 genes and 1000 conditions. This allows us to evaluate the performance of this approach when facing very large data sets, such as is expected for biological whole-genome microarray data sets.

The biological microarray used in this chapter was originally used by (Spellman, Sherlock et al. 1998) and was taken from (Dhollander, Sheng et al. 2007) supplementary material site, where it was used to analyze the QDB algorithm. Routinely it was first parsed the set to remove any extraneous data (duplicate rows, tests, comparisons, etc); columns and rows with an excessive amount of missing data are also removed (columns: $\geq$ 25% missing, rows: $\geq$ 33%). Any remaining missing

values are imputed using a *KNN* imputation scheme (Troyanskaya, Cantor et al. 2001; Janssen, Donders et al. 2010), resulting in a 5498×74 gene expression matrix with no missing values.

The Drosophila data set consists of a 8623×66 gene expression matrix derived from the Arbeitman et al. Life Cycle of Drosophila data set (Arbeitman, Furlong et al. 2002). This data set is pre-processed in the same way as the Yeast data set.

This algorithm is intended to draw upon prior knowledge in the form of lists of genes known to participate in the same cellular function. A list of seed genes is used to train the machine learning part of the algorithm to produce a result list of genes participating in the same function as the input set, as well as a list of experiments where the gene expression values are most closely related. This work is compared with three approaches proposed for the same task: the Iterated Signature Algorithm (Ihmels, Bergmann et al. 2004); the Gene Recommender algorithm (Owen, Stuart et al. 2003); and the Query-driven Biclustering algorithm (Dhollander, Sheng et al. 2007). Each algorithm uses a list of genes to train a machine learning algorithm in order to produce a list of related genes as output; the iHMM algorithm performs very well on a variety of input data sets. Section 4.2.2 provides comparative results obtained with publicly available implementations of GR and QDB.

4.2.1 HMM-Generated Synthetic Data Set

The effect of multiple iterations Chapter 3 presented a Hidden Markov Model-based algorithm to detect functionally related genes based on similarities to in the gene expression signature compared to a given set of seed genes. This algorithm worked very well in the HMM-generated data set, especially compared to the Signature Algorithm (Ihmels, Friedlander et al. 2002).

With the iHMM algorithm feedback has been added to the training section of the algorithm. Once an HMM is trained and all genes in the data set have been scored, the top-ranking genes are selected as seed gene set for the next algorithm run. Fig. 7 showed how the sorted gene log-scores show a tail at the high and the low end. Genes in the high-scoring tail are selected as new seed genes. The tail is detected by comparing three consecutive slopes between two scores; when all three slopes have reached the average of the slopes in the center of the curve then the selection algorithm ends.

This enables the algorithm to better learn the predominant features of the seed genes in the data set. If the tail of scores produces fewer seed genes than the original set of seed genes there is a risk of overtraining and HMM parameters are relaxed by reducing the number of discrete HMM symbols. The second constraint in the section is to limit the number of new seed genes. The size of the seed gene set cannot exceed a certain percentage of the original seed gene size. This percentage is chosen as parameter. In this chapter it is set to 50%. The effect of multiple iterations of the

algorithm can be seen in Fig. 8. Here the ROC curves of four consecutive iterations are shown for a data set at noise level 7:

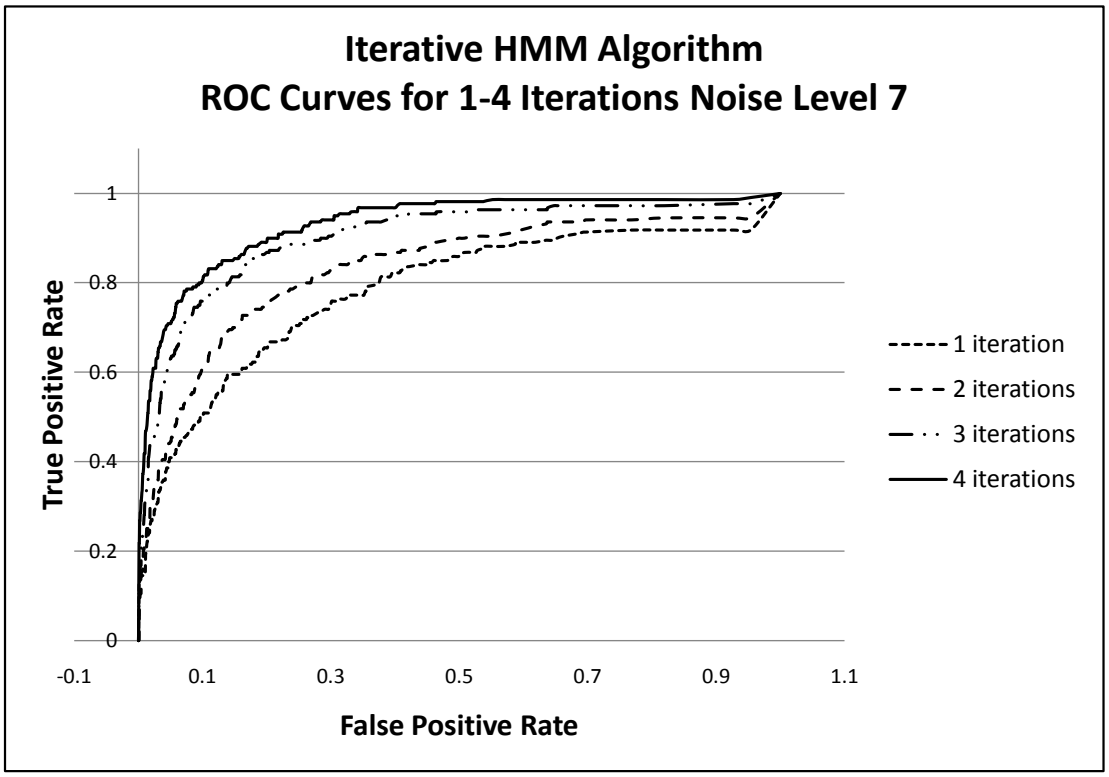

Fig. 8: ROC curves for the same set of seed genes and the same microarray data set at a high noise level 7. Each consecutive iteration uses the top-scoring results from the previous algorithm run as seed gene list. Each iteration increases the detection performance of the algorithm.

The ROC curve for 1 iteration in Fig. 8 is comparable to the results presented for the algorithm in (Senf and Chen 2009) presented in chapter 3. It can be seen how each iteration increases the ability of the algorithm to recover the pathway in this very noisy data set.

The effect of integrating multiple data sets The original algorithm is improved further by looking towards the availability of multiple datasets. The tremendous increase in available microarray data makes it now possible to find multiple data sets covering a pathway or a gene set of interest. In the simulated data scenario multiple different data sets are used that were generated independently using the same HMM model producing distinct observation sequences for each data set.

Fig. 9 shows the ROC curves of five individual data set runs using a data set with very high level noise 8, plus the ROC curve after integrating the results of the five runs using a simple majority-vote Meta analysis scheme:

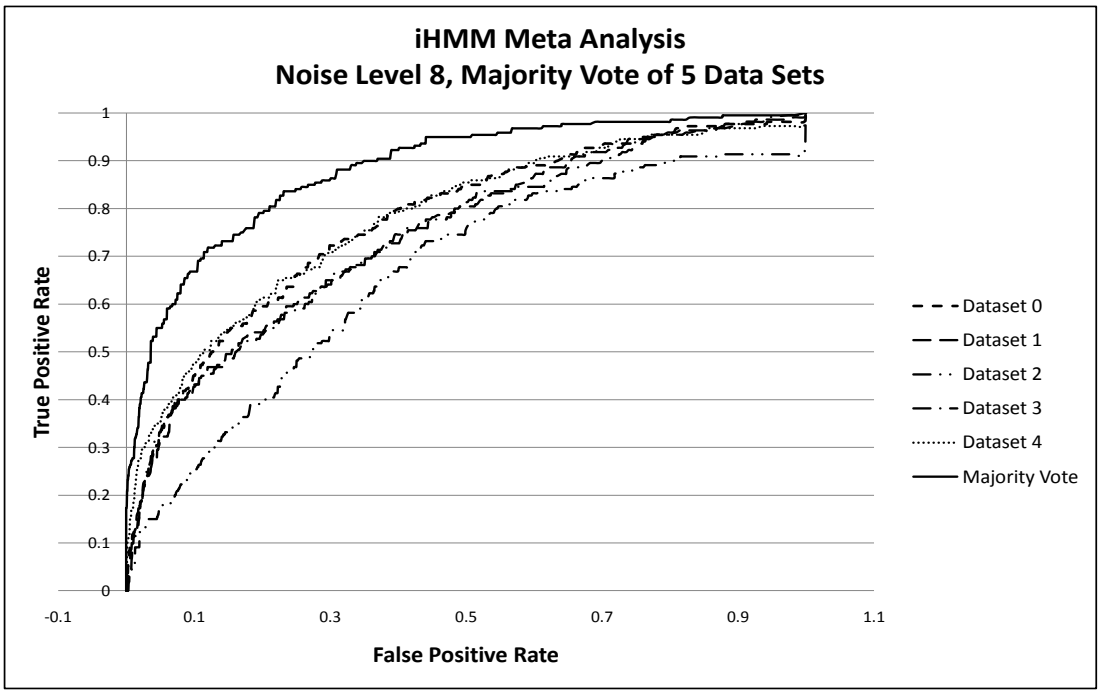

Fig. 9: ROC curves for the same set of seed genes and five different microarray data set, all at a very high noise level 8. Due to the random nature of the added noise, each ROC curve is different even though the same set of seed genes is used. The solid line shows the ROC curve of genes selected if they were in a simple majority of result sets (3, in this case). This produces a dramatic increase in detection performance and rivals the algorithm performance with substantially less noise.

The result shows that the iHMM results are very receptive to the Meta analysis approach if the data sets are reasonably similar. This approach allows us to cancel out some of the effects of noise that was added independently to each data set.

Comparison to Gene Recommender In application this algorithm is closely related to the Gene Recommender algorithm (Owen, Stuart et al. 2003), which also produces a ranked list of genes based on a set of seed genes. The Gene Recommender algorithm is freely available as R package (Gentleman, Carey et al. 2004; Dessau and Pipper 2008). To compare this algorithm with this algorithm the same seed genes are used on the same data sets as used for the iHMM algorithm, using noise levels from 0 to 9.

The ROC curves for Gene Recommender in Fig. 10 below clearly show that the algorithm employed by GR does not work well in data sets where pathways are characterized by similarity of expression signatures rather than by extreme (high or low) expression values. These results resemble the results obtained using the Signature Algorithm in (Senf and Chen 2009) and in chapter 3; these results are essentially random. Multiple iterations and Meta analysis did not improve these results, which is expected given the unsatisfactory performance even a very low noise levels with this data set.

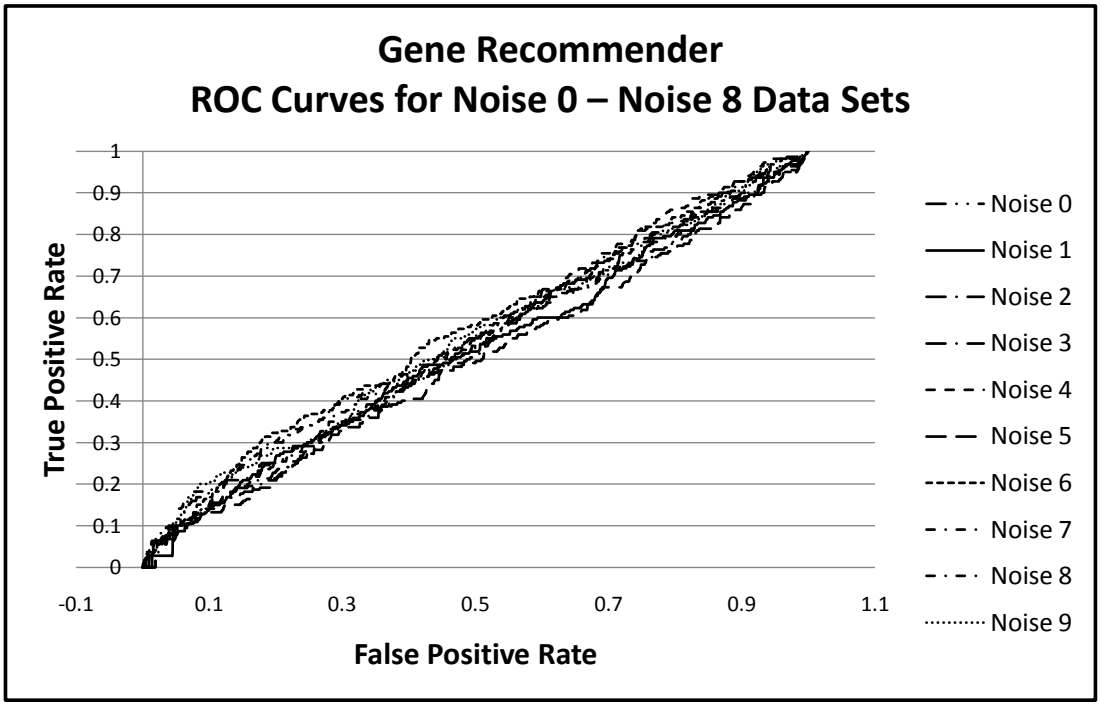

Fig. 10: ROC curves for the same set of seed genes used in Fig. 8 and Fig. 9 for the Gene Recommender Algorithm. Even at minimum noise levels GR cannot detect the correct genes; the ROC curve exhibits random performance at all noise levels. The reason is GR is clustering genes with extreme expression levels; but in this data set modules are characterized by expression-level similarities, not necessarily very high or low expression levels.

4.2.2 Multiple-Module Synthetic Data Set

The algorithm most similar in its approach to use the similarity of the pathway gene expression signatures is the Bayesian Query-driven Biclustering (QDB) algorithm (Dhollander, Sheng et al. 2007). QDB uses a set of seed genes to initialize the model parameters of a Bayesian model, which is then iteratively refined. The result of this algorithm is a set of modules characterized by genes and experimental conditions. This algorithm is also freely available as R package (Team). Because it does not generate individual gene scores, QDB is compared to iHMM using a combined

cluster match/significance score, which is the same metric used in (Dhollander, Sheng et al. 2007). A single gene is randomly selected from each of the ten inscribed modules. The percentage of correctly identified genes is averaged over all 10 modules to produce one match score for the entire data set. This procedure is repeated at increasing levels of noise to produce the match score curves shown in Fig. 11:

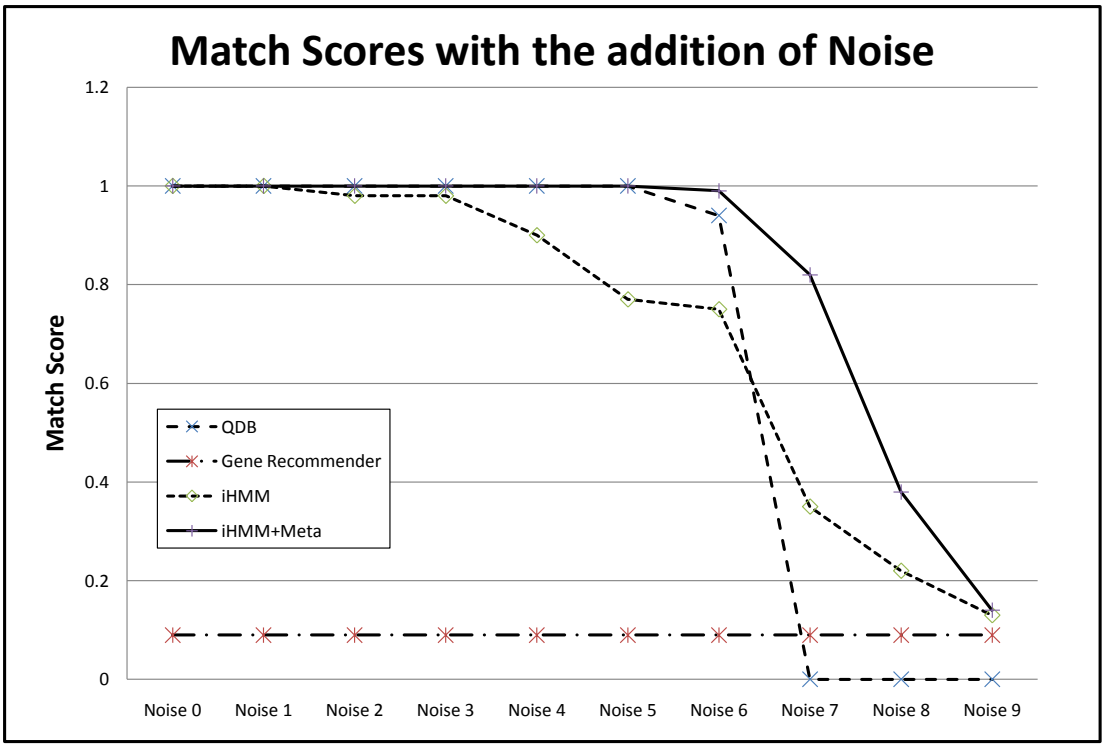

Fig. 11: Match scores for noise levels 0-9 for iHMM, GR, and QDB. A score of 1 indicates a match of 100% of all modules. Each score represents the average match score from individual algorithm runs for all 10 modules using one gene from each module as seed gene.

It is interesting to note that QDB produced very high-quality results until the data set gets too noisy, at which point no clusters are detected at all; a steep drop in match score performance. For very noisy data sets the iHMM algorithm seems to be better suited. Combined with the Meta analysis scheme the iHMM algorithm is able to out-

perform the QDB algorithm at all noise levels. This result is significant in light of the fact that HMMs are not meant to be trained with just a single gene as input, yet this algorithm still performs competitive to other published algorithms.

4.2.3 Large Clustering Benchmark Set

The purpose of any clustering or gene detection algorithm is to be used in real world data sets. With recent advances in microarray technology more and more whole-genome microarray sets are generated. These data sets can become very large; for this reason a large synthetic data set is included with the results. The data set used in this comparison is essentially similar to a widely used benchmark set originally from used by (Prelic, Bleuler et al. 2006) for a comparison of current bi-clustering algorithms. The primary difference of this set is its size: 20,000 genes and 1,000 conditions. This data set was used in (Li, Ma et al. 2009) and is available on the supplementary data for this chapter.

At this size the performance of the feature selection algorithm becomes increasingly important. While it is easily possible to use all 50 conditions for HMM training in the previous data set (the iHMM typically gains by including more conditions), it is not feasible to train HMMs with 1,000 conditions. Fig. 12 shows the layout of the synthetic microarray data set with the inscribed gene modules, and the average match score over the entire data set:

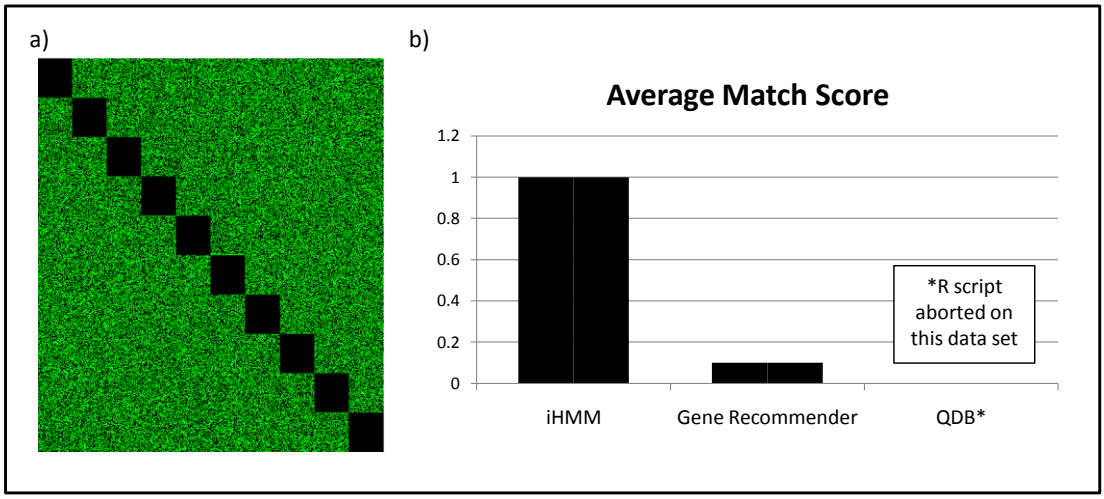

Fig. 12: (a) Layout of the very large data set with 20,000 genes and 1,000 experiments and the 10 inscribed modules. (b) The iHMM algorithm detects all modules while Gene Recommender produces random results, for the same reason as seen in Fig. 10. The provided R script for the QDB algorithm aborted on this data set with a memory error. This algorithm needs to be re-implemented before it can be used on large data sets.

The low performance of the Gene Recommender algorithm is not surprising given the performance on the previous (Prelic, Bleuler et al. 2006) data set. Gene Recommender is not intended for data sets containing modules where the similarity of expression patterns extends over a relatively short fraction of the number of experiments.

4.2.4 Biological Sets

Spellman Yeast Cell Cycle Data Set The goal of any clustering or gene finding algorithm is to be used on biological microarray data sets and to produce results with significant biological meaning and application. Yeast (*S. Cerevisiae*) is a model

organism in molecular biology, which makes it a good test case for proposed algorithms in the literature. The well-known Spellman Yeast Cell Cycle data set (Spellman, Sherlock et al. 1998) is selected to demonstrate the utility of this algorithm with biological data sets. The data set was parsed to remove any extraneous data (duplicate rows, tests, etc); columns and rows with excessive missing data were also removed (columns: ≥ 25% missing, rows: ≥ 33%). Any remaining missing values were imputed using a *KNN* imputation scheme (Troyanskaya, Cantor et al. 2001; Janssen, Donders et al. 2010). After pre-processing this data sets contains 74 experiments for 5498 genes.

A biological application might start with a set of genes that has been determined to cooperate functionally; it is then of interest if there are further genes cooperating with the group. This information could be used to determine target genes for new experiments, or to better understand a biological pathway in the cell. A set of seven cell cycle-related genes was taken from (Spirin and Mirny 2003) as seed genes for this algorithm. Overrepresentation analysis on these genes confirms that the dominant functions are the regulation of cell cycle and phosphorous metabolic processes. The seed gene group consists of YGR108W, YPL124W, YBR135W, YPR119W, YBR160W, YLR210W, and YPL256C (YDL155W is a seed gene, but is not in the microarray data set).

The resulting gene group after this algorithm consists of 79 genes, 71 of which have current GO term annotations. Fig. 8 shows the overrepresentation analysis graph

of the set of 8 seed genes obtained from (Spirin and Mirny 2003) and the same GO terms of the resulting set containing 71 annotated genes.

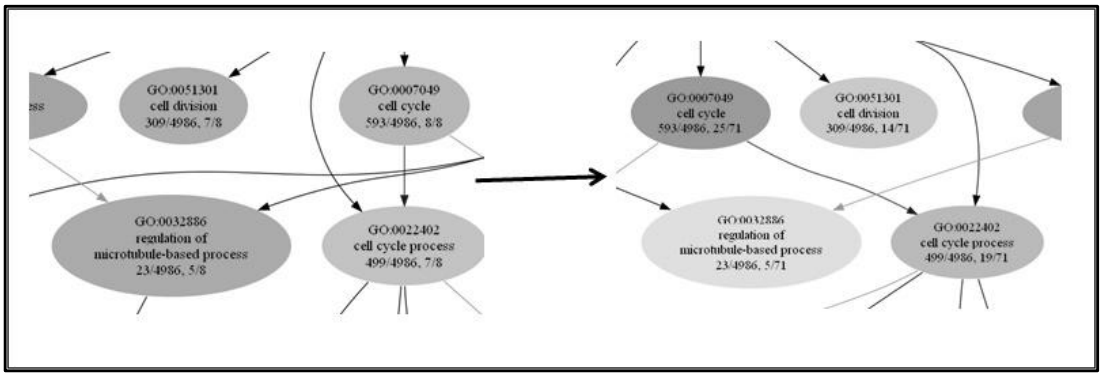

Fig. 13: An excerpt of the GO term overrepresentation (enrichment) analysis graphs of the group of seed genes on the left and the result group on the right. The full graphs are available in the supplementary material (Supplementary Fig. 1 & 2). This graph demonstrates that new genes added by the algorithm do not alter the functional characteristic of the gene group, although a large group of genes was added.

The close overlap of overrepresented terms clearly indicates that the group of genes found by the iHMM algorithm successfully enlarges the functional module provided in the seed genes by a substantial number of 73 additional genes without changing the predominant function of the module.

The 8 genes without GO term annotation are YBR206W, YCLX01W (withdrawn ORF), YJL018W (computational GO annotation: cell cycle, microtubule organization), YJL188C, YJR157W, YLR062C, YOR102W, and YOR325W. Given the result set overrepresentation analysis these putative genes can reasonably be inferred to have common functionality with a cell-cycle or closely related process. The one verified ORF (not marked as dubious ORF) has been independently

annotated in SGD with the cell cycle/microtubule organization processes, in line with results produced by this algorithm (Cherry, Ball et al. 1997).

The same analysis is performed using the GR and QDB algorithms to get a better understanding of the comparative performance on a real biological data set. GR produces a ranked list of genes, so in this case overrepresentation analysis of result groups containing the top-25, top-50 and top-75 genes is performed. Each group shows an overrepresentation of the GO cell cycle process with several genes added to the original seven genes, but the functional alignment with the seed genes is much weaker compared to the iHMM results. The best alignment is produced by the top-25 result set, although at a much weaker level compared to iHMM. All top-25 results genes have current GO term annotations.

QDB was run using the same microarray and seed gene set, but it only produced clusters of size 2,814 and larger - too large to represent an actual functional module, indicating that the biological cell cycle module is weakly represented in the microarray, does not correspond to the definition of a module used in QDB.

Life Cycle of Drosophila Fruit Fly Data Set The result on the Yeast data set indicated that the seed genes used did not provide sufficient information for the QDB algorithm to detect biologically meaningful clusters. To demonstrate the utility of this approach with a variety of biological data further experiments using a data set from a different organism is performed.

A widely used data set is the Life Cycle of Drosophila set, which contains gene expression data for the fruit fly from conception through its life span of 30 days. After pre-processing this data sets in the same way as the Yeast data it contained 66 experiments for 8623 genes.

Initially a group of six seed genes is selected with a common GO term annotation of GO:0050770 (regulation of axonogenesis): FBgn0014020, FBgn0014011, FBgn0000606, FBgn0026375, FBgn0010333, and FBgn0023172. Initial enrichment analysis shows the seed gene group to have a predominant functions of axonogenesis, (regulation of) cell development (GO:0048468, GO:0050793), also multicellular organismal process (GO:0032501), and others; the machine learning algorithm may pick up on any of the presented functions in the seed gene set.

From the results the top-50 scoring genes are selected for enrichment analysis. Predominant among the genes picked up in the result set are cell development, cellular developmental process, and multicellular organismal process, indicating that the expression gene similarity pattern in the data set is caused in large part to functions relating to multicellular development and organization.

This algorithm found a significant number of genes (33) also participating in the same functions; it further picked up a GO Molecular Function of binding (GO:0005488) which was not enriched in the seed gene set, further evidence of the multicellular functional character of the result set.

Gene Recommender and QDB are also run using the same data set and seed genes to compare the performance of the iHMM algorithm in this setting. The results are comparable to what was found in the Yeast data set: the clusters defined by the seed genes are too vague for QDB to produce small and functionally enriched clusters. Instead the algorithm produces clusters containing over 2,000 genes. Gene Recommender similarly does not produce top-*n* sets genes enriched with gene functions. These results are repeated over various biological data sets.

4.3 Conclusion on Multiple Iterations

An iterative Hidden Markov Model (iHMM) algorithm is developed to learn the expected expression levels across microarray gene expression data sets for groups of genes involved in the same cellular function. The trained iHMM then scores all genes in the microarray to detect additional genes participating in the same function. The algorithm may iterate using the result group as input to a new algorithm run. Runs from multiple comparable microarray sets covering the same set of genes may then be combined using majority vote Meta analysis.

The algorithm is analyzed extensively on a variety of synthetic data sets against related algorithms in the literature (Gene Recommender, Query-driven Biclustering) and validated the application with a well-known biological data set. The algorithm has shown to run very well with the synthetic data sets, consistently outperforming

the Gene Recommender and perform above the level of QDB especially in data sets with very high levels of noise.

Running the algorithm with the Yeast Cell Cycle and Drosophila data sets demonstrates the applicability for real biological data analysis tasks, and the successful run with a very large synthetic data set demonstrate readiness for full-genome microarray data sets of all species.

It became evident that all machine learning algorithms are dependent on the quality of the training data. Enrichment analysis of biological seed gene sets reveal in most cases that multiple genetic functions are enriched; which function is predominantly learned by an algorithm depends on the experiments used in the generation of the microarray data set.

A first look at integrating multiple data sets with a Meta analysis approach combining the results of various individual algorithm runs using majority vote to add genes to the result set is shown. This has shown to be very successful in the synthetic data.

The availability of multiple data sets covering the same genes is increasing rapidly and poses an opportunity to combine these sets to improve algorithm performance. At this point the focus is on developing more advanced techniques to integrate multiple data sets during the analysis itself.

Chapter 5 Combined Data Set Analysis

5.1 Heterogeneous Microarray Combination

Chapter 4 saw the introduction of a Meta Analysis scheme to combine the results from multiple microarray data sets. Synthetic data set results have shown a marked increase in the performance of the detection algorithm, especially in cases where data sets contained high levels of noise. However, even as the number of microarray gene expression data sets is increasing rapidly, not all microarray data sets are similar enough to gainfully use the Meta Analysis approach. Yet much information can be gained by analyzing data from multiple data sources even if they are very different. In this chapter an algorithm is developed to dynamically integrate multiple microarray data sets based on a set of query genes to be analyzed with a Hidden Markov Model. The performance of the algorithm is tested on Drosophila time-series data sets and show its capability to detect new and biologically meaningful genes for the ATP synthesis metabolic process.

5.1.1 Background

In Chapter 3 a Hidden Markov Model based machine learning algorithm was developed for the analysis of individual microarray data. Chapter 4 saw an iterative feedback loop and the benefits of integrating similar microarray data sets. It would also be beneficial to integrate data sets that cover the same genes in an organism, but

are very different from an experimental point of view; more experiments allow for a better characterization of the gene expression behavior of functionally related genes.

There are two basic approaches when it comes to integrating multiple microarray data sets: (1) Meta-Analysis, which is concerned with the integration of the results from individual data sets (Grutzmann, Boriss et al. 2005; Hong and Breitling 2008; Wren 2009; Yang, Zou et al. 2009), and (2) statistical data integration, which attempts to combine multiple data sets into a single data set before analysis (Choi, Yu et al. 2003; Ng, Tan et al. 2003). This category also includes efforts to normalize data sets (Pieler, Sanchez-Cabo et al. 2004).

As chapter 4 introduced, the primary challenge with these efforts is the fact that every microarray data set is generated under a unique set of environmental and experimental conditions; every variable has an impact on the behavior of the genome so that data sets cannot simply be concatenated. Meta-analysis sidesteps this issue to a degree by performing statistical analysis for each data set separately and then integrating the results. Data set combination and integration prior to analysis primarily attempts to adaptively normalize each data set to render the gene expression levels comparable between data sets. The algorithm developed in this chapter performs both types of analysis on a set of synthetic and biological time-series microarrays to determine advantages and disadvantages of each approach.

The analysis task itself attempts to find groups of related genes, which is typically formulated in terms of a clustering problem (Eisen, Spellman et al. 1998; Tavazoie,

Hughes et al. 1999). While unsupervised clustering is very useful, a biologically more interesting approach starts with a set of genes already known to be functionally related and explores the microarray for additional genes participating in the same function as well as the specific conditions where this function is active in the cell. This allows for directed "queries" of the data sets to study specific functions or pathways.

5.1.2 Methods and Data Sets

The input to the algorithm developed in this dissertation continues to consist of a set of genes known to be functionally related and a set of microarray data sets. A synthetic data set was generated from the same data set used in (Senf and Chen 2009) and in chapter 3 by breaking it into multiple partially overlapping smaller microarray data sets. For biological data sets 26 publicly available time-series microarray data sets from Drosophila were taken. Analysis is performed individually on each microarray data set as well as on an integrated data set, dynamically comprised of all microarray sets based on a set of query genes.

Each microarray data set is preprocessed by removing experiments (columns) and genes (lines) with an excessive rate of missing values. The remaining missing values are imputed using a KNN-imputing scheme to produce complete data sets with no missing values (Troyanskaya, Cantor et al. 2001; Scheel, Aldrin et al. 2005; Janssen, Donders et al. 2010). The data set integration algorithm then rejects all data sets with

less than half of the query genes present; advances in technology and increased used of whole-genome gene chips renders this step less of an issue now.

5.1.3 Functional Feature Reduction Algorithm

Chapter 3 already introduced the notion that cellular functions do not typically span the entire set of experiments covered by a microarray. The first step in the analysis then is to detect a reasonable subset of experiments in which the input genes show the greatest level of similarity, as measured by the average absolute pairwise correlation coefficients, the Functional Features of the microarray and the input gene set. A novel algorithm is developed to select these experiments; this algorithm chooses *n* Functional Features from a microarray with *m* experiments where genes are highly correlated, as outlined in Alg. 1 in section 3.1.2.

This algorithm has already proven to produce biologically relevant results. Its primary shortcoming is that it is a fixed-length algorithm. Different sets of input gene lists, however, are likely to share correlated behavior over a different number of experimental conditions. While the HMM machine learning algorithm reduces the impact of any set of conditions with low similarity between the input genes, it would be preferable to limit the number of experimental conditions in response to the given set of input genes.

An extension to this algorithm measures the absolute average pairwise correlation for a window of conditions at a time. Once the maximum correlation for a window drops below a threshold the feature reduction algorithm stops. This produces a different number of conditions for each data set.

The algorithm uses a fixed-size window of size *e* to find the *n* experiments resulting in the highest average pairwise correlation coefficient for all query genes, similar to the Functional Feature selection algorithm presented in (Senf and Chen 2009). The window then slides across the microarray in steps of half the size of the window, repeating the same task to find *e/2* more experiments. This process is repeated until the within-window average pairwise correlation drops below a threshold, at which point the algorithm terminated. Functional Features are then comprised of experiments from all windows up to the one below the threshold. Algorithm 2 demonstrates this procedure. This approach naturally filters out any data not contributing to the functional relation between the query genes, while at the same time reducing the computational load on the machine learning algorithm.

Let w_i denote the set of positions corresponding to experiments in the microarray data set at the *i*th iteration of the algorithm. Alg. 2 uses a window of 10 experimental conditions, $w_0 = \{1,\ldots,10\}$. The initial iteration of this algorithm works exactly as Alg. 1 is described. Subsequent iterations move the window forward by 5 conditions, $w_1 = \{5,\ldots,15\}$. The conditions selected in iteration 0 remain fixed, which means the second iteration of the algorithm chooses additional conditions $\{11, \ldots,15\}$. Iteration 3 moves the window by 5 conditions and selects conditions $\{16,\ldots,20\}$. If the average

correlation for a window of size 10 drops below a threshold the algorithm terminates. The total set of selected conditions included all positions selected up to this point.

Algorithm 2: Window Functional Feature selection

```
1:  Select initial window start, a = 1, and window end, e
2:  Select delta, δ, by which to move the window at each iteration
3:  Select a threshold, t, to terminate algorithm
4:  Repeat until average pairwise correlation, c, over a … e is below t
5:    Select experimental conditions a … e
6:    for i = (a+δ) … e  (in the first iteration δ=0)
7:      calculate average pairwise correlation, c, over a … e
8:      for j = (a+δ)+1 … e
9:        exchange experimental features i and j
10:       calculate average absolute pairwise correlation, d
11:       if d < c then
12:         reverse, exchange features i and j again
13:       Else
14:         keep features exchanged
15:       end if
16:     end for
17:     Move window: a = a+δ, e = e+δ
18:   end for
19: Select conditions 1 … e for final set of correlated conditions
```

5.1.3. Data Set Pre-Processing

Individual Functional Feature Selection Cellular functions are limited to certain experiments, so that genes only exhibit cooperative expression behavior for some of the experiments in a microarray. This step is different for every combination of input genes and microarray. One of the issues with multiple microarray data sets is the fact

that not all data sets capture experimental conditions where the desired function is active. For example, data from an experiment designed to measure heat shock response may not be very useful to study normal metabolic functions. So a vital pre-processing step is to analyze data sets individually and determine which ones are useful and which are noise, depending on a given set of query genes. The sliding-window-based Functional Feature selection algorithm that selects a variable number of features from each data set provided to the algorithm, according to the level of cooperation exhibited by the query genes in each set.

Functional Learning using HMMs This step is not changed from the algorithm used in the previous two chapters.

Integration Two data set integration schemes are used: statistical integration and meta-analysis. Meta-analysis is performed by simple majority vote. If a gene is in the result groups of a multiple of individual microarray analysis runs, it is added to the final result set; this is the approach taken in Chapter 4.

The statistical integration approach makes use of a special property of the linear HMM machine learning algorithm: rather than attempting to normalize each data set prior to integration, the problem of different biases between data sets is inherently handled by this HMM due to its linear sequence of states. Each data set is practically limited to a section of the entire HMM, which means that biases from one data set do

not influence parameters learned from other data sets. Each data set, although combined into one matrix, remains separate with respect to the HMM states covering the corresponding experiments.

Fig. 14 compares the two approaches in a situation with 4 individual microarray data sets. It becomes clear how the linear HMM topology allows for some of the advantages of meta-analysis to be carried over to the integrative data set analysis:

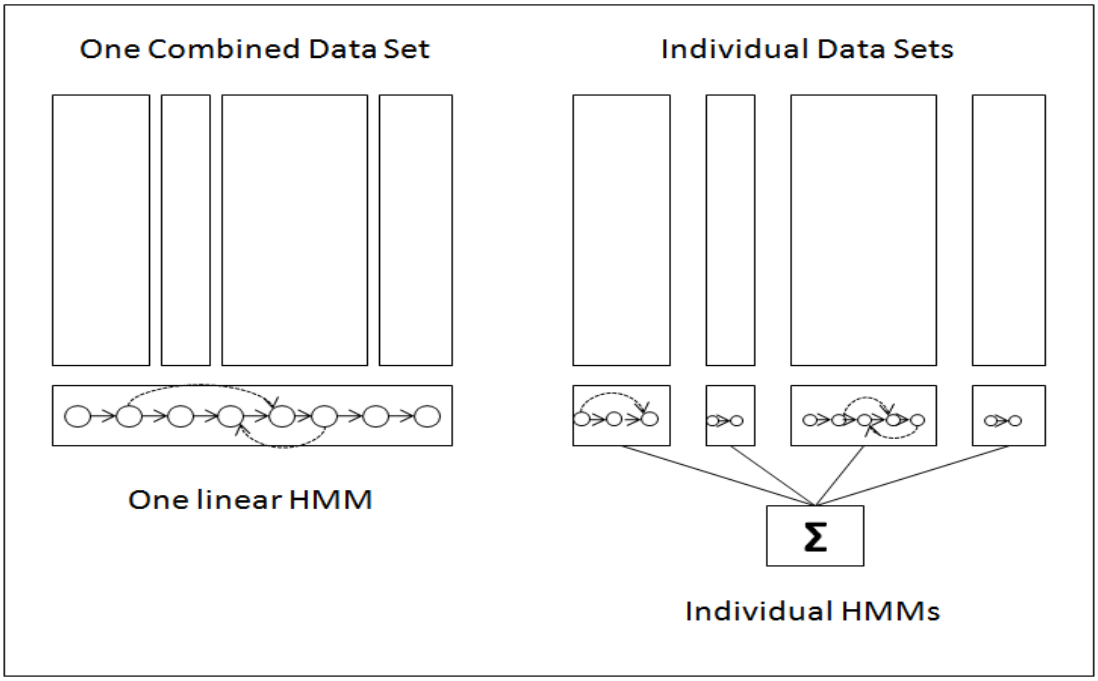

Fig. 14. Integrative vs. Meta-Analysis. The advantage of the integrative analysis is that one HMM incorporates the information from multiple arrays simultaneously. This allows for relationships between the data in multiple data sets to be detected. Meta-analysis combines the results obtained from multiple singe-data set analysis runs.

The majority vote approach works well with synthetic data but requires more attention with the biological data, because the individual HMM scores are not directly comparable in biological sets.

5.2. Results

The performance of this algorithm is tested on a synthetic data set and a series of biological data sets. Synthetic data sets are useful to explore some of the properties of the algorithm, such as the effects of increasing levels of noise, variations in parameter settings, or direct comparison to other algorithms. Synthetic data set performance, however, only has limited predictive value with respect to real biological data sets. A set of 26 time series data sets for the Fruit Fly, *D. Melanogaster*, are used to assess the biological relevance of results produced by the integrative approach in this chapter.

5.2.1. Synthetic Data Set Results.

A series of synthetic microarray subsets is generated using the same synthetic set used in Chapter 3 containing 2,500 genes and 80 experiments, with a pathway covering 220 genes and 40 experiments. This data set was divided into seven overlapping subsets of 20 experiments, each containing all 2,500 genes. This process was repeated for data sets with increasing levels of noise.

These individual data sets were then combined into one microarray data set using the window-based Functional Feature Selection algorithm. One of the effects of noise in the data sets was a reduction in the number of conditions selected by the window-

based Functional Feature selection algorithm. The correlation between query gene expression values declined in proportion to the increase in noise. Leaving the window-correlation threshold constant for all data sets caused fewer conditions to be selected with higher noise levels. Fig. 15a illustrates this decline in the synthetic data set. Simple concatenation of the data in the seven data sets would result in 140 total experiments in the combined data set.

A second advantage of the window-based Functional Feature selection algorithm is the elimination of data sets where the pathway genes are not related. Of the seven partial data sets, two contain no part of the functional module, while two overlap it to 50%. Correspondingly, fewer data is selected from these data sets. This is illustrated in Fig. 15b:

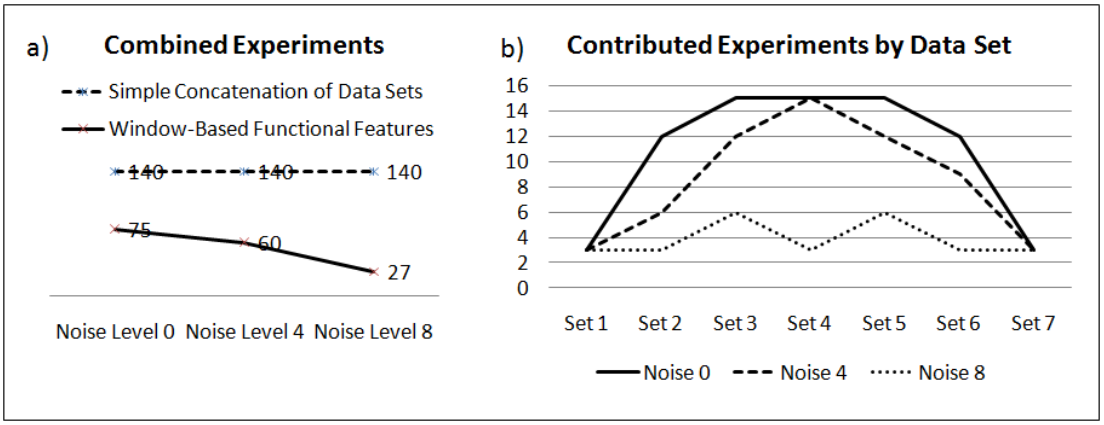

Fig. 15. Synthetic data set combination: This figure shows the number of experiments included for final analysis depending on noise levels with fixed-sized and with window-based feature selection. In this example each data set has 20 experiments. Data sets 1 and 7 do not contain any part of the functional module. In data sets 2 and 6 50% of the experiments contain a functional module; data sets 3, 4, and 5 contain the functional module foe 100% of their experiments. Part b) shows the number of features contributed to the combined data set based on noise using the window-based Functional Feature Algorithm. Part a) shows the total number of features in the combined data set at noise level 8 between widow-based and fixed-size algorithms. Fewer features mean faster algorithm runs.

Detection rates for this data set remain comparable to the best single-data set case. Looking at a medium noise level scenario the detection rate of the combined data set is higher than of any of the individual data sets as well as the meta-analysis approaches. Fig. 16 shows the ROC curves of the combined data set next to all 7 individual data sets and the 2 Meta-Analysis approaches. The 2 sets with no overlap of the functional module (sets 1,7) and the 2 sets with 50% overlap (sets 2, 6) can clearly be distinguished:

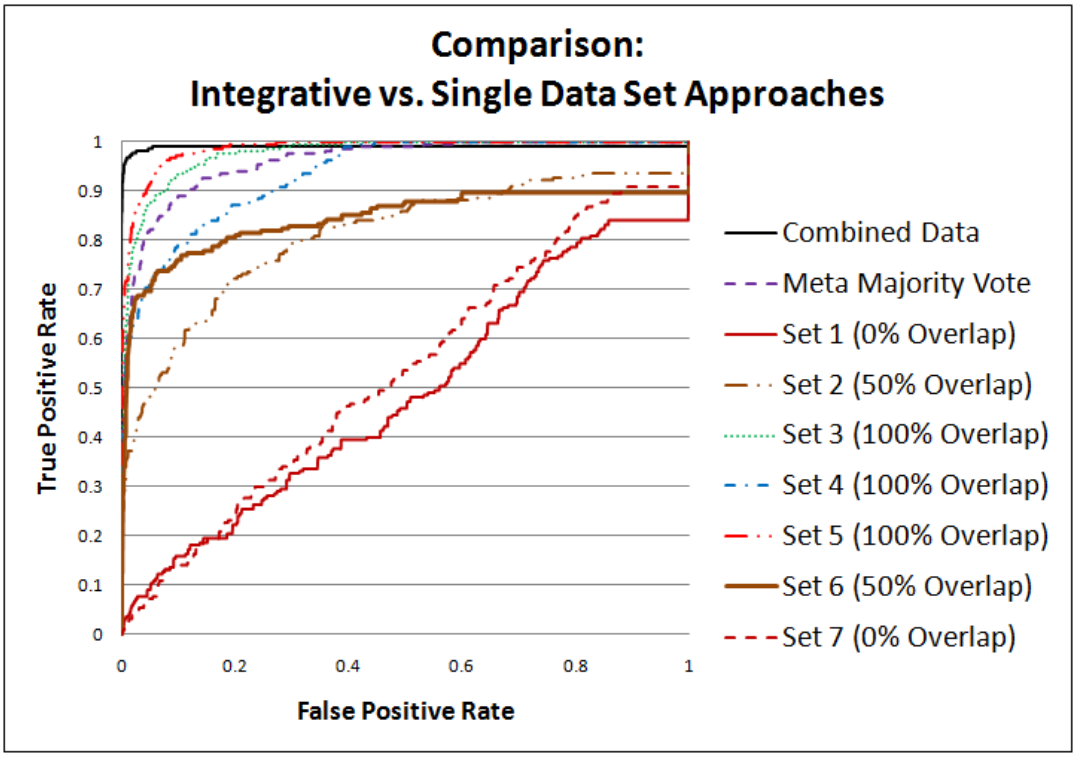

Fig. 16. Synthetic data set combination. Here individual ROC curves are shown for 7 individual data sets as well as for the two data set combination approaches. Chapter 4 showed the ability for Meta-Analysis to produce very good results, if the combined data sets are similar. What is interesting in this analysis is that the Meta-Analysis approach is actually less powerful than 2 of the 100% overlap data sets. The dynamic combination approach, however, outperforms even the best individual data sets.

It is notable here that the best results were obtained without prior knowledge of which data sets contained were best-suited for the detection task, as would be expected with a set of biological data sets. Majority voting performs almost as well as the best single data set, but is hampered by the inclusion of data sets with no part of the functional module.

5.2.2. Biological Data Set Results.

The more interesting application is in finding new information in biological organisms, such as the Fruit Fly. The fly is of particular interest because it is a simple model organism that allows some conclusions to be drawn to *H. Sapiens* (Bier 2005; Jeibmann and Paulus 2009). All time-series data sets for *D. Melanogaster* from the ArrayExpress database (Parkinson, Kapushesky et al. 2007; Parkinson, Kapushesky et al. 2009) were selected. Table 1 lists all 26 sets selected after pre-processing:

Table 1. Drosophila Data Sets Used

ArrayExpress Time-Series Microarray	Number of Genes	Number of Experiments
TABM_248	8345	12
TABM_250	8340	24
TABM_251	8339	12
E-FLYC-3	4841	12
E-GEOD-562	12233	26
E-GEOD-2784	12233	6
E-GEOD-2828	12233	12
E-GEOD-3057	12233	27
E-GEOD-3069	12233	18
E-GEOD-3826	12233	12

E-GEOD-3828	12233	12
E-GEOD-3829	12233	12
E-GEOD-3830	12233	12
E-GEOD-3831	12233	12
E-GEOD-3832	12233	12
E-GEOD-3842	12233	72
E-GEOD-4174	12233	30
E-GEOD-5147	12233	36
E-GEOD-6490	12233	12
E-GEOD-6491	12233	12
E-GEOD-6492	12233	12
E-GEOD-6493	12233	12
E-GEOD-6542	12233	48
E-GEOD-9552	12932	15
E-GEOD-10014	12233	27
E-MEXP-1287	12233	34

Data set integration is dependent on the set of query genes used, so each study will produce a different set. Time-series microarrays are best suited to study cellular processes, so for this study ATP Synthesis metabolism genes are selected. The list of genes was taken from the recent literature (Greenberg, Stockwell et al. 2008). The list of query genes contains a total of 32 genes.

The combined data set based on the ATP Synthesis query gene set contains 258 experiment and 12,233 genes. The combined analysis was run using the 125 best of the combined experiments.

Performing overrepresentation analysis with just the query genes provides a baseline for the gene functions to be expected from this query. Using the Gene Ontology (GO) terminology the dominant biological process is proton transport (GO:0015992) and the dominant molecular functions are hydrogen ion transmembrane transporter activity (GO:0015078), ATPase activity, coupled to

transmembrane movement of ions, phosphorylative mechanism (GO:0015662), and hydrogen-exporting ATPase activity, phosphorylative mechanism (GO:0008553). This forms the expected content of the result set.

The algorithm scored all genes in the combined data set based on the HMM learned from the query genes. The top-50 genes were included in the result set for further analysis and performed the same overexpression analysis on this set. The results are interesting: added to the original biological process terms are *organelle ATP synthesis coupled electron transport* (GO:0042775) and two *mitochondrial electron transport* terms (GO:0006122, GO:0006120). On the molecular function branch terms were added for *NADH dehydrogenase* activity (GO:0003954) and *oxireductase activity* (GO: 0016681).

Full overrepresentation graphs for query and result gene sets show a very high degree of overlap; Fig. 17 shows the expression values of the result gene set in the combined data set. Clearly, the resulting set of genes is functionally very closely related to the set of query genes.

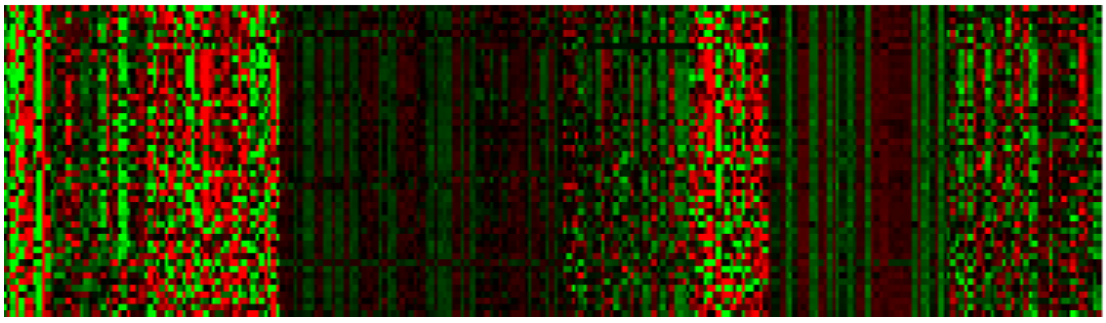

Fig. 17. Combined Result Gene Expressions. This graph shows the result genes over the integrated data set

This close alignment of overrepresented genes provides confidence that the genes in the result set are functionally related. In addition to the overrepresented genes there are 13 genes in the top-50 result set without any GO term annotation. Given the confidence in this result set it can be claimed that these genes are also functionally related to ATP synthesis. The genes are: CG3955, CG13868, CG9350, Ppat-Dpck, CG12817, CG5214, CG17556, CG18600, Sox100B, FBgn0040873, CG31030, Neb-cGP, omega.

To get an idea of the performance of the combined data set approach, the same analysis is performed individually on each of the 26 data sets. Of these, only two data sets produced top-50 results with significant similarity in overrepresented genes to the query gene set: E-GEOD-5147 and E-GEOD-3057; a significant number of data sets produced no overrepresented terms at all. This demonstrates the value of combining data sets for analysis.

For a meta-analysis approach of the same analysis task individual results are then combined and the genes repeated in a simple majority of result sets are selected. However, the results from the individual runs are too dissimilar to produce any result genes present in 14 or more of the 26 result sets.

5.3 Conclusions on Dynamic Data Set Integration

Chapter 5 presented an algorithm to dynamically combine multiple microarray data sets based on sets of query genes to be analyzed by a single machine learning algorithm. The capability of the algorithm was demonstrated on a synthetic data set and by integrating 26 *D. Melanogaster* time-series microarrays using a set of query genes related to ATP synthesis.

A feature selection algorithm was developed to select experiments from each data set where the query genes exhibit highest correlation. By using a constant correlation threshold over a sliding window of experiments, this algorithm selects a varying number of experiments from each data set according to the correlation exhibited by the query genes in each set.

The results demonstrated that the integration of individual data sets provides added information to the machine learning algorithm, resulting in improved gene detection performance over the use of individual microarrays. The result also demonstrated better performance for an integration of data sets prior to HMM training compared to an integration of multiple individual data sets HMM result scores.

The integrated analysis of *D. Melanogaster* time-series microarray data sets has provided several potential annotations for genes without prior functional annotations.

The initial results of this approach are very promising. The algorithm can be improved in two areas: results of each algorithm run could be fed back as input to a new run. This iterative process promises to further improve the performance of the algorithm, but as an automated tool it will require an improved detection of the proper cutoff threshold for genes to be included in the result set.

Another improvement towards providing this algorithm as an easy-to use tool to researchers is an improved use of increasingly parallel resources available on PCs. Much of the algorithm can be implemented in parallel to improve the runtime performance, especially with larger data sets.

Chapter 6 Discussion and Future Work

The task this dissertation set out to accomplish is to provide a semi-supervised algorithm to analyze microarray data sets for functionally related genes and functional modules based on input genes known to be related. The focus of this dissertation is on the analysis of time-series microarray data sets. This goal is accomplished by the development of a machine learning algorithm that learns the essence of the gene expression behavior of related genes to detect additional genes also participating in the same functional behavior. This algorithm out-performs currently existing algorithms on the detection of functionally related genes and functional modules.

The machine learning algorithm is based on a linear Hidden Markov Model designed to capture the sequential nature of time-series microarray experiments. This model matches the number of hidden states to the number of experiments and learns the behavior of a set of input genes provided by a knowledgeable source. Additional genes are selected from the microarray based on the closeness of a match between their gene expression behavior to the model.

A feedback loop in the training section of the algorithm allows for incremental improvements to the Hidden Markov Model and improves its ability to detect genes less directly related to the input gene set. This feedback loop also allows for the use

of very small sets of input genes, down to single gene sets, without adverse effects for the algorithm.

Meta-Analysis is shown to improve the performance of the algorithm in situations where the functional modules are obscured by very high levels of noise. If multiple similar data sets are available then combining the results from individual analysis runs significantly improves the detection performance over the individual results.

More common is the situation where multiple different microarray data sets are available for an organism. Combining these data sets dynamically for analysis with a single Hidden Markov Model enables the detection of functionally related genes even in situations where individual data sets do not perform well for an analysis task. The increased amount of data and the combination of multiple data sources increases the amount of information the machine learning algorithm can extract. This situation is most closely related to the intended target application: enabling individual researchers to utilize the vast amount of gene expression microarray data sets to detect functional modules of interest.

Two feature reduction algorithms are developed to reduce the size of the data set to be analyzed. Functional Feature Reduction is performed to produce a fixed-seized subset of experiments in a microarray data set where the input genes show a higher level of correlated behavior. A sliding-window approach allows for a variable-sized subset to be selected based on the actual correlation of a set of input genes in a

particular data set. This allows for the combination of multiple data sets without degrading performance due to a large number of features.

The performance of this approach compares favorably with the current literature. Each aspect of the algorithm has been shown to supersede the performance of currently published alternatives. The algorithm even performs well with non-time series data sets, which demonstrates its versatility. However, there are still multiple directions open to improve this algorithm for future research.

6.1 Future Research

6.1.1 Computational Complexity

One of the drawbacks of the current approach is the computational complexity. Two aspects particularly contribute to the run-time of the algorithm: (1) the number of features and the corresponding number of HMM states and (2) the calculation of a P-value for each gene based on a Parzen density function. The necessity to provide random gene scores for the construction of a PDF necessitates repeated execution of the algorithm on random input gene sets.

There are two possible solutions to the first issue. First, the implementation of the HMM training and scoring algorithm could be realized in parallel. Distributing the training task over multiple compute cores would alleviate the performance impact of the training of large models. This would be a significant contribution, as there are

currently no algorithms to run Baum-Welch training in parallel with any significant performance gains. Baum-Welch does not lend itself to parallel implementations.

Second, multiple consecutive states could be combined to be served by a single node in the HMM. This would reduce the number of nodes in the model and increase its performance. This only works, however, if the consecutive states are similar enough as to not dilute the performance of the algorithm, or to lose the connection to the sequential nature of time-series data sets.

The second problem has an easily parallelizable solution. The calculation of individual random-set gene scores is completely independent and can be distributed over multiple compute cores. The problem here is to produce a large number of random scores to generate a high-quality PDF. This type of problem indicates that the massively parallel compute engines in today's graphics hardware could be used at great advantage. OpenCL and CUDA offer several thousand compute cores inside a normal desktop machine at a low cost. The solution to this problem should be in the implementation of at least parts of the algorithm in CUDA kernels.

6.1.2 Cutoff Values

One problem of the algorithm is presently the correct selection of a cutoff value for genes to be included in the result set. At present the delta between consecutive gene scores and usage of fixed-sized result sets are successfully used. The algorithm would

benefit from a comprehensive analysis of the best cutoff to be used, based on the number, type, and quality of the input data sets.

6.1.3 Server

The intended users of the algorithm are researchers working on the functional analysis of genes and genomes. For any algorithm to find widespread adoption and usage it must be available in an easy-to use format, automating as much as possible while maintaining high performance.

This can be achieved by integrating individual aspects of the algorithm into one server (stand-alone or Web) and developing intelligent algorithms to determine the best parameter settings at run-time to take into account different input data sets.

References

1. Alter, O., P. O. Brown, et al. (2000). "Singular value decomposition for genome-wide expression data processing and modeling." Proc Natl Acad Sci U S A **97**(18): 10101-10106.
2. Arbeitman, M. N., E. E. Furlong, et al. (2002). "Gene expression during the life cycle of Drosophila melanogaster." Science **297**(5590): 2270-2275.
3. Ashburner, M., C. A. Ball, et al. (2000). "Gene ontology: tool for the unification of biology. The Gene Ontology Consortium." Nat Genet **25**(1): 25-29.
4. Baggerly, K. A., K. R. Coombes, et al. (2001). "Identifying differentially expressed genes in cDNA microarray experiments." J Comput Biol **8**(6): 639-659.
5. Bansal, M., V. Belcastro, et al. (2007). "How to infer gene networks from expression profiles." Mol Syst Biol **3**: 78.
6. Bar-Joseph, Z., G. K. Gerber, et al. (2003). "Computational discovery of gene modules and regulatory networks." Nat Biotechnol **21**(11): 1337-1342.
7. Barabasi, A. L. and Z. N. Oltvai (2004). "Network biology: understanding the cell's functional organization." Nat Rev Genet **5**(2): 101-113.
8. Barrett, T., D. B. Troup, et al. (2009). "NCBI GEO: archive for high-throughput functional genomic data." Nucleic Acids Res **37**(Database issue): D885-890.
9. Bauer, S., S. Grossmann, et al. (2008). "Ontologizer 2.0--a multifunctional tool for GO term enrichment analysis and data exploration." Bioinformatics **24**(14): 1650-1651.
10. Baum, L., T. Petrie, et al. (1970). "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains." The Annals of Mathematical Statistics **41**(1): 164-171.
11. Bergmann, S., J. Ihmels, et al. (2003). "Iterative signature algorithm for the analysis of large-scale gene expression data." Phys Rev E Stat Nonlin Soft Matter Phys **67**(3 Pt 1): 031902.
12. Bier, E. (2005). "Drosophila, the golden bug, emerges as a tool for human genetics." Nat Rev Genet **6**(1): 9-23.
13. Blake, J. A. and M. A. Harris (2008). "The Gene Ontology (GO) project: structured vocabularies for molecular biology and their application to genome and expression analysis." Curr Protoc Bioinformatics **Chapter 7**: Unit 7 2.
14. Blatt, M., S. Wiseman, et al. (1996). "Superparamagnetic clustering of data." Phys Rev Lett **76**(18): 3251-3254.
15. Brown, M. P., W. N. Grundy, et al. (2000). "Knowledge-based analysis of microarray gene expression data by using support vector machines." Proc Natl Acad Sci U S A **97**(1): 262-267.
16. Buchler, N. E., U. Gerland, et al. (2003). "On schemes of combinatorial transcription logic." Proc Natl Acad Sci U S A **100**(9): 5136-5141.
17. Cherry, J. M., C. Ball, et al. (1997). "Genetic and physical maps of Saccharomyces cerevisiae." Nature **387**(6632 Suppl): 67-73.
18. Choi, J. K., U. Yu, et al. (2003). "Combining multiple microarray studies and modeling interstudy variation." Bioinformatics **19 Suppl 1**: i84-90.

19. Davis, N. A., A. Pandey, et al. (2010). "Real-world comparison of CPU and GPU implementations of SNPrank: a network analysis tool for GWAS." Bioinformatics.
20. Dempster, A. P., N. M. Laird, et al. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." Journal of the Royal Statistical Society. Series B (Methodological) **39**(1): 1-38.
21. Dessau, R. B. and C. B. Pipper (2008). "["R"--project for statistical computing]." Ugeskr Laeger **170**(5): 328-330.
22. Dhollander, T., Q. Sheng, et al. (2007). "Query-driven module discovery in microarray data." Bioinformatics **23**(19): 2573-2580.
23. Dittrich, M. T., G. W. Klau, et al. (2008). "Identifying functional modules in protein-protein interaction networks: an integrated exact approach." Bioinformatics **24**(13): i223-231.
24. Edgar, R., M. Domrachev, et al. (2002). "Gene Expression Omnibus: NCBI gene expression and hybridization array data repository." Nucleic Acids Res **30**(1): 207-210.
25. Eisen, M. B., P. T. Spellman, et al. (1998). "Cluster analysis and display of genome-wide expression patterns." Proc Natl Acad Sci U S A **95**(25): 14863-14868.
26. Francois, J.-M. Jahmm - An implementation of HMM in Java.
27. Friedman, N. (2004). "Inferring cellular networks using probabilistic graphical models." Science **303**(5659): 799-805.
28. Gansner, E. and S. North (2000). "An open graph visualization system and its applications to software engineering." Soft\-ware --- Prac\-tice and Experience **30**(11): 1203-1233.
29. Ge, H., Z. Liu, et al. (2001). "Correlation between transcriptome and interactome mapping data from Saccharomyces cerevisiae." Nat Genet **29**(4): 482-486.
30. Gentleman, R. C., V. J. Carey, et al. (2004). "Bioconductor: open software development for computational biology and bioinformatics." Genome Biol **5**(10): R80.
31. Getz, G., E. Levine, et al. (2000). "Coupled two-way clustering analysis of gene microarray data." Proc Natl Acad Sci U S A **97**(22): 12079-12084.
32. Greenberg, A. J., S. R. Stockwell, et al. (2008). "Evolutionary constraint and adaptation in the metabolic network of Drosophila." Mol Biol Evol **25**(12): 2537-2546.
33. Grossmann, S., S. Bauer, et al. (2007). "Improved detection of overrepresentation of Gene-Ontology annotations with parent child analysis." Bioinformatics **23**(22): 3024-3031.
34. Grotkjaer, T., O. Winther, et al. (2006). "Robust multi-scale clustering of large DNA microarray datasets with the consensus algorithm." Bioinformatics **22**(1): 58-67.
35. Grutzmann, R., H. Boriss, et al. (2005). "Meta-analysis of microarray data on pancreatic cancer defines a set of commonly dysregulated genes." Oncogene **24**(32): 5079-5088.
36. Hartwell, L. H., J. J. Hopfield, et al. (1999). "From molecular to modular cell biology." Nature **402**(6761 Suppl): C47-52.
37. Henikoff, S., J. G. Henikoff, et al. (1995). "Automated construction and graphical presentation of protein blocks from unaligned sequences." Gene **163**(2): GC17-26.

38. Hermsen, R., S. Tans, et al. (2006). "Transcriptional regulation by competing transcription factor modules." PLoS Comput Biol **2**(12): e164.
39. Herrero, J. and J. Dopazo (2002). "Combining hierarchical clustering and self-organizing maps for exploratory analysis of gene expression patterns." J Proteome Res **1**(5): 467-470.
40. Hirose, O., R. Yoshida, et al. (2008). "Statistical inference of transcriptional module-based gene networks from time course gene expression profiles by using state space models." Bioinformatics **24**(7): 932-942.
41. Hong, F. and R. Breitling (2008). "A comparison of meta-analysis methods for detecting differentially expressed genes in microarray experiments." Bioinformatics **24**(3): 374-382.
42. Hughes, J. D., P. W. Estep, et al. (2000). "Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae." J Mol Biol **296**(5): 1205-1214.
43. Huttenhower, C., M. Hibbs, et al. (2006). "A scalable method for integration and functional analysis of multiple microarray datasets." Bioinformatics **22**(23): 2890-2897.
44. Ihmels, J., S. Bergmann, et al. (2004). "Defining transcription modules using large-scale gene expression data." Bioinformatics **20**(13): 1993-2003.
45. Ihmels, J., G. Friedlander, et al. (2002). "Revealing modular organization in the yeast transcriptional network." Nat Genet **31**(4): 370-377.
46. Istrail, S. and E. H. Davidson (2005). "Logic functions of the genomic cis-regulatory code." Proc Natl Acad Sci U S A **102**(14): 4954-4959.
47. Jansen, R., D. Greenbaum, et al. (2002). "Relating whole-genome expression data with protein-protein interactions." Genome Res **12**(1): 37-46.
48. Janssen, K. J., A. R. Donders, et al. (2010). "Missing covariate data in medical research: to impute is better than to ignore." J Clin Epidemiol **63**(7): 721-727.
49. Jeibmann, A. and W. Paulus (2009). "Drosophila melanogaster as a model organism of brain diseases." Int J Mol Sci **10**(2): 407-440.
50. Jin, V. X., A. Rabinovich, et al. (2006). "A computational genomics approach to identify cis-regulatory modules from chromatin immunoprecipitation microarray data--a case study using E2F1." Genome Res **16**(12): 1585-1595.
51. Kanehisa, M. and S. Goto (2000). "KEGG: kyoto encyclopedia of genes and genomes." Nucleic Acids Res **28**(1): 27-30.
52. Karplus, K., C. Barrett, et al. (1999). "Predicting protein structure using only sequence information." Proteins **Suppl 3**: 121-125.
53. Kemmeren, P., N. L. van Berkum, et al. (2002). "Protein interaction verification and functional annotation by integrated analysis of genome-scale data." Mol Cell **9**(5): 1133-1143.
54. Kholodenko, B. N., A. Kiyatkin, et al. (2002). "Untangling the wires: a strategy to trace functional interactions in signaling and gene networks." Proc Natl Acad Sci U S A **99**(20): 12841-12846.
55. Kitano, H. (2002). "Systems biology: a brief overview." Science **295**(5560): 1662-1664.
56. Kohonen, T. (1987). "Adaptive, associative, and self-organizing functions in neural computing." Appl Opt **26**(23): 4910-4918.
57. Krogh, A., M. Brown, et al. (1994). "Hidden Markov models in computational biology. Applications to protein modeling." J Mol Biol **235**(5): 1501-1531.
58. Lee, H. K., A. K. Hsu, et al. (2004). "Coexpression analysis of human genes across many microarray data sets." Genome Res **14**(6): 1085-1094.

59. Li, G., Q. Ma, et al. (2009). "QUBIC: a qualitative biclustering algorithm for analyses of gene expression data." Nucleic Acids Res **37**(15): e101.
60. Li, H. and M. Zhan (2008). "Unraveling transcriptional regulatory programs by integrative analysis of microarray and transcription factor binding data." Bioinformatics **24**(17): 1874-1880.
61. Liolios, K., I. M. Chen, et al. (2010). "The Genomes On Line Database (GOLD) in 2009: status of genomic and metagenomic projects and their associated metadata." Nucleic Acids Res **38**(Database issue): D346-354.
62. Lyne, R., R. Smith, et al. (2007). "FlyMine: an integrated database for Drosophila and Anopheles genomics." Genome Biol **8**(7): R129.
63. Manavski, S. A. and G. Valle (2008). "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment." BMC Bioinformatics **9 Suppl 2**: S10.
64. Mateos, A., J. Dopazo, et al. (2002). "Systematic learning of gene functional classes from DNA array expression data by using multilayer perceptrons." Genome Res **12**(11): 1703-1715.
65. Ng, S. K., S. H. Tan, et al. (2003). "On combining multiple microarray studies for improved functional classification by whole-dataset feature selection." Genome Inform **14**: 44-53.
66. Owen, A. B., J. Stuart, et al. (2003). "A gene recommender algorithm to identify coexpressed genes in C. elegans." Genome Res **13**(8): 1828-1837.
67. Pang, H., A. Lin, et al. (2006). "Pathway analysis using random forests classification and regression." Bioinformatics **22**(16): 2028-2036.
68. Park, T., S. G. Yi, et al. (2003). "Statistical tests for identifying differentially expressed genes in time-course microarray experiments." Bioinformatics **19**(6): 694-703.
69. Parkinson, H., M. Kapushesky, et al. (2009). "ArrayExpress update--from an archive of functional genomics experiments to the atlas of gene expression." Nucleic Acids Res **37**(Database issue): D868-872.
70. Parkinson, H., M. Kapushesky, et al. (2007). "ArrayExpress--a public database of microarray experiments and gene expression profiles." Nucleic Acids Res **35**(Database issue): D747-750.
71. Parzen, E. (1962). "On the estimation of a probability density function and mode." Annals of Mathematical Statistics **33**: 1065-1076.
72. Pena, J. M., J. Bjorkegren, et al. (2005). "Growing Bayesian network models of gene networks from seed genes." Bioinformatics **21 Suppl 2**: ii224-229.
73. Pereira-Leal, J. B., A. J. Enright, et al. (2004). "Detection of functional modules from protein interaction networks." Proteins **54**(1): 49-57.
74. Petti, A. A. and G. M. Church (2005). "A network of transcriptionally coordinated functional modules in Saccharomyces cerevisiae." Genome Res **15**(9): 1298-1306.
75. Pieler, R., F. Sanchez-Cabo, et al. (2004). "ArrayNorm: comprehensive normalization and analysis of microarray data." Bioinformatics **20**(12): 1971-1973.
76. Prelic, A., S. Bleuler, et al. (2006). "A systematic comparison and evaluation of biclustering methods for gene expression data." Bioinformatics **22**(9): 1122-1129.
77. Rabiner, L. R. (1989). "A tutorial on hidden Markov models and selected applications in speech recognition." Proceedings of the IEEE **77**(2): 257-286.
78. Ravasz, E. and A. L. Barabasi (2003). "Hierarchical organization in complex networks." Phys Rev E Stat Nonlin Soft Matter Phys **67**(2 Pt 2): 026112.

79. Ravasz, E., A. L. Somera, et al. (2002). "Hierarchical organization of modularity in metabolic networks." Science **297**(5586): 1551-1555.
80. Reiter, L. T., L. Potocki, et al. (2001). "A systematic analysis of human disease-associated gene sequences in Drosophila melanogaster." Genome Res **11**(6): 1114-1125.
81. Schafer, J. and K. Strimmer (2005). "An empirical Bayes approach to inferring large-scale gene association networks." Bioinformatics **21**(6): 754-764.
82. Scheel, I., M. Aldrin, et al. (2005). "The influence of missing value imputation on detection of differentially expressed genes from microarray data." Bioinformatics **21**(23): 4272-4279.
83. Senf, A. and X. W. Chen (2009). "Identification of genes involved in the same pathways using a Hidden Markov Model-based approach." Bioinformatics **25**(22): 2945-2954.
84. Shi, H., B. Schmidt, et al. (2010). "A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware." J Comput Biol **17**(4): 603-615.
85. Snel, B., P. Bork, et al. (2002). "The identification of functional modules from the genomic association of genes." Proc Natl Acad Sci U S A **99**(9): 5890-5895.
86. Spellman, P. T., G. Sherlock, et al. (1998). "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization." Mol Biol Cell **9**(12): 3273-3297.
87. Spirin, V. and L. A. Mirny (2003). "Protein complexes and functional modules in molecular networks." Proc Natl Acad Sci U S A **100**(21): 12123-12128.
88. Stark, C., B. J. Breitkreutz, et al. (2006). "BioGRID: a general repository for interaction datasets." Nucleic Acids Res **34**(Database issue): D535-539.
89. Suchard, M. A., Q. Wang, et al. (2010). "Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures." J Comput Graph Stat **19**(2): 419-438.
90. Tamayo, P., D. Slonim, et al. (1999). "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation." Proc Natl Acad Sci U S A **96**(6): 2907-2912.
91. Tanay, A., R. Sharan, et al. (2004). "Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data." Proc Natl Acad Sci U S A **101**(9): 2981-2986.
92. Tavazoie, S., J. D. Hughes, et al. (1999). "Systematic determination of genetic network architecture." Nat Genet **22**(3): 281-285.
93. Team, R. D. C. R: A Language and Environment for Statistical Computing.
94. Tornow, S. and H. W. Mewes (2003). "Functional modules by relating protein interaction networks and gene expression." Nucleic Acids Res **31**(21): 6283-6289.
95. Trapnell, C. and M. C. Schatz (2009). "Optimizing Data Intensive GPGPU Computations for DNA Sequence Alignment." Parallel Comput **35**(8): 429-440.
96. Troyanskaya, O., M. Cantor, et al. (2001). "Missing value estimation methods for DNA microarrays." Bioinformatics **17**(6): 520-525.
97. Venter, J. C., M. D. Adams, et al. (2001). "The sequence of the human genome." Science **291**(5507): 1304-1351.

98. Wistrand, M. and E. L. Sonnhammer (2005). "Improved profile HMM performance by assessment of critical algorithmic features in SAM and HMMER." BMC Bioinformatics **6**: 99.
99. Wong, S. L., L. V. Zhang, et al. (2004). "Combining biological networks to predict genetic interactions." Proc Natl Acad Sci U S A **101**(44): 15682-15687.
100. Wren, J. D. (2009). "A global meta-analysis of microarray expression data to predict unknown gene functions and estimate the literature-data divide." Bioinformatics **25**(13): 1694-1701.
101. Wu, C. J. and S. Kasif (2005). "GEMS: a web server for biclustering analysis of expression data." Nucleic Acids Res **33**(Web Server issue): W596-599.
102. Wu, H., Z. Su, et al. (2005). "Prediction of functional modules based on comparative genome analysis and Gene Ontology application." Nucleic Acids Res **33**(9): 2822-2837.
103. Yan, X., M. R. Mehan, et al. (2007). "A graph-based approach to systematically reconstruct human transcriptional regulatory modules." Bioinformatics **23**(13): i577-586.
104. Yang, J., Y. Zou, et al. (2009). "Identifying differentially expressed genes in human acute leukemia and mouse brain microarray datasets utilizing QTModel." Funct Integr Genomics **9**(1): 59-66.
105. Yi, G., S. H. Sze, et al. (2007). "Identifying clusters of functionally related genes in genomes." Bioinformatics **23**(9): 1053-1060.
106. Yuh, C. H., H. Bolouri, et al. (1998). "Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene." Science **279**(5358): 1896-1902.