BIOINFORMATICS ORIGINAL PAPER Vol. 25 no. 22 2009, pages 2945–2954
doi:10.1093/bioinformatics/btp521

Systems biology

Identification of genes involved in the same pathways using a Hidden Markov Model-based approach

Alexander Senf and Xue-wen Chen*

Department of Electrical Engineering and Computer Science, University of Kansas, 1520 West 15th Street, Lawrence, KS 66045, USA

Received on March 24, 2009; revised and accepted on August 25, 2009

Advance Access publication August 31, 2009

Associate Editor: Jonathan Wren

ABSTRACT

Motivation: The sequencing of whole genomes from various species has provided us with a wealth of genetic information. To make use of the vast amounts of data available today it is necessary to devise computer-based analysis techniques.

Results: We propose a Hidden Markov Model (HMM) based algorithm to detect groups of genes functionally similar to a set of input genes from microarray expression data. A subset of experiments from a microarray is selected based on a set of related input genes. HMMs are trained from the input genes and a group of random gene input sets to provide significance estimates. Every gene in the microarray is scored using all HMMs and significant matches with the input genes are retained. We ran this algorithm on the life cycle of Drosophila microarray data set with KEGG pathways for cell cycle and translation factors as input data sets. Results show high functional similarity in resulting gene sets, increasing our biological insight into gene pathways and KEGG annotations. The algorithm performed very well compared to the Signature Algorithm and a purely correlation-based approach.

Availability: Java source codes and data sets are available at http://www.ittc.ku.edu/~xwchen/software.htm

Contact: xwchen@ittc.ku.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

A milestone in molecular and computational biology has been reached with the sequencing of a complete human genome, marking the beginning of the post-genomic era (Venter *et al.*, 2001). With a wealth of genomic sequence data now available from multiple species, the main goal in molecular and computational biology has shifted to better understand the functions of genes in a cellular context. Genes encode the information necessary to build proteins; functions using certain proteins cause a higher rate of expression for the associated genes. 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. 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 (Hartwell, 1999; Ravasz and Barabási, 2002; Ravasz *et al.*, 2002). 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 (Barabási and Oltavi, 2004; Wu *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 interaction networks. Complex interactions of cellular functions can be viewed as interaction networks of simpler subunits and functional modules (Friedman, 2004; Kholodenko *et al.*, 2002; Petti and Church, 2005).

The technology most prevalent in the study of genetic functional modules is the microarray. Microarray gene expression 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 indicate the level of activity of a gene, which points to genes that are responding to the conditions set by the experiment.

Functionally related genes show some measure of similarity in their expression profiles. Standard approaches for detecting functional modules are to cluster these genes using hierarchical clustering techniques (Eisen, 1998; Grotkjær *et al.*, 2006), *k*-means clustering (Tavazoie, 1999), techniques such as Self Organizing Maps (SOM) (Tamayo *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. 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 *et al.*, 2000) introduced the notion that clusters contain subsets of genes and experiments. A notable work using this idea is the Signature Algorithm (Bergmann *et al.*, 2003; Ihmels *et al.*, 2002, 2004). This work develops the concept of a transcriptional module as a subset of genes and experimental conditions. A set of seed genes is first scored with all experimental conditions and highest-scoring conditions are retained. Then all genes in the microarray are scored with a separate scoring function over the subset of selected conditions. All high-scoring genes are included in the result set. Some of the original

*To whom correspondence should be addressed.

seed genes may be dropped if they score low and new genes are added (Ihmels *et al.*, 2002). The algorithm may then iterate until it converges on a set of genes which is taken to be a functional module (Bergmann *et al.*, 2003; Ihmels *et al.*, 2004).

Similar work is also performed at the level of protein–protein interaction data (Dittrich *et al.*, 2008; Pereira-Leal *et al.*, 2004; Snel *et al.*, 2002; Spirin and Mirny, 2003). 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. 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 *et al.*, 2004) or physical interactions, protein function, and network topology characteristics (Wong *et al.*, 2004) across multiple diverse data sources.

This article analyzes microarray data of the fruit fly (*Drosophila melanogaster*) with the purpose of detecting genes participating in cellular pathways. The resulting groups of related genes produced by our algorithm can be used as basis for the recovery of regulatory interaction networks and for the assignment of functional annotations to genes with few or no current annotations. Our machine-learning approach is related to remote homology detection; it is capable of detecting weak similarity signals in data sets where other algorithms produce no significant results. The open nature of the machine-learning algorithm also allows us to study data sets based on different sources of similarities

The next section provides our motivation and an overview of the proposed algorithm, followed by a detailed description of the important steps and the data sets used. Results are presented and discussed in the following sections and the article ends with a conclusion and outlook on further research.

2 METHODS

We apply Hidden Markov Models (HMMs) to the problem of detecting genes related to a group of input genes in microarray data sets because of the excellent capability of HMMs in handling time-series data (Rabiner, 1989). The problem of searching for functionally related genes is similar to the protein remote homology detection problem, which is concerned with finding protein sequences related to a group of proteins with known structural similarities. The information from multiple protein sequences is commonly presented as a Position-Specific Scoring Matrix (PSSM, also referred to as 'profiles') (Gribskov et *al.*, 1987) describing the probability of finding amino acids at a position in a protein sequence, based on a set of input proteins. Krogh *et al.* (1994) introduced the idea of using HMMs to model profiles ('Profile HMMs'). Profile HMMs represent the probability distributions of amino acids at all positions in the multiple sequence alignment. The topology of a Profile HMM is a linear series of states that are traversed sequentially, starting from the leftmost (starting) node.

2.1 Algorithm overview

The algorithm proposed in this article is designed to work with input genes based on common Kyoto Encyclopedia of Genes and Genomes (KEGG) pathway (Kanehisa *et al.*, 2004), common functional properties, or common Gene Ontology (GO) annotations (The Gene Ontology Consortium, 2000). 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 HMM is trained on the input data set, consisting of the gene expression values of the input genes at the subset of experimental conditions from Step 3. Additional sets of 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 2.4.

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

2.2 Microarray data sets and data preparation

There are several databases publishing microarray data sets. Prominent among them are The European Bioinformatics Institute's (EBI) ArrayExpress (Parkinson *et al.*, 2007) and the National Center for Biotechnology Information's (NCBI) Gene Expression Omnibus (GEO) data base (Barrett *et al.*, 2006; Edgar *et al.*, 2002). The data set used in this article is selected from the GEO database, accession number GDS191, the life cycle of Drosophila (Arbeitman *et al.*, 2002).

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.

At this stage our algorithm imputes missing values using a KNN-based method (Troyanskaya *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.

A subset of genes is then selected with an average absolute pairwise correlation coefficient above a certain threshold. We use the commonly used Pearson correlation coefficient to calculate the similarity between two gene expression profiles. The input for this algorithm is a set of genes known to

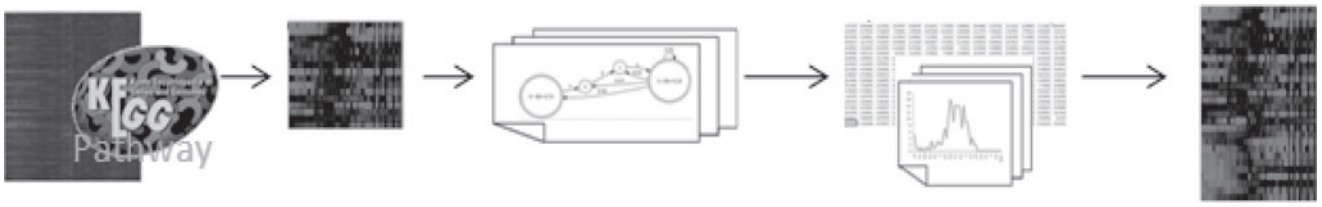

Fig. 1. Algorithm outline: given a microarray and one pathway from the KEGG database, functional features (subset of features where pathway genes show very high correlation) are calculated. This input set, along with multiple random-generated input sets with the same number of genes, is used to train 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.

be involved in the same pathway or cellular function. In this article we select our list of input genes list from the KEGG pathway database.

2.3 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. We are calling these conditions 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 Algorithm 1.

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
```

Algorithm 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 pair-wise correlation of the input genes is largest. The set of input genes, now each 60 elements long, is then used to train a HMM. 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.

2.4 Functional learning using HMMs

2.4.1 HMM Are machine-learning tools for modeling hidden (unobservable) generative processes from observable events (Baum *et al.*, 1970). The hidden process in an HMM is assumed to be a first-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 protein sequence alignments.

HMMs are essentially finite state machines that produce output symbols according to an emission probability distribution B at each state [$B=\{b_i\}$, $b_i=P(O_t=v_k|X_t=a_{it})$], given a sequence of observations, $O=(o_1,o_2,\ldots,o_T)$ of length T, and observation alphabet, $V=(v_1,v_2,\ldots,v_M)$, of all events that could possibly be observed. M denotes the number of discrete observable events. 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 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, which is a subset of V. In continuous-value applications V is obtained by binning the range of observable values into M bins. In our algorithm V corresponds to a set of numbers denoting the bin into which a microarray gene expression value falls, $V=\{1,\ldots,M\}$. In the most general HMM model the length of an observation sequence, T, can be shorter or longer than the number of states.

Sequential applications of HMMs often use profile HMMs (Eddy, 1998; Grundy *et al.*, 1997; Karplus *et al.*, 1999). Profile HMMs are very popular in remote homology detection (Karplus *et al.*, 1999), sequence alignments (Krogh *et al.*, 1994), and other sequential applications such as motif detection and database search (Henikoff *et al.*, 1995). In this algorithm we build a profile of the gene expression patterns for genes with the same function, or genes in the same pathway. HMMs are very well suited for this task. Finding related genes, then, becomes similar to finding homologous protein sequences, except that we look at gene expression values instead of amino acid residues.

Figure 2 shows the topology of the HMMs used in our application. A HMM represents the probability distribution of gene expression profiles at different time points. In this model, a hidden state is related to a time point and the emissions at each state represent gene expression levels (e.g. up- and down-regulated). Thus, the emission probabilities are the probability distributions for the expected gene expression levels at a particular time (state) and the transition probabilities describe the probability of the expression level of a gene at time $t+1$ given its expression level at time t. In our model, HMMs have a matching number of states and observation sequence events ($N=T$). Our model deviates from profile HMMs described in Krogh *et al.* (1994) and the approach taken in BLOCKS (Henikoff *et al.*, 1995) 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 our HMM models tend to converge to true sequential HMMs for the majority of nodes; exceptions generally have very low state transition probabilities. This better enables the algorithm to model noise in the data set.

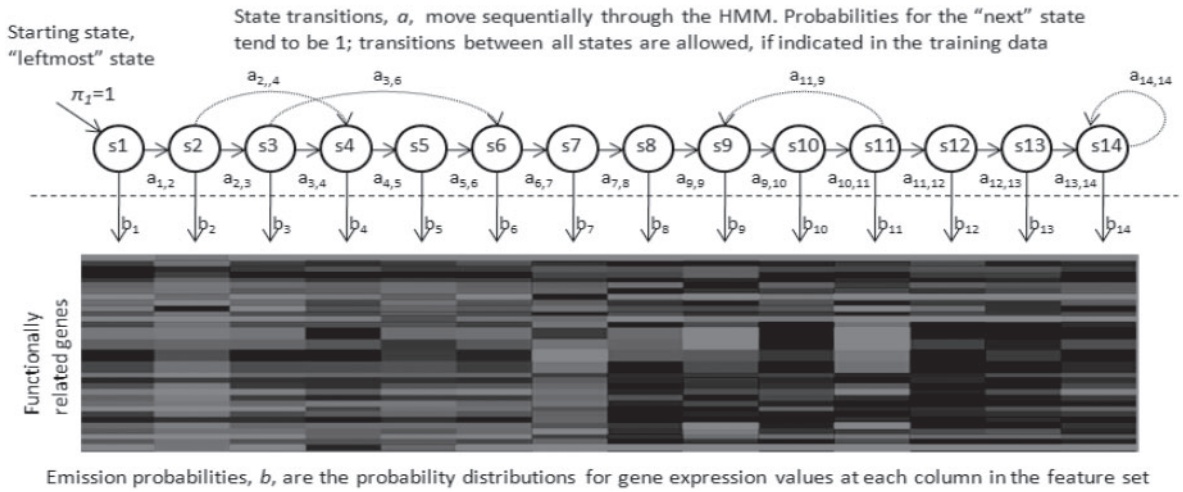

Fig. 2. A HMM with 14 states models a series of functionally related genes based on their microarray gene expression values. The topology of the HMM is very linear, with state transitions $a_{i,i+1}$ tending towards 1. If the training data indicates an advantage for transitions between other states our model will accommodate it. Emissions at each state are gene microarray expression values. This HMM learns the 'profile' of gene expression for a group of related genes. Other genes with significantly high probabilities to be generated by this HMM ('HMM score') are taken to be functionally related. The algorithm presented in this article uses HMMs with 60 states.

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

These parameter settings provide an early approximation of the structure of the final trained HMM, without locking parameters firmly into place. Profile HMMs usually have a fixed starting state and always transition to a state associated with the next observation. If the training data demands changes to the basic HMM model then our model can accommodate that. Otherwise it converges to a basic HMM model. Figure 1 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 we expect π_1 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 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 Expectation Maximization (EM) algorithms and are guaranteed to converge to a local maximum (Dempster *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 HMM we expect I 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. We can view Q as the hidden variables and define $Q(\lambda, \lambda^g)$ as the auxiliary functions to be maximized iteratively:

$$Q(\lambda, \lambda^g) = E[\log p(O_{1:N}, Q_{1:N}|\lambda)|O_{1:T}, \lambda^g], \quad (2)$$

$$= \sum_{q_{1:N}} [\log p(o_{1:N}, q_{1:N}|\lambda)]|p(o_{1:T}, q_{1:N}|\lambda^g) \quad (3)$$

Each iteration of the Baum–Welch EM algorithm updates parameters λ by maximizing Q with respect to λ:

$$\lambda^i = \arg\max_{\lambda} Q(\lambda, \lambda^i). \quad (4)$$

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

2.5 Random HMM generation and result genes

The fully trained HMM is used to score every gene in the microarray, using only the Functional Features identified in the previous step. The score is calculated as the probability that an observation sequence, i.e. a gene expression profile, was generated by the HMM. A high probability score indicates a high likelihood that the gene was generated by the HMM and belongs to the same functional module as the input genes used for training. We are looking for genes that score very high given the trained HMM. Our result group of genes then consists of all genes from the microarray with statistically significant HMM scores.

To estimate the significance of the obtained probability scores, we generate multiple groups of genes randomly selected from the entire microarray; each group contains the same number of genes as the input gene set, and the same training process is then used to produce fully trained HMM models from these random groups. Each gene in the microarray is scored again using all random-trained HMM models. The resulting scores are used to estimate the Parzen density distribution function (PDF) (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-trained HMM scores, the gene is included in the functional module set.

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

$$\hat{p}(x) = \frac{1}{V} \sum_{i=1}^{n} K_i(x_i). \quad (5)$$

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 Equation (6):

$$G(x; \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \quad (6)$$

Here d refers to the dimensionality of the data; μ is the center of the kernel, Σ and the covariance matrix.

Significance estimates are calculated as the p-value of the original HMM score given the PDF derived from scores of all random-trained HMMs. 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. Genes with significantly high scores are added to the input genes and form the set of result genes

2.6 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 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.

In our analysis, we characterize the biological function of the input gene set by the list of GO term biological process annotations of all genes in the set. The modules we find are parts of the complex biological machinery of the cell, so we are primarily interested in whether genes are involved in related biological processes.

The list of GO terms for any new gene in the result set is first enlarged by including all direct parent and child terms in the GO term hierarchy. Then all GO terms from gene homologues and orthologs listed in the FlyMine database (Lyne *et al.*, 2007) are added, and annotations from interacting proteins from the BioGRID database (Stark *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 lists, and by comparison of statistically overrepresented GO terms (given only the genes in the microarray) in both sets. We are looking for high match rates in both categories.

Overrepresentation analysis was performed using the freely available Ontologizer software (Bauer *et al.*, 2008; Grossmann *et al.*, 2007). 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 we used. Calculations were 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 RESULTS

We performed tests 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 Figure 3b.

A second synthetic data set (SA set) was generated based on Bergman *et al*. (2003) to compare the performance of our 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 we generated is shown in Figure 3a.

The biological microarray used in this article was taken from GEO, a large repository hosted by the National Institutes of Health (NIH) (Barrett *et al.*, 2006; Edgar *et al.*, 2002). The data set accession is GDS 191, which is the Life Cycle of Drosophila (Arbeitman *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: $\geq$25%, rows: $\geq$33% missing), due to limitations in the ability to impute very large blocks of missing values. Any remaining missing values are imputed, resulting in a 2646 $\times$ 158 gene expression matrix with no missing values.

In sections 3.2 and 3.3, our algorithm is run with two biological input gene data sets: (i) the KEGG cell cycle reference pathway, and (ii) the KEGG dme03012 pathway (translation factors).

The most closely related algorithm to the work presented in this paper is the Signature Algorithm (Ihmels *et al.*, 2002). Similar to the Signature Algorithm, we select a subset of experimental conditions before proceeding with the selection of similar genes. We found, however, that the Signature Algorithm did not perform well with the data sets we were studying. In section 3.4 we provide comparative results obtained with an implementation of the Signature Algorithm as it is described in the literature.

3.1 Synthetic data set—evaluation

3.1.1 Feature selection Our 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. We use a synthetic microarray data set to evaluate the ability of our Functional Feature algorithm to select the correct features with respect to an increasing amount of noise.

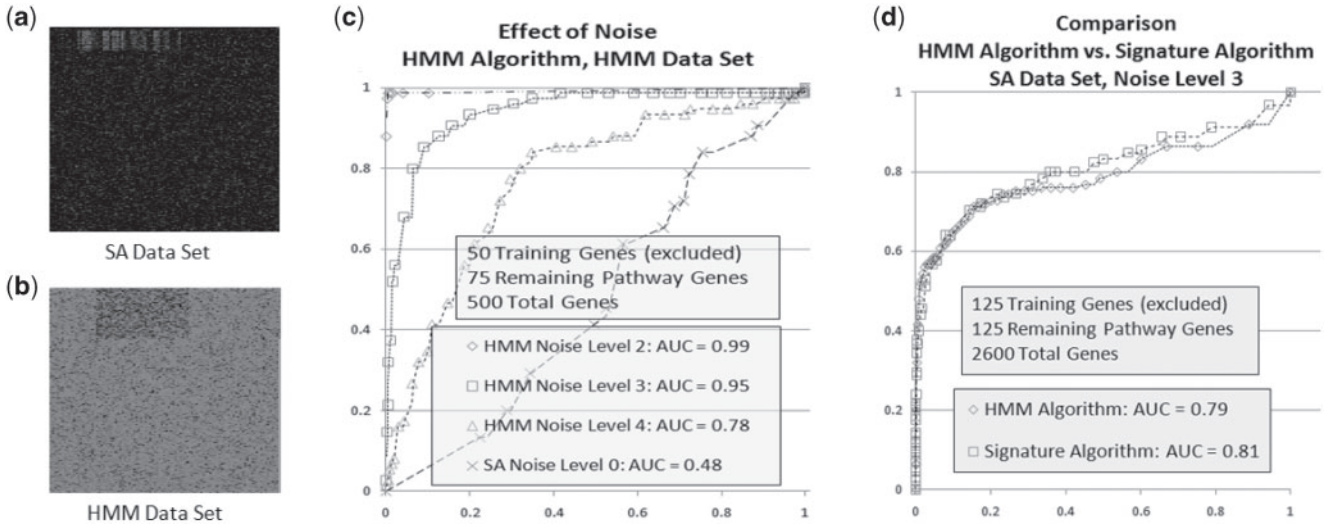

Fig. 3. (**a**) Synthetic Signature Algorithm (SA) data set, created according to Bergman *et al*. (2003) (see Section 3.1 for details). The pathway contains 250 genes and extends over 40 experiments. The entire data set contains 2600 genes and 100 experiments. (**b**) HMM data set is created by generating 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**) ROC curves using our HMM Algorithm and the HMM data set. We used 50 genes 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. An ROC curve for the Signature Algorithm on the HMM data set at Noise Level 0 (no noise added) is included to demonstrate that the SA algorithm does not perform well on the HMM data set. (**d**) Shows a comparison between the Signature Algorithm and our HMM Algorithm using the SA data set. In this test we use 125 randomly selected training genes and 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.

Using the HMM data set we select the first 50 genes of the pathway as input gene list. The purpose of this step in our algorithm is to select a good set of n features for the HMM training and evaluation algorithm.

Our 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 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.

3.1.2 Gene selection Using the two synthetic data sets (HMM set) and (SA set) we evaluate the performance of our HMM algorithm in comparison to the Signature Algorithm and the effect of increasing noise on the ability to recover pathway genes. Figure 3c and d shows the 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. It was embedded in a 500×100 microarray consisting of randomized expression values. To simulate noise, Gaussian noise multiplied by factors 0–4 was added to the entire microarray. Our algorithm shows a good ability to recover pathway genes even in noisy data.

The Signature Algorithm does not work well with this synthetic microarray (Fig. 3c). 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, we generated a 2600×100 microarray using the same method described in Bergman *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. We then added Gaussian noise to the entire microarray. This data set works with the strengths of the Signature Algorithm.

Our analysis in Figure 3d shows that the HMM Algorithm is competitive to the Signature Algorithm on this data set.

3.1.3 Run time evaluation Our 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. We tested the timing of these parts of the algorithm 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 s with the synthetic

HMM data set. The complexity bound for this state is given by $O[N(N-M)K\log(K)]$, where N is the number of features, M is the number of experimental conditions in the microarray data set, and K is the number of genes in the gene input list.

(2) HMM training and gene scoring. 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. Our tests were performed using 400 random HMMs and were run using 4 Java threads on a 2.4 GHz quad-core Intel Q6600 CPU (running at an elevated 3.0 GHz). Run time for this part of the algorithm using the synthetic data set with 400 random HMMs is 7 min, 46 s. The Training step uses the Baum–Welch algorithm with a complexity bound of $i*O(K^2N)$, where i is the number of iterations, K is the number of states and N the length of the sequence; we use 500 iterations in our training step. With our linear HMM topology the number of states tends towards 1 for each observation, but the step is repeated for multiple HMMs, so the bound is given as: $h*i*O(N)$, where h is the number of HMMs used in an algorithm run. In a similar fashion the bound for scoring is given for the Forward algorithm as $h*O(K^2N)$ where K also tends towards 1.

(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 s, including disk output of the results. The complexity of this step only depends on the number of scores provided, which corresponds to the number of HMMs produced in the previous step. The complexity bound is thus constant, $O(K)$, where K is the number of HMM scores, which is independent of the size of the microarray and gene input set.

The total run time using 50 input genes is 9 min, 12 s. Run times vary slightly between runs due to the random nature of this algorithm. The major factor for the total run time is the use of random HMMs to estimate the significance of a gene score. Each random HMM is generated and scored using the same algorithm as the original HMM, and doubling the number of HMMs nearly doubles its run time.

3.2 KEGG cell cycle data set (derived)

There currently is no cell cycle pathway available for *D.melanogaster* in KEGG, so the data set 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. Eighteen 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.

Eight-hundred 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 as used as significance threshold for the inclusion of new genes, given the PDF. Our algorithm produced 27 new genes with these settings, 23 of them with current GO term annotations.

Figure 4a 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. Eighty-three percent 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 we predict genes with no current GO term annotations 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'.

3.3 KEGG translation factors (dme03012) data set

The translation factor pathway genes taken from KEGG (KEGG Database) exhibit a low average pairwise correlation in our microarray data set. 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 we found two subgroups of five 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. The overrepresentation graph for the pathway created by Ontologizer is shown in Supplementary Figure 1.

This is an interesting situation. To find genes associated with the Translation factors pathway, we ran the algorithm twice—once for each of the subgroups. The results were combined again for the analysis step.

New genes were 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 results are shown in Figure 4b. The two independent runs produced 6 and 19 new genes, respectively, for a total of 25 genes added to the original 10 genes. Figure 4c 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 genes is annotated with a different pathway ('purine metabolism') and one gene with a single, but different GO term ('protein amino acid glycosylation'), producing a match rate of 88%. Eight genes have no prior GO term or pathway annotation and can be annotated with the translation factors pathway.

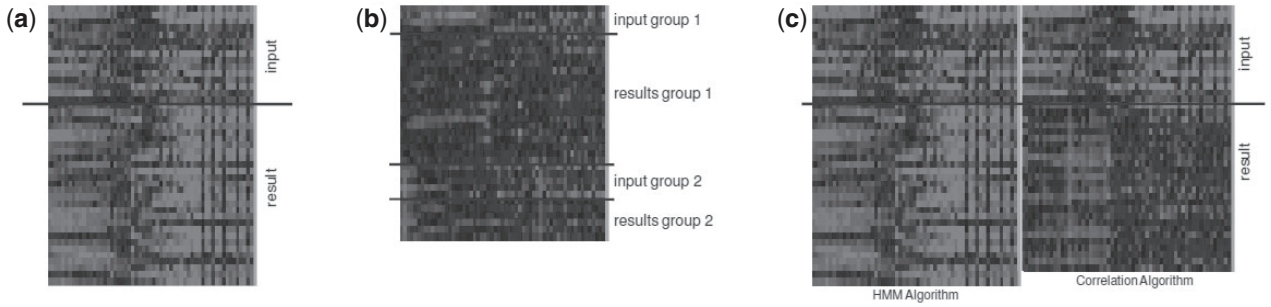

Fig. 4. (**a**) Input and result genes for the cell cycle pathway. Result genes show a very similar expression behavior as the input genes. (**b**) Two input gene groups, and result genes for translation factors pathway. In this case the pathway was split into two groups of genes, each producing their own result genes. Expression levels are generally low, but show similarities for each group. (**c**) Comparison between HMM- and correlation-based re-sults 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.

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. The overrepresentation graph produced by the Ontologizer software is listed in Supplementary Figure 2.

3.4 Discussion—cell cycle data comparisons

3.4.1 Why not just use correlation? Our algorithm 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.

We ran the purely correlation-based algorithm 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? Figure 4c shows a comparison between HMM- 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.

3.4.2 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.

This algorithm has proven to run well with combined yeast microarray data sets (Ihmels *et al.*, 2002). We have run the Signature Algorithm with the same data sets used in our 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 our algorithm.

While we were able 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, we ran the algorithm 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. These results are shown in Supplementary Table 1.

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. Our 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, we were not able to generate result data sets with recurrent genes produced by slightly modified input data sets. The microarray we used apparently does not contain features strong enough to be detected by the Signature Algorithm.

This suggests that our algorithm 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.

4 CONCLUSION

The purpose of this algorithm is to detect functionally related genes for cellular pathways and functional modules based on similarities in the gene expression patterns of microarray data sets. To achieve this goal, we propose a machine-learning algorithm that selects new genes from a microarray data set based on characteristics in the expression profiles of a group of input genes.

Data preprocessing ensures that the microarray data set is complete and the input gene data set shows a reasonable level of similarity in the selected microarray. A subset of experimental conditions is selected from the microarray where the input genes have the highest absolute pairwise correlation coefficient. Multiple HMMs are then trained using the subset of conditions, one using the set of input genes, all others using random groups of genes. Each gene in the microarray is then evaluated and the statistical significance of the score produced by the input-gene-trained HMM is evaluated using the PDF derived from random-gene-trained HMMs. Genes with *p*-values below a threshold are added to the result gene data set.

Existing algorithms do not perform well and do not produce biologically meaningful results with the data set we are studying. The results presented in this article show how well our HMM-based algorithm performs with a difficult data set compared to its closest relative, the Signature Algorithm and with correlation-only based algorithms. New genes were added to the original input gene data set with evident functional ties to the input genes, allowing for the characterization of genes with previously unknown functions with putative pathway and GO term annotations.

Our algorithm has shown to provide good results from a single microarray data set, generating modules of functionally related genes from various sources of input. We propose to expand this algorithm in two primary directions: integration of multiple microarray data sets in the analysis, and iterating the current algorithm until it converges on a set of genes and conditions.

Other proposed enhancements to the current algorithm include improvements in the run time performance by utilizing the parallel architecture of current video graphics hardware, and to use this algorithm to perform semi-supervised clustering operations. Given the nature of this machine-learning algorithm it requires prior knowledge to train one HMM for each cluster. This may be used to cluster a set of genes based on all KEGG pathways.

Other methods of selecting features in microarrays have been proposed (Schäfer and Strimmer, 2005). While our Functional Feature algorithm has been shown to perform very well, we also propose to improve this step in our algorithm to enable us to apply it to a larger range of data sets where the similarity relationships may not be captured optimally by correlating observation sequences.

Funding: US National Science Foundation Award IIS-0644366.

Conflict of interest: none declared.

REFERENCES

Arbeitman,M.N. *et al.* (2002) Gene expression during the life cycle of *Drosophila melanogaster*. *Science*, **297**, 2270–2275.

Barabási,A.-L. and Oltvai,Z.N. (2004). Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.*, **5,** 101–113.

Barrett,T. *et al.* (2006) NCBI GEO: mining tens of millions of expression profiles—database and tools update. *Nucleic Acids Res.*, **35** (Database Issue), D760–D765.

Bauer,S.S. *et al.* (2008) Ontologizer 2.0—a multifunctional tool for GO term enrichment analysis and data exploration. *Bioinformatics,* **24**, 1650–1651.

Baum,L.E. *et al.* (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statist.*, **41**, 164–171.

Bergmann,S. *et al.* (2003) Iterative signature algorithm for the analysis of large-scale gene expression data. *Phys. Rev.*, **E 67**, 10.1103/PhysRevE.67.031902.

Dempster,A.P. *et al.* (1977) Maximum likelihood for incomplete data via the EM algorithm. *J. Royal Statist. Soc. B*, **39**, 1–38.

Dittrich,M.T. *et al.* (2008) Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics*, **24**, i223–i231.

Eddy,S.R. (1998) Profile hidden Markov models. *Bioinformatics*, **14**, 755–763.

Edgar,R. *et al.* (2002) Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res*., **30**, 207–210.

Eisen,M.B. *et al.* (1998) Cluster analysis and display of genome-wide expression patterns, *Proc. Natl Acad. Sci. USA*, **95**, 14863–14868.

Friedman,N. (2004) Inferring cellular networks using Probabilistic Graphical Models. *Science*, **303**, 799–805.

Getz,G. *et al.* (2000) Coupled two-way clustering analysis of gene microarray data. *Proc. Natl. Acad. Sci. USA*, **97**, 12079–12084.

Gribskov,M. *et al*. (1987) Profile analysis: detection of distantly related proteins. *Proc Natl. Acad. Sci USA*, **84**, 4355–4358.

Grossmann,S. *et al.* (2007) Improved detection of overrepresentation of Gene-Ontology annotations with parent-child analysis. *Bioinformatics*, **23**, 3024–3031.

Grotkjær,T. *et al.* (2006) Robust multi-scale clustering of large DNA microarray datasets with the consensus algorithm. *Bioinformatics,* **22**, 58–67.

Grundy, W.N. *et al*. (1997) Meta-MEME: motif-based hidden Markov models of protein families. *Comput. Appl. Biosci.*, **13**, 397–406.

Hartwell,L.H. *et al.* (1999) From molecular to modular cell biology. *Nature*, **402**, C47–C52.

Herrero,J. and Dopazo,J. (2002) Combining hierarchical clustering and self-organizing maps for exploratory analysis of gene expression patterns. *J. Proteome Res.* **1**, 467–470.

Henikoff,S. *et al.* (1995) Automated construction and graphical presentation of protein blocks from unaligned sequences. *Gene*, **163**, GC17–GC26.

Ihmels,J. *et al.* (2002) Revealing modular organization in the yeast transcriptional network. *Nature Genet.*, 10.1038/ng941.

Ihmels,J. *et al.* (2004) Defining transcription modules using large-scale gene expression data. *Bioinformatics*, **20**, 1993–2000.

Kanehisa,M. *et al.* (2004) The KEGG resource for deciphering the genome. *Nucleic Acids Res.*, **32** (Suppl. 1), D277–D280.

Karplus,K. *et al.* (1999) Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, **14**, 846–856.

KEGG Database, Pathway dme03012. Available at http://www.genome.jp/kegg-bin/get_htext?htext=dme03012.keg&filedir=%2fkegg%2fbrite%2fdme&extend=A2&close=A2#A2

Kholodenko,B.N. *et al.* (2002) Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. *Proc. Natl Acad. Sci. USA*, **99**, 12841–12846.

Krogh,A. *et al.* (1994) Hidden Markov Models in computational biology: applications to protein modeling. *J. Mol. Biol.*, **235**, 1501–1531.

Lyne,R.R. *et al.* (2007) FlyMine: an integrated database for Drosophila and Anopheles genomics. *Genome Biol.*, **8**, R129.

Parkinson,H.M. *et al.* (2007) ArrayExpress—a public database of microarray experiments and gene expression profiles. *Nucleic Acids Res.*, **35** (Suppl. 1), D747–D750.

Parzen (1962) On the estimation of a probability density function and mode. *Ann. Math. Statist.*, **14**, 1065–1076.

Pereira-Leal,J.B. *et al.* (2004) Detection of functional modules from protein interaction networks. *Bioinformatics*, **54**, 54–57.

Petti,A.A. and Church,G.M. (2005) A network of transcriptionally coordinated functional modules in Saccharomyces cerevisiae. *Genome Res.*, **15**, 1298–1306.

Rabiner,L.R. (1989) A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. IEEE*, **77,** 257–286.

Ravasz,E. and Barabási,A.-L. (2002) Hierarchical organization in complex networks. *Phys. Rev. E*, **67**, 026122.

Ravasz,E. *et al.* (2002) Hierarchical organization of modularity in metabolic networks. *Science*, **297**, 1551–1555.

Schäfer,J. and Strimmer,K. (2005) A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Stat. Appl. Genet. Mol. Biol.*, **4**, Article 32.

Snel,B. *et al.* (2002) The identification of functional modules from the genomic association of genes. *Proc. Natl Acad. Sci. USA*, **99**, 5890–5895.

Spirin,V., and Mirny,L.A. (2003) Protein complexes and functional modules in molecular networks. *Proc. Natl Acad. Sci. USA*, **100**, 12123–12128.

Stark,C. *et al.* (2006) BioGRID: a general repository for interaction datasets. *Nucleic Acids Res*., **34** (Suppl. 1), D535–D539.

Tamayo,P. *et al.* (1999) Interpreting patterns of gene expression with self-organizing-maps: methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA*, **96**, 2907–2912.

Tanay,A. *et al.* (2004) Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proc. Natl Acad. Sci. USA*, **101**, 2981–2986.

Tavazoie *et al.* (1999) Systematic determination of genetic network architecture. *Nat. Genet.*, **22**, 281–285.

Tornow,S. and Mewes,H.W. (2003) Functional modules by relating protein interaction networks and gene expression. *Nucleic Acids Res.*, **32**, 6283–6289.

The Gene Ontology Consortium (2000) Gene Ontology: tool for the unification of biology. *Nat. Genet.*, **25**, 25–29.

Troyanskaya,O. *et al.* (2001) Missing value estimation for DNA microarrays. *Bioinformatics*, **16**, 520–525.

Venter,J.C. *et al.* (2001) The sequence of the human genome. *Science*, **291**, 1304–1351.

Wong,S.L. *et al.* (2004) Combining biological networks to predict genetic interactions. *Proc. Natl. Acad. Sci. USA*, **101**, 15682–15687.

Wu,H. *et al.* (2005) Prediction of functional modules based on comparative genome analysis and Gene Ontology application. *Nucleic Acids Res.*, **33**, 2822–2837.