A Deterministic Model to Infer Gene Networks from Microarray Data

Isabel Nepomuceno-Chamorro,
Jesús S. Aguilar–Ruiz,
Norberto Díaz–Díaz, Domingo S. Rodríguez–Baena, Jorge García

BIGS BioInformatics Group Seville
University of Seville
Pablo de Olavide University
{isabel,ndiazjgarcia}@lsi.us.es,jsagurui,dsrodbae@upo.es

Abstract. Microarray experiments help researches to construct the structure of gene regulatory networks, i.e., networks representing relationships among different genes. Filter and knowledge extraction processes are necessary in order to handle the huge amount of data produced by microarray technologies. We propose regression trees techniques as a method to identify gene networks. Regression trees are a very useful technique to estimate the numerical values for the target outputs. They are very often more precise than linear regression models because they can adjust different linear regressions to separate areas of the search space. In our approach, we generate a single regression tree for each genes from a set of genes, taking as input the remaining genes, to finally build a graph from all the relationships among output and input genes. In this paper, we will simplify the approach by setting an only seed, the gene ARN1, and building the graph around it. The final model might gives some clues to understand the dynamics, the regulation or the topology of the gene network from one (or several) seeds, since it gathers relevant genes with accurate connections. The performance of our approach is experimentally tested on the yeast Saccharomyces cerevisiae dataset (Rosetta compendium).

1 Introduction

In a microarray experiment the mRNA expression level of several thousands of genes is measured systematically, and this allows researchers to analyze data by using statistics or data mining techniques. Lately, a new approach for the interpretation of high–throughput gene expression has been proposed, *gene networks* which is based on the study of network topologies and it addresses a variety of biological systems, metabolic networks, networks of proteins, protein interactions, etc. A gene network is a direct graph, in which each node represents a gene and the relationship among different genes is represented by the edges. Several methods have been proposed to model such gene network that can be derived from gene expression data. These methods identify genes with similar expression in different situations and cluster them according to this similarity [1]. One of the differences between such models is whether they are deterministic or probabilistic. Amongst deterministic models, we consider: discrete boolean network [2], differential equations [3] as continuous method and petri nets [4] as hybrid method. The first is more computationally tractable and less accurate than continuous methods. The hybrid method can model systems with continuous flows and the discrete part models the logic functioning. Amongst probabilistic models, we consider: probabilistic boolean networks [5]; co-expression graphs [6], which are easy to interpret but they cannot distinguish direct from indirect dependencies between genes; conditional independence models as sparse Gaussian Graphic Models [7] and mutual information [8], these methods cannot estimate relationships if the number of variables is large compared to the number of samples; and finally bayesian networks [9].

In regression domains, Quinlan presented the system M5 [10]. It builds multivariate trees using linear models at the leaves. In the pruning phase for each leaf a linear model is built. Recently Witten and Frank have presented M5' in [11], a rational reconstruction of Quinlan's M5 algorithm. M5' first constructs a regression tree by recursively splitting the instance space using tests on single attributes that maximally reduce variance in the target variable. After the tree has been grown, a linear multiple regression model is built for every inner node, using the data associated with that node and all the attributes that participate in tests in the subtree rooted at that node. Then the linear regression models are simplified by dropping attributes, if this results in a lower expected error on future data (more specifically, if the decrease in the number of parameters outweighs the increase in the observed training error). After this has been done, every subtree is considered for pruning. Pruning occurs if the estimated error for the linear model at the root of a subtree is smaller or equal to the expected error for the subtree. After pruning terminates, M5' applies a smoothing process that combines the model at a leaf with the models on the path to the root to form the final model that is placed at the leaf. Also Karalic studied the influence of using linear regression in the leaves of a regression tree [12]. As in the work of Quinlan, Karalic shows that it leads to smaller models with increase of performance. Torgo has presented an experimental study about functional models for regression tree leaves [13]. Later, the same author [14] presented the system Regression Tree (RT). RT is a system which is able to use several functional models at the leaves, including partial linear models. RT builds and prunes a regular univariate tree. Then at each leaf a linear model is built using the examples that fall at this leaf. In the regression setting few works consider multi–variable splits. One of them has been presented by Li et al. in [15], where decision nodes contain linear regressions and data is split according to the sign of the residuals.

Using regression trees has the goal of showing that functional relationships can be found in the genetic network obtained from those trees. We propose to take each gene as output for the regression tree and the remaining genes as input. The output gene and its input genes define a gene–gene interaction, i.e., an edge in the graph. Therefore, the topology of the gene network is based on the accuracy of linear regressions.

The paper is organized as follows: Section 2 presents other approaches that use regression techniques to infer gene networks; Section 3 presents the approach based on regression trees; the experiments are discussed in Section 4; finally, the most interesting conclusions are summarized in Section 5.

2 Related Work

There is a wide range of approaches available to build gene network up. Several of this methods are mentioned before. The novelty of this work is using regression technique to infer gene–gene relationship. As mentioned above, our approach is based on the accuracy of linear regressions and we use parallel regressions of each gene onto the other genes by trees. This involves splitting the instances space and we use regression trees. Therefore our approach can be classified as a deterministic model.

The most significant approaches using bayesian network are [16, 17]. The latter involves learning a probabilistic model from partially observed data by using SEM (Stochastic Expectation and Maximization) algorithm. First, the algorithm is initialized by using a standard expression clustering technique to choose assignments of genes to process. Second, successive iterations of the algorithm are run to learn the graph topology and the parameters of each conditional probability distribution. Learning graph topologies is much harder than learning parameters and [16] uses regression trees to learn those graph topologies, i.e. they use this techniques for each biological process (represented by a subset of genes from the microarray) that maximizes bayesian score. However, unlike Battle, we use the regression trees to study the relationship between genes and learning gene–gene interactions from the whole microarray data.

3 Approach

The goal of our approach is to discover distinct expression patterns in which a set of genes is associated to each gene. The groups of ORFs are obtained from their prediction ability by means of regressions. We use regression trees because these representations work like several linear regressions at the same time, each of them identified by a leaf of the tree. The main advantage of this methodology is that each regression is specialized in a specific area of the search space, and hence the regression tree is generally more accurate than a global linear regression.

3.1 Building regression trees

A decision tree induction algorithm builds an initial tree without pruning. We use in this algorithm smoothed predictions to compensate sharp discontinuities that will inevitably appear between adjacent linear models. This is an usual problem for models constructed from a small number of training instances.

We use as splitting criterion of the search space the attribute of the data set that maximizes the expected error reduction. This error is called SDR for Standard Deviation Reduction and is calculated by:

$$SDR = sd(T) - \sum_i \frac{|T_i|}{|T|} \times sd(T_i)$$

where T_1, T_2,...,T_k are the sets that result from splitting the node according to the chosen attribute, and *sd* is the standard deviation.

The BMT (Build a Model Tree) and SP (SPlit node in branchs) algorithms (see Algorithms 1 and 2, respectively) present the pseudo–code for the regression tree algorithm explained above. Inputs of the BMT algorithm are a gene and a matrix that represents the microarray experiments. The output is a regression tree that represents the input.

3.2 Building the gene network

Next, we build a gene network from the regression trees, i.e., from a representation of each gene expression. The idea behind our approach is to build a regression tree for each input gene from the dataset. Every regression tree has internal genes, which divide the search space, and regression genes, which are involved in the linear regressions. They are considered as output genes of the algorithm SP. Internal genes (IG) split the search space according to their values and regression genes (RG) adjust the linear regressions in the subspace defined by internal genes. When the algorithm is run, a direct labelled graph defined from dependencies among output and input genes are provided by our approach.

A graph is defined as a tuple *(G,E)* of nodes *G* and edges *E*. An edge is an ordered pair of nodes *(gene1, gene2)* and we can interpret that the first gene or node encodes a knows transcription factor that binds to the promoter of the second gene. We build a forest of trees and we learn the gene network in terms of input–output dependency of genes between input genes and output genes (i.e. IG and RG) of each regression tree. So the relationships among output and input genes define the edges of the graph that represents the topology of the gene networks.

The algorithm BGN (see Algorithm 3) builds the forest of regression trees and the gene network step by step. The construction of gene networks from expression data is a difficult and problematic task due to the huge amount of

Algorithm 1 BMT - Build a model tree

```
INPUT matrix: a set of instances in a data set (gene expression data)
OUTPUT RT: Regression Tree
begin
  SD $\leftarrow$ Calculate the standard deviation of the class values of matrix
  RT $\leftarrow$ SP(matrix,SD)
end
```

Algorithm 2 SP - Split node in branchs

```
INPUT matrix, SD: gene expression data and standard deviation of the class values
  of a set of instances
OUTPUT RT: Regression Tree
begin
  sd ← standard deviation of the class values of matrix
  if sd < 0.05 × SD then
    type of node ← LEAF
  else if sd > 0.05 × SD then
    type of node ← INTERMEDIATE
    for all genes do
      for all possible split positions of the gene do
        Calculate the attribute's SDR
      end for
    end for
    gene ← attribute with maximum SDR
    $matrix_l$ ← conditions from the matrix to the left of the gene's split
    $matrix_r$ ← conditions from the matrix to the right of the gene's split
    gene.left ← SP($matrix_l$, SD)
    gene.right ← SP($matrix_r$, SD)
  end if
end
```

data, thousand of genes [18] and [19]. This implies deciding which genes must be included in the learning process. As an example, Peér et al. work with 565 initial genes and excludes the remaining 5751 in the yeast study.

Peña et al. [20] construct a gene network from seed genes. In this paper, we adopt the same idea, because of its simplicity, although we have increased the gene network by means of an iterative process. The algorithm BGN starts with a set S of genes, which at the beginning has only one seed. For this seed–gene, we build the regression tree and the output genes (from the regression tree) are added to the set S of genes. The genes recently added are then used as output to enlarge the levels of the graph. This is an iterative process, in which the number of iterations is calculated by experimentation.

To measure how frequent is used a local regression by experimental conditions, the *support* of a leaf is taken into account: $Support = \frac{LM*100}{N}$ where N is the number of instances in the data set and LM is the number of conditions that are classified in the specific area of the search space covered by the regression. The estimate error is defined as follows:

$$error = \frac{100 \times MSE}{global\ absolute\ deviation}$$

where MSE is the mean squared error defined by:

$$MSE = \frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{n}$$

Algorithm 3 BGN - Build Gene Network

```
INPUT  matrix: gene expression data
OUTPUT  GR, GN: Gene rank and Gene Network
begin
  GR ← ∅
  GN ← ∅
  SeedGenes ← {ARN1}
  for all attribute–gene from SeedGenes do
    RT ← BMT(matrix, attribute–gene) //build a regression tree
    GR ← update the rank of genes GR from RT
    GN ← update the gene network GN from RT
    SeedGenes ← ∅
    SeedGenes ← update the SeedGenes from RT
  end for
end
```

with p_1, p_2,. . .,p_n being the predicted values on the test instances, a_1, a_2,. . .,a_n being the real values and n is the number of attributes.

Among all the trees in the forest, we consider those whose relative error is lower than a threshold value. In a regression tree generated for an output gene, we only consider the branches of the tree whose support is greater than a threshold value, i.e., linear regressions with small number of conditions met are not accepted. In addition, we consider the genes that appear as input in a linear regression if its estimated error is smaller than a threshold value. Then the edge defined by the output and input genes is added to the graph. Otherwise, the gene is not considered in the graph, and hence it is not in the gene network either.

4 Experiments

We use the Rossetta compendium to evaluate the results obtained and to analyze if a gene network is biologically coherent. The Rosetta compendium consists of 300 full–genome expression profiles of the yeast Saccharomyces cerevisiae. Therefore, we work with a dataset with 300 examples and 6316 genes.

We use the iron homeostasis pathway in yeast (which regulate the uptake, storage, and utilization of iron so as to keep it at a non–toxic level) to evaluate the accuracy of our model. The most important genes in this process are: FRE1, FRE2, FTR1 and FET3, which control the reductive mechanism; ARN1, ARN2, ARN3 and ARN4, which control the non reductive mechanism and, finally, FIT1, FIT2 y FTI3 which are connected in the iron transport. Relevant set of genes have been taken from [21]. This set of genes has been used to measure the accuracy of bayesian gene networks in previous projects, as in [22], [23] and [20]. Specifically, those papers show models of the iron homeostasis pathway from the Rossetta compendium and are centered on ARN1 gene. Therefore, we study our approach starting at ARN1 gene in the first iteration.

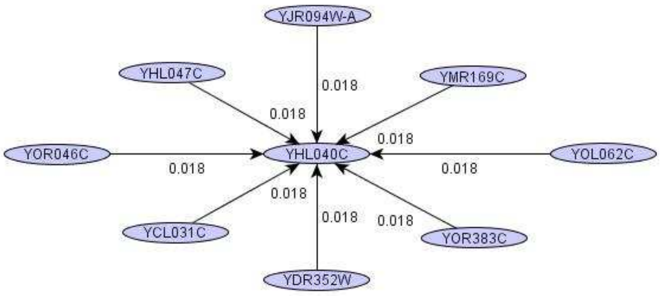

Fig. 1. An example of gene network generated using a seed gene (ARN1) and only one iteration for the algorithm. We weight the edges of the network. The weight is calculated by: $weight = \frac{1}{error \times frequency}$ where *error* is the relative absolute error and *frequency* is the number of times that a gene is in the regression tree

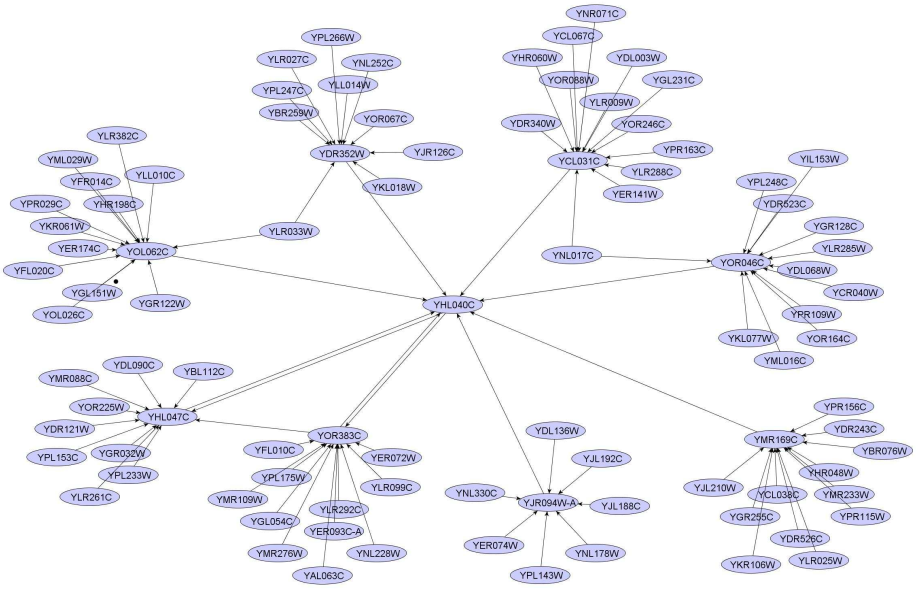

Fig. 2. Gene network for the iron homeostasis process, starting with the gene ARN1 and allowing up to 12% of the instances, i.e. each leaf of the regression trees have to cover al least 12% of the total number of instances in the dataset.

The algorithm starts using as a seed the gene ARN1 (ORF YHL040C). In order to control the accuracy of each leaf, each of them has to cover at least 33% of the total number of instances in the dataset. In the first iteration (distance 1 from the seed gene), the algorithm finds the gene FIT3 (ORF YOR383C) (see Figure 1) whereas the next gene is found in the fourth iteration. If the constraint is relaxed (allowing up to 12% of the instances), the algorithm finds in the first iteration two genes related to the iron homeostasis: FIT3 (ORF YOR383C) and ARN2 (ORF YHL047C). The gene network is depicted in Figure 2.

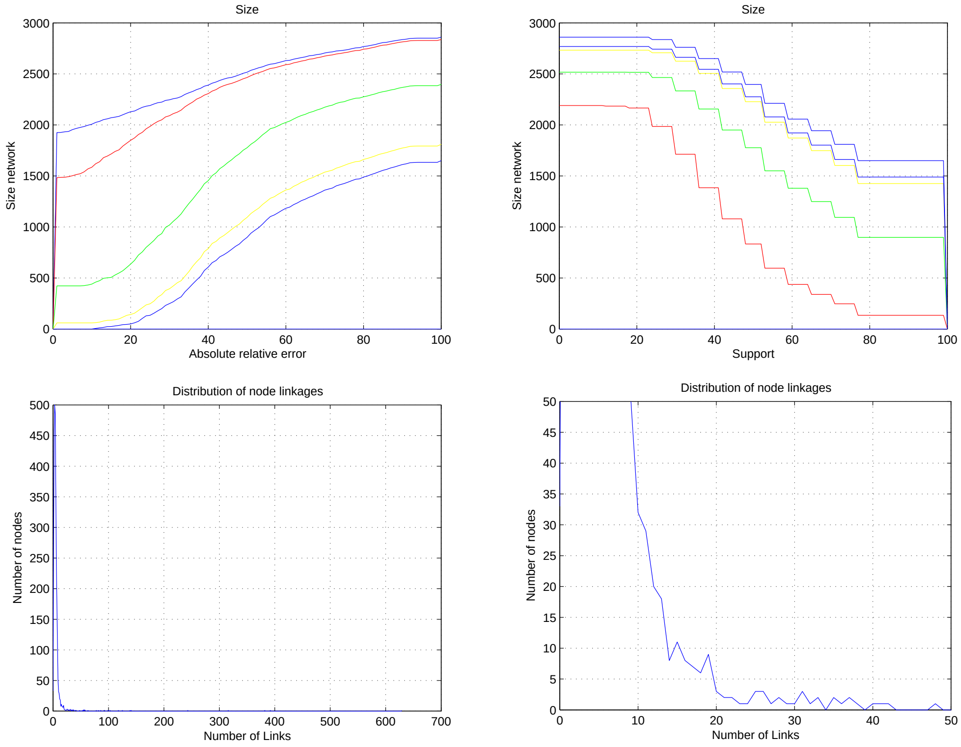

Fig. 3. The top figures show the size of the networks as a function of the relative absolute error and support and the bottom figures show the distribution of node linkages.

In the experiment above we run the algorithm from a seed gene and a gene network is built step by step from that seed gene. This is an iterative method where the number of iterations must be calculated by experimentation. Now, we study the results of the algorithm studying the whole regression trees generated for each gene. Evaluating the results is a difficult task because of the huge network generated taking into account all the genes of the dataset. Therefore the structure of the network is analyzed considering, in the task of build the network, only those regression trees whose relative absolute error is lower than a threshold value together with the branches of the trees whose support is greater than a threshold value. These two thresholds are studied by experimentation as we can see in Figure 3. The top-left figure shows the size of the network as a function of the relative absolute error. Each line is obtained by setting the value of the other threshold as 0%, 25%, 50%, 75% and 100%. We can observe that the size of the network increases slowly for an absolute relative error higher than 60. The top-right figure shows the size of the network as a function of the support.

Each line is obtained by setting the value of the error as 0%, 25%, 50%, 75% and 100%. In the bottom-left and bottom-right figures, we depict the distribution of node linkages. The bottom-right figure is another representation of the bottom-left one, in which we can better appreciate that it follows a power law in that most nodes have just a few connections and only a few have a huge number of links.

5 Conclusions

In this work we present a method to build gene networks from regression trees. The approach has several advantages: first, the gene network is based on the accuracy and support of the regression trees, which leads to solutions driven by the user; second, the gene–gene interactions are based on the local regressions, much more accurate than global regressions; third, many possible relationships are initially pruned if they do not fit the thresholds; and fourth, the depth and size of the graph is controlled by the number of iterations or the number of the studied genes, so some relationships among genes are discovered indirectly, as they might not be directly related.

The approach has an accurate performance, although we are working on designing better pruning techniques, focused on selecting only relevant genes from the regression trees to be further included in the gene network. Secondly, we are working on preparing an experimental study in which the performance of our approach is compared with other methods. And finally, we are working on determining the number of iterations and the threshold values of our approach.

Results presented in this paper, using only one gene as a seed to build the gene network are very promising, as the gene network involves genes that might have interesting biological properties for the process in study.

References

1. Tavazoie, S., Hughes, J., Campbell, M., Cho, R., Church, G.: Systematic determination of genetic network architecture. Nature Genetics **22** (1999) 281–285
2. Silvescu, A., Honavar, V.: Temporal Boolean Network Models of Genetic Networks and Their Inference from Gene Expression Time Series. Complex Systems **13** (2001) 54–70
3. Chen, K., Calzone, L., Csikasz-Nagy, A., Cross, F., Novak, B., Tyson, J.: Integrative Analysis of Cell Cycle Control in Budding Yeast. Molecular Biology of the Cell **15** (2004) 3841–3862
4. Matsuno, H., Doi, A., Nagasaki, M., Miyano, S.: Hybrid Petri net representation of gene regulatory network. Pacific Symposium on Biocomputing **5** (2000) 87
5. Shmulevich, I., Dougherty, E., Zhang, W.: Gene perturbation and intervention in probabilistic Boolean networks. Bioinformatics **18** (2002) 1319–1331
6. Stuart, J.M., Segal, E., Koller, D., Kim, S.K.: A gene-coexpression network for global discovery of conserved genetic modules. Science **302** (2003) 249–255
7. Magwene, P.M., Kim, J.: Estimating genomic coexpression networks using first-order conditional independence. Genome Biol **5** (2004)

8. Basso, K., Margolin, A., Stolovitzky, G., Klein, U., Dalla-Favera, R., Califano, A.: Reverse engineering of regulatory networks in human B cells. Nature Genetics **37** (2005) 382–390
9. Friedman, N.: Using bayesian networks to analyze expression data. Journal of Computational Biology **7** (2001) 601–620
10. Quinlan, J.: Learning with continuous classes. In: Procedings Australian Joint Conference on Artificial Intelligence. (1992) 343–348
11. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Technical report, Morgan Kaufmann, San Mateo (2000)
12. Karalic, A.: Linear regression in regression tree leaves. In: Proceedings of the ISSEK92, Bled, Slovenia. (1992)
13. Torgo, L.: Functional models for regression tree leaves. In: Proceedings of the Fourteenth International Conference on Machine Learning. (1997) 385–393
14. Torgo, L.: Partial linear trees. In: Proceedings of the 17th International Conference on Machine Learning (ICML 2000). (2000) 1007–1014
15. Li, K., Lue, H., Chen, C.: Interactive Tree-Structured Regression Via Principal Hessian Directions. Journal of the American Statistical Association **95** (2000)
16. Battle, A., Segal, E., Koller, D.: Probabilistic discovery of overlapping cell processes and their regulation. In: Eight Annual International Conference on Research in Computational Molecular Biology (RECOMB), San Diego, CA (2004)
17. Segal, E., Shapira, M., Regev, A., Pe'er, D., Botstein, D., Koller, D.: Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. Nature Genetics **34** (2003) 166–176
18. Hughes, T., Marton, M., Jones, A., Roberts, C., Stoughton, R., Armour, C., Bennett, H., Coffey, E., Dai, H., He, Y.: Functional Discovery via a Compendium of Expression Profiles. Cell **102** (2002) 109–126
19. Fink, G., Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., Futcher, B.: Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces cerevisiae by Microarray Hybridization. Molecular Biology of the Cell **9** (1998) 3273–3297
20. Peña, J., Bjorkegren, J., Tegner, J.: Discovering statistically significant biclusters in gene expression data. Bioinformatics **In press** (2005)
21. Lesuisse, E., Blaiseau, P., Dancis, A., Camadro, J.: Siderophore uptake and use by the yeast Saccharomyces cerevisiae (2001)
22. Margolin, A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla Favera, R., Califano, A.: Aracne:an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics **7** (2006) S7
23. Peer, D., Regev, A., Elidan, G., Friedman, N.: Inferring Subnetworks from Perturbed Expression Profiles. Bioinformatics **17** (2002) 215–224