Gene Regulatory Network Reverse Engineering using Population Based Incremental Learning and K-means

Leon Palafox*
The University of Tokyo
7-3-1, Hongo, Bunkyo
Tokyo, Japan
leon@iba.t.u-tokyo.ac.jp

Hitoshi Iba†
The University of Tokyo
7-3-1, Hongo, Bunkyo
Tokyo, Japan
iba@iba.t.u-tokyo.ac.jp

ABSTRACT

Finding interactions among genes is one of the main problems in molecular biology. In this paper, we use a novel approach to model the gene's regulations, or Gene Regulatory Networks (GRNs). We use a Recursive Neural Network (RNN) to model the networks, and then use Population Based Incremental Learning (PBIL) enhanced with K-means to find the optimum parameters of the Neural Network. We present experiments with real data, we compare our algorithm with others approaches by calculating different statistics for the solutions.

Categories and Subject Descriptors

J.3 [**Life and Medical Sciences**]: Biology and genetics; G.1.6 [**Optimization**]: Global Optimization

Keywords

Gene Regulation Network, Estimation of Distribution, Population Based Incremental Learning, Bioinformatics

1. INTRODUCTION

Inferring the interactions between genes is an important problem in Molecular Biology, these relations contain information about genetic diseases and processes inside organisms

Among the different models for Gene Regulatory Networks (GRNs), we can find Bayesian Networks[4] and dynamic systems. The latter use a set of ODEs to model the networks; this set, when solved, express the data from the real experiments. The right set of ODEs's parameters will reconstruct the network's time data. Inference using ODEs, however, is slow, since for each candidate solution, it calculates the solution of N differential equations, where N is the number of genes [3].

In this paper, we use a Recursive Neural Network (RNN)[6] to model the interdependencies of the GRNs and optimize the RNN using Population Based Incremental Learning (PBIL). Since it models all the solutions as unimodal distributions, classic PBIL, is a bad fit for problems with local minima. We extend the algorithm using a clustered PBIL. The choice of the number of clusters, however, is different for each problem. In this work we analyze these properties for a real network using a K-means clustering.

2. MODELING GRN USING RNN

Since the inputs for a classic Neural Network are taken iid from the training set, NNs are not suited to model temporal data; RNN, however, is a closed loop NN useful to model dynamic systems.

Vohradský [6] modeled the gene's regulations with this architecture, assuming that each of the neurons in the output unit is a gene.

The equation of an RNN resembles a standard NN, but with additional variables for the feedback loop. The discretized version is:

$$e_i(t+\Delta t) = \frac{\Delta t}{\tau_i} \times f\left(\sum_{j=1}^{N} w_{ij} e_j(t) + \beta_i\right) + \left(1 - \frac{\Delta t}{\tau_i}\right) e_i(t) \quad (1)$$

where, $f()$ is a nonlinear function that acts as a classification function, we use the sigmoid function $f(z) = 1/(1+e^{-z})$. The values w_{ij} are the connecting weights of the network, which represent the connections between gene i and j. The variable e_j represents the expression level for the gene, which is the data we receive from the microarray experiments. And finally, β is the bias parameter of the network.

2.1 Training of RNN models

Xu et al [7] used Particle Swarm Optimization (PSO) to optimize the RNN weights, and it has been shown that evolutionary techniques perform on par with traditional gradient descent and other classic optimization algorithms in NN.

In this paper, we use PBIL to find the weights of the RNN. PBIL, unlike PSO, can find a global minimum in a non-convex set and it only has to update an inferred probability distribution. We improve PBIL using a mixture of distributions, instead of a unimodal distribution to model the candidates. There are many choices to infer a mixture of distributions form sampled data, we use K-means.

*Department of Electric Engineering.

†Department of Information Science and Technology.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

Table 1: Specificity, Sensitivity and F-Score for different models of the *E.coli* SOS Network

	#Regs	#TP	#FP	#TN	#FN	Sensitivity	Specificity	F-Score	Time[h]
Bayesian Network[4]	6	4	2	18	3	0.571	0.900	0.615	0.01
S-Tree[1]	7	6	1	19	1	0.857	0.950	0.857	35
LTV[2]	13	7	6	14	0	1.000	0.700	0.7	0.1
DE[3]	8	5	3	17	2	0.714	0.850	0.667	0.3
ClusteredIPBL	11	7	4	13	3	0.7	0.765	0.67	0.05

3. PBIL AND CLUSTERING

Population Based Incremental Learning (PBIL)[5] finds the best candidates of a function by inferring a probability distribution from each of the dimensions of the best candidate's population. This creates N distributions, where N is the dimension of the problem. The algorithm uses a fitness function to score each of the candidates, and then sets a threshold that selects only the best candidates to infer a new distribution.

This approach, however, has setbacks, like assuming that each dimension is iid sampled, which is a naive approach. Also, since the search space is non-convex, using standard Gaussians distributions to model the problem is unrepresentative for multimodal settings. Using a mixture of Gaussians as a surrogate for a multimodial distribution has presented promising results for settings with local minima.

It can be proved that K-means can be used as relaxation of a mixture of Gaussian distributions with symmetric variances. We model the candidates as a mixture of K Gaussian distributions, to have at a set of $N \times K$ clusters modeling the best candidates of the problem for each of the N dimensions.

4. ALGORITHM TO INFER THE GRN

PBIL, for Regulatory Network inference, uses the following fitness function:

$$F(Cand_i) = \frac{1}{TN} \sum_{t=0}^{T} \sum_{i=0}^{GeneNo} (e_i(t) - Cand_i(t))^2 \quad (2)$$

where T is the time samples we used for each time series and $GeneNo$ is the number of genes in the net. The values $e_i(t)$ and $Cand_i(t)$ are the values for the experimental data and the RNN evaluation of the candidate w_{ij}, respectively.

The algorithm first creates a set of random entries, using the fitness function it selects the best scores, and calculates a new probability distribution using the K-means approach.

5. EXPERIMENTS & RESULTS

We tested the algorithm with the SOS network for *E.coli* and counted the total number of true positives and true negatives to calculate variables like Recall, Precision and F-Score. To test the effect of the variables in the PBIL, we changed the population size, as well as the cluster number in the K-means. We did 2000 iterations per run, with each iteration lasting at most 5 minutes for the architectures with 500 candidates. To compare our results, we compiled results from other papers working with the SOS Net. Table 1 shows this compilation.

Our approach is in line with the best results when comparing the F-Score, only the S-Tree approach and the linear time-variant model has better results. Both algorithms, however, take longer to do the inference.

Furthermore, we found the true positives for at least 80% of the trials with a low standard deviation. Only one other approach was capable of finding the seven correct regulations[2], and its inference time was higher that ours.

6. CONCLUSIONS

We have presented the use of a Recursive Neural Network to model the Gene Regulations in an artificial and a real system. We used a clustered version of PBIL to find the optimal weights in the RNN. We compared the real SOS Network inference with similar works in the area, and found that the algorithms behaves on par with some of the best implementations.

However, the hard clustered approach is a naive attempt to model the dynamics of the population. We saw how, for different population sizes, the optimal number of clusters was different. We will implement other approaches for dynamic clustering like the novel Dirichlet Processes, which generate the necessary number of clusters for each population.

7. REFERENCES

[1] D.-Y. Cho, K.-H. Cho, and B.-T. Zhang. Identification of biochemical networks by S-tree based genetic programming. *Bioinformatics*, 22(13):1631–1640, 2006.

[2] M. Kabir, N. Noman, and H. Iba. Reverse engineering gene regulatory network from microarray data using linear time-variant model. *BMC bioinformatics*, 11 Suppl 1:S56, Jan. 2010.

[3] N. Noman and H. Iba. Inferring gene regulatory networks using differential evolution with local search heuristics. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 4(4):634–47, 2007.

[4] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. D'Alche-Buc. Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19(Suppl 2):ii138–ii148, Oct. 2003.

[5] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *Parallel Problem Solving from Nature*, pages 418–427. Springer, 1998.

[6] J. Vohradský. Neural network model of gene expression. *The FASEB journal : official publication of the Federation of American Societies for Experimental Biology*, 15(3):846–54, Mar. 2001.

[7] R. Xu, I. I. Donald Wunsch, and R. Frank. Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 681–692, 2007.