

Online Supplement

Boolean Dynamics of Genetic Regulatory Networks Inferred from Microarray Time Series Data

Shawn Martin¹, Zhaoduo Zhang², Anthony Martino³ and Jean-Loup Faulon^{4,*}

¹Sandia National Laboratories, Computational Biology Dept., P.O. Box 5800, Albuquerque, NM, 87185-1316, USA; ²Sandia National Laboratories, Biosystems Research, P.O. Box 969, Livermore, CA 94551-9291, USA; ³Sandia National Laboratories, Biomolecular Analysis and Imaging, P.O. Box 5800, Albuquerque, NM 87185-0895, USA; ⁴Sandia National Laboratories, Computational Biosciences Dept., P.O. Box 5800, Albuquerque, NM 87185-1413, USA.

ABSTRACT

Summary: This online supplement contains pseudo-code and additional explanation for the network inference algorithms, as well as a comparison of the effect of different clustering algorithms on meta-gene discretization.

Availability & Contact: For questions and code, please contact Jean-Loup Faulon at jfaulon@sandia.gov.

1 NETWORK INFERENCE PSEUDO-CODE

In this section we describe in detail the algorithms used to infer networks from gene expression profiles. These algorithms count, sample, enumerate, identify attractors, and simulate the dynamics of all the possible networks matching a given set of discretized expression profiles. Networks are counted, sampled, and enumerated at the node (meta-gene) level. For every node we determine all the possible sets of nodes that might control the expression profile of that node. Expression profiles are given over time and the algorithms can accept several time courses corresponding to different initial conditions. These initial conditions can be different stimuli, or various knockouts experiments.

The algorithms take as input a set of n nodes $\{v_1, v_2, \dots, v_n\}$ which correspond to the meta-genes, and a set c profiles $\{P^1, P^2, \dots, P^c\}$, where each profile is a binary matrix of gene expression for a given initial condition. P^l is an $n \times m_l$ matrix with entry $P^l_{i,j} = 1$ if gene i is up-regulated at time j , or $P^l_{i,j} = 0$ if gene i is down-regulated at time j . Typically each profile matrix P^l has the same number of time points m (so that $m_l = m$ for all l) but we can allow different profiles to have different numbers of time points. Further, different nodes typically have different time series expressions, although this requirement is also not necessary to run the algorithms.

The fundamental code used in all the algorithms is `INFER_FUNCTION`. This routine determines if a set of nodes v_1, v_2, \dots, v_q with $q \leq n$ can explain the expression profile of a given node v_i . In other words, `INFER_FUNCTION` returns the Boolean function by which v_1, v_2, \dots, v_q control the expression of v_i . This function is empty if v_1, v_2, \dots, v_q do not control v_i . A further check is performed to determine whether or not the function is an activation-inhibition function of the type $v_i(t) = (v_1(t) \text{ OR } v_2(t) \text{ OR } \dots) \text{ AND NOT } (v_j(t) \text{ OR } v_{j+1}(t) \text{ OR } \dots)$, where $v_1(t), v_2(t), \dots$ are activators and $v_j(t), v_{j+1}(t), \dots$ are inhibitors.

```

INFER_FUNCTION ( $P^1, \dots, P^c, v_i, v_1, \dots, v_q$ )
define  $f = * * \dots *$  (string of  $2^k$  characters)
for each profile  $P^l$  in  $\{P^1, \dots, P^c\}$  do
  for each time  $t$  in profile  $P^l$  do
     $input = P^l_{1,t}P^l_{2,t} \dots P^l_{q,t}$ 
    ( $input$  is an integer in binary format)
    if  $f(input) = *$ 
      then  $f(input) = P^l_{i,t+1}$ 
    else if  $f(input) \neq P^l_{i,t+1}$  return ( $\emptyset$ )
  done
done
if  $f$  is activation-inhibition return ( $f$ )
else return ( $\emptyset$ )

```

As an example of `INFER_FUNCTION`, suppose we are given a time series with 6 time points for 3 genes v_1, v_2 , and v_3 , as shown in Table 1, where we write 1 when the gene is up-regulated and 0 when down-regulated.

		Time					
		1	2	3	4	5	6
Gene	v_1	0	0	1	1	0	0
	v_2	1	0	0	1	1	0
		1	0	0	1	0	0

Table 1. An example of discrete time course with 3 genes and 6 time points. Zero (red) denotes down-regulation and 1 (green) denotes up-regulation.

We are interested in determining an activation-inhibition function for v_3 assuming that v_1 and v_2 are potential inputs. Can we explain v_3 expression profile using v_1 as the only input? Consider the relevant information, as extracted in Table 2.

		Time					
		1	2	3	4	5	6
Gene	v_1	red	red	green	green	red	
	v_2			green	green	green	green
		green	green	red	red	green	red
		green	green	red	red	green	red

Table 2. The effect of the gene expression of v_1 on the gene expression of v_3 at the next time step.

In Table 2 we see that v_3 is down-regulated when v_1 is down-regulated at the previous time step. However, v_3 can be either up- or down-regulated when v_1 is up-regulated at the previous time

*To whom correspondence should be addressed.

step. Thus v_1 cannot explain v_3 . The same is true when v_2 is considered as input for v_3 . What if both v_1 and v_2 are considered as inputs of v_3 ? In this case we have the situation shown in Table 3.

		Time					
		1	2	3	4	5	6
Gene	v_1	Red	Red	Green	Green	Red	
	v_2	Green	Red	Red	Green	Green	
		↓	↓	↓	↓	↓	
	v_3	Red	Red	Red	Red	Red	

Table 3. The effect of both v_1 and v_2 on v_3 .

From Table 3 we see that v_3 is down-regulated when v_2 is up-regulated at the previous time step (v_2 inhibits v_3), and that v_3 is up-regulated when v_1 is up-regulated at the previous time step (v_1 activates v_3). In the parlance of the pseudo-code for `INFER_FUNCTION` we have a Boolean function f such that $f(00) = 0$, $f(01) = 0$, $f(10) = 1$, and $f(11) = 0$. Or, written in a more conventional notation we have $v_3(t+1) = v_1(t) \text{ AND NOT } v_2(t)$.

Using `INFER_FUNCTION`, inferring networks can easily be done by processing each node in order. The code for performing this inference is called `INFER_NETWORKS`. `INFER_NETWORKS` takes as input the profiles P^1, P^2, \dots, P^c , the nodes v_1, v_2, \dots, v_n ; and the maximum number $Q \leq n$ of connections allowed for a given node. `INFER_NETWORKS` returns as output the possible connections and associated Boolean functions for each node v_i . This information is stored in a set `NETWORKS(v_i)`. This set is composed of a sequence of $(q+1)$ -tuples, where q is variable but $q \leq Q$. Each tuple contains the list of q inputs as well as the associated Boolean function returned by `INFER_FUNCTION`.

```
INFER_NETWORKS ( $P^1, P^2, \dots, P^c, v_1, v_2, \dots, v_n, Q$ )
for each node  $v_i$  do
  NETWORKS( $v_i$ ) =  $\emptyset$ 
  for  $k = 1$  to  $Q$  do
    for all  $k$ -tuples  $v_1, \dots, v_k$  do
       $f = \text{INFER\_FUNCTION}(P^1, P^2, \dots, P^c, v_i, v_1, \dots, v_k)$ 
      if  $f \neq \emptyset$  then
        add  $(v_1, \dots, v_k, f)$  to NETWORKS( $v_i$ )
    done
  done
done
return (NETWORKS)
```

Using `INFER_NETWORKS` it is quite easy to count the number of possible networks matching a given set of expression profiles. Note that the number of possible networks is simply the product of the number of possible inputs for each node.

```
COUNT_NETWORKS ( $v_1, v_2, \dots, v_n, \text{NETWORKS}$ )
 $P = 1$ 
for each node  $v_i$  do
   $P = P * |\text{NETWORKS}( $v_i$ )|$ 
done
return ( $P$ )
```

Next, we can select at random possible networks for the set `NETWORKS` using the `SAMPLE_NETWORKS`.

```
SAMPLE_NETWORKS ( $v_1, v_2, \dots, v_n, \text{NETWORKS}$ )
for each node  $v_i$  do
  select at random  $f$  from NETWORKS( $v_i$ )
  print  $v_i, f$ 
done
```

Now, to enumerate all networks, we run `INFER_NETWORKS` then list and print all possible inputs for each node. Recall that `INFER_NETWORKS` generates for each node a list of possible inputs and associated Boolean functions. In order to enumerate all possible networks it is enough to simply enumerate all possible inputs in order. This necessitates maintaining a variable for each node called $I\#(v_i)$, which is the current input number for node v_i . The enumeration algorithm essentially prints the input of each node for all possible values of the vector $I\#$, and thus reduces to an enumeration of all possible $I\#$ vectors.

```
ENUMERATE_NETWORKS ( $v_1, v_2, \dots, v_n, \text{NETWORKS}$ )
set  $I\#(v_i) = 1$  for all nodes  $v_i$ 
while  $I\# \leq (|\text{NETWORKS}( $v_1$ )|, \dots, |\text{NETWORKS}( $v_n$ )|)$  do
  for each node  $v_i$  do
     $f = I\#(v_i)$  element of NETWORKS( $v_i$ )
    print  $v_i, f$ 
  done
  next ( $I\#$ )
done
```

Using `SAMPLE_NETWORKS` or `ENUMERATE_NETWORKS`, we are now in a position to consider the dynamics of the different possible networks. These dynamics are analyzed by running the networks and analyzing the attractors.

We first generate a gene expression profile P^* by running a given network `NETWORK` taken from the set `NETWORKS` inferred using the `SAMPLE_NETWORKS` or `ENUMERATE_NETWORKS` routines. The `RUN_NETWORK` routine takes one of the initial condition profiles P^t from P^1, P^2, \dots, P^c ; the nodes v_1, v_2, \dots, v_n ; a specific network `NETWORK`; and a maximum time T for the output profile P^* .

```
RUN_NETWORK ( $P^1, v_1, v_2, \dots, v_n, \text{NETWORK}, T$ )
 $t_1 = \text{second time step (first is } t_0)$ 
for  $t = t_1$  to  $T$  do
  for each node  $v_i$  do
    let  $v_1, \dots, v_k, f$  be the input nodes and
      associated Boolean function for node
       $v_i$  as recorded in NETWORK( $v_i$ )
     $P^*_{i,t} = f(P^*_{v1,t-1}, \dots, P^*_{vk,t-1})$ 
  done
done
for each  $v_i$  do
  print  $P^*_{i,t_0}, \dots, P^*_{i,T}$ 
done
```

Using `RUN_NETWORK`, we can now analyze the attractors of the network. An attractor is a cyclic pattern of expression that any network will eventually exhibit due to the finite nature of Boolean networks. Assuming a profile P^* is given up to a predefined time T , the following routine will return the time step t_1 at which an attractor is found. The time step t_1 is the first time step such that the expression profile of the nodes at time t_1 is the same as the expression profile of the nodes at time T .

```

ATTRACTOR ( $P^*, v_1, v_2, \dots, v_n, T$ )
for  $t_1 = 0$  to  $T-1$  do (loop 1)
  for each node  $v_i$  do (loop 2)
    if  $P^*_{i,t_1} \neq P^*_{i,T}$  then
      break (loop 1)
  done
  if  $P^*_{:,t_1} = P^*_{:,T}$  then
    break (loop 2)
  (note  $P^*_{:,t_1}$  is the  $t_1$  column of  $P^j$ )
done
if  $P^*_{:,t_1} = P^*_{:,T}$ 
  then return( $t_1$ )
  else return( $\infty$ )

```

2 ALTERNATE CLUSTERING ALGORITHMS

Going from a full set of microarray to a reduced set of discrete meta-genes is sure to involve some loss of information and/or introduction of error. In our case, the most likely source of error is the clustering step. Not only did we choose k -means from a host of algorithms, we also chose a value for k , and a random starting condition for k -means. To examine the effect of these choices, we compared k -means with itself using different initial starting conditions, with SOMs (Tamayo, Slonim et al. 1999), and with hierarchical clustering (Eisen, Spellman et al. 1998). We made these comparisons by repeating our entire discretization procedure using k -means, SOMs, and hierarchical clustering, each using a range of values for k , and (in the case of k -means) a number of random starting conditions. We then discretized the resulting meta-genes using SVR as described in the manuscript for each algorithm, each k , and each random starting condition. We compared the different discretizations using a simple measure of set agreement.

Our measure is computed by considering a discretization to be a set of discrete time courses, where each time course is a vector, so that a discretization is a set of vectors. If we have two such sets A and B , then we can compute their similarity by computing

$$\frac{A \cdot B}{\sqrt{|A||B|}}, \text{ where } A \cdot B = \sum_i \max_j \frac{a_i \cdot b_j}{\sqrt{\|a_i\| \|b_j\|}}, |A| \text{ denotes the}$$

cardinality of A , and we assume that $|A| \leq |B|$. We note that this measure is between 0 and 1 (inclusive), and is 1 if and only if $A = B$.

We used the IL-2 stimulated T cell immune response dataset to examine the robustness of our discretization relative to different clustering algorithms and different random starting conditions for k -means. We compared k -means with SOMs and hierarchical clustering. Each of these algorithms can be used to partition a dataset, but each requires different input and provides different results. For k -means, we provide the number k of clusters that will be produced as well as the initial (random) locations of the cluster centers. In this experiment, we let k vary from 2 to 40 and we re-started the algorithm (with random initial conditions) 25 times for each value of k .

A SOM provides both a partition of a dataset and a visualization of that dataset in two or three dimensions. Both the partition and visualization are provided by specifying a topology in advance. This topology is given by, for example, a $w \times h$ matrix of cells. The SOM is then a map for the original input space to the $w \times h$ grid. This map is computed to preserve both similarity between both data points mapped into the cells and similarity between data in nearby cells. An SOM computed using a 5×5 rectangular grid on the IL-2 dataset is shown in Figure 1. The data points within

the cells are considered to be clusters and therefore partition the dataset. In our case we used SOMs generated by GeneCluster (Tamayo, Slonim et al. 1999) with rectangular grid topologies of size $w \times h$, where $w \leq 2h$ and $2 \leq w \times h \leq 40$.

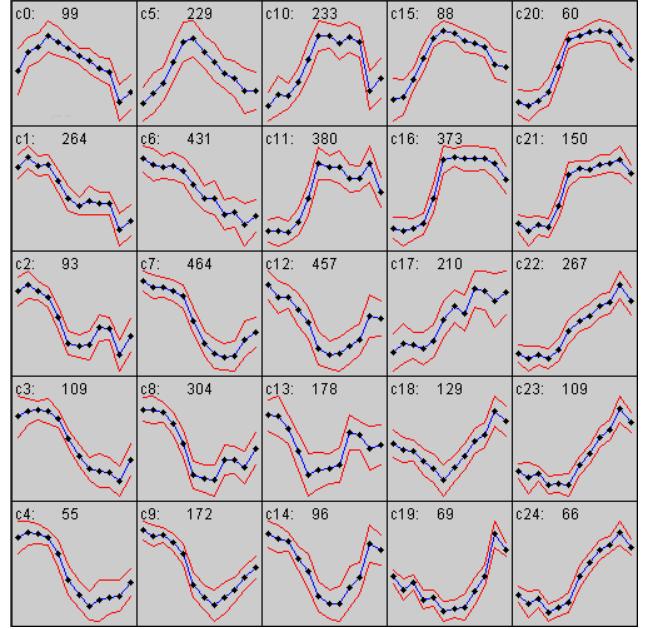


Figure 1. A SOM generated by GeneCluster on a 5×5 rectangular grid using the IL-2 microarray data. Each cell of the grid contains a cluster whose average profile is given by the blue line, with standard deviation given by the red lines. The SOM organizes the cells so that adjacent cells have similar cluster profiles.

Hierarchical clustering produces a dendrogram tree as shown in Figure 2. Each leaf in this tree corresponds to a gene and gene clusters are given by subtrees, so that larger clusters are obtained as we traverse towards the root of the tree. This tree can be used to partition a dataset by partitioning the tree into a set of subtrees below a given vertical cutoff. We used Euclidean distance and complete linkage to obtain the dendrogram and partitioned the tree using different cutoffs to obtain partitions with 2 to 40 clusters.

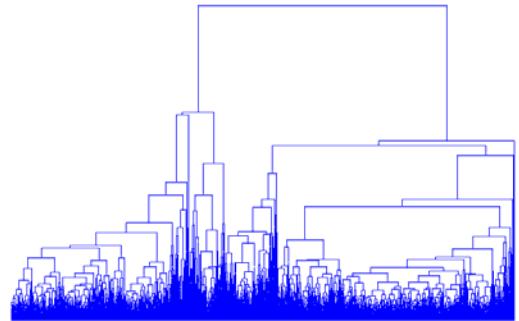


Figure 2. A dendrogram tree generated by hierarchical clustering using Euclidean distance and complete linkage with the IL-2 microarray data. The leaves correspond to genes and the subtrees correspond to gene clusters (meta-genes).

The above clustering methods yielded 25×39 (random iterations \times values of k) different partitions of the IL-2 dataset using k -means, 1×18 partitions using SOMs, and 1×39 partitions using hierarchical clustering, all with number of clusters k between 2 and 40. (For SOMs there were 18 values of $k = w \times h$ using our criterion $w \leq 2h$ and $2 \leq w \times h \leq 40$.) These partitions were compared using our similarity measure described previously. In particular, we computed the average similarity measure for each of the algorithms (relative to k -means) for each value of k . The resulting curves are shown in Figure 3.

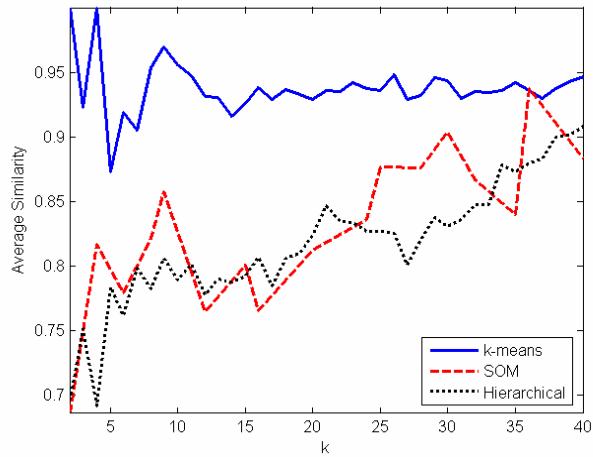


Figure 3. Average similarity values versus k for k -means with itself using 25 random re-starts, SOM with k -means, and hierarchical clustering with k -means.

The curves in Figure 3 show that that k -means is stable with regard to random initial conditions for k above 20 and that SOM and hierarchical clustering give results very similar to k -means as k increases. Hence the random initial starting conditions and choice of clustering algorithm is somewhat arbitrary for our chosen value of $k = 23$ in the case of the IL-2 dataset. At $k = 23$ we had >80% agreement between discretizations in all clustering algorithms.

These curves may also indicate that we could have selected an even larger value for k (e.g. 40). We chose $k = 23$ as a compromise between increased agreement between the clustering algorithms and increased computational complexity associated with having additional meta-genes.

REFERENCES

Eisen, M. B., P. T. Spellman, et al. (1998). "Cluster analysis and display of genome-wide expression patterns." *Proc Natl Acad Sci U S A* **95**(25): 14863-8.

Tamayo, P., D. Slonim, et al. (1999). "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation." *Proc Natl Acad Sci U S A* **96**(6): 2907-12.