RegnANN: Reverse Engineering Gene Networks Using Artificial Neural Networks

Marco Grimaldi*, Roberto Visintainer, Giuseppe Jurman

Fondazione Bruno Kessler, Trento, Italy

Abstract

RegnANN is a novel method for reverse engineering gene networks based on an ensemble of multilayer perceptrons. The algorithm builds a regressor for each gene in the network, estimating its *neighborhood* independently. The overall network is obtained by joining all the neighborhoods. RegnANN makes no assumptions about the nature of the relationships between the variables, potentially capturing high-order and non linear dependencies between expression patterns. The evaluation focuses on synthetic data mimicking plausible submodules of larger networks and on biological data consisting of submodules of *Escherichia coli*. We consider Barabasi and Erdös-Rényi topologies together with two methods for data generation. We verify the effect of factors such as network size and amount of data to the accuracy of the inference algorithm. The accuracy scores obtained with RegnANN is methodically compared with the performance of three reference algorithms: ARACNE, CLR and KELLER. Our evaluation indicates that RegnANN compares favorably with the inference methods tested. The robustness of RegnANN, its ability to discover second order correlations and the agreement between results obtained with this new methods on both synthetic and biological data are promising and they stimulate its application to a wider range of problems.

Citation: Grimaldi M, Visintainer R, Jurman G (2011) RegnANN: Reverse Engineering Gene Networks Using Artificial Neural Networks. PLoS ONE 6(12): e28646. doi:10.1371/journal.pone.0028646

Editor: Paolo Provero, University of Turin, Italy

Received August 19, 2011; **Accepted** November 11, 2011; **Published** December 28, 2011

Funding: The authors acknowledge funding by the European Union FP7 Project HiperDART, by the Italian Ministry of Health Project ISITAD (RF2007conv. 42), and by the PAT funded Project ENVIROCHANGE. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: marco.grimaldi@gmail.com

Introduction

The task of gene regulatory network (GRN) inference is a daunting task not only in terms of devising an effective algorithm, but also in terms of quantitatively interpreting the obtained results [1]. Only recently efforts have been carried out towards an objective comparison of network inference methods also highlighting occurring limitations, e.g.: [2–5]. As an example, the Dialogue for Reverse Engineering Assessments and Methods (DREAM) challenge is one of the prominent efforts that aims at evaluating the success of GRN inference algorithms on synthetic benchmarks data sets.

Early network inference models were based on the analysis of the correlation coefficients [6,7] between expression patterns of all pairs of genes to infer *co-expression* networks. On the basis that correlation coefficients fail to capture more complex statistical dependencies among expression patterns (e.g. non-linearity), more recently general methods based on measures of dependency such as mutual information have been proposed [7]. Of this class of algorithms, ARACNE [8] and CLR [9] have been adopted to address a wide range of network deconvolution problems - from transcriptional [8] to metabolic networks [10] - and they are often used as reference benchmark algorithms (e.g.: [4,7,11,12]). Generally, the inference methods proposed are of very different nature, ranging from deterministic (systems of differential equations [13] and Gröbner bases [14]) to stochastic approaches, e.g.: Boolean [15] or Bayesian [16] algorithms. Such approaches may also start from different types of gene expression data: time-course or steady states. Furthermore, the detail and the complexity of the considered network can also vary: links may carry information about the direction of the relation (directed graph) and a weight may be associated to the strength of each link (weighted graph) [17,18]. Generally, the reconstruction accuracy is far from being optimal due to drawbacks related to both the methods and the available data [19].

One of the aspects that makes network inference a daunting task is its intrinsic underdetermination [2] related to the size of the search space. It is often the case that the expression profiles of thousands of genes (e.g. approximately 4500 in the case of *Escherichia coli*), controlled by hundreds of regulators (e.g. approximately 300 known transcription factors in *Escherichia coli*), are recorded for a limited amount of experimental conditions - about 450 for the last publicly available *Escherichia coli* gene expression data set. Thus, considering also possible combinatorial regulations and feedback loops, the number of possible solutions of the inference problem becomes prohibitively large compared to the available experimental measurements at hand.

In this work we introduce a novel inference method called Reverse Engineering Gene Networks with Artificial Neural Networks (RegnANN). This inference algorithm, trained using steady state data as provided by microarray data, builds a multivariable regressor (one to many) for each gene in the network. The algorithm is based on an ensemble of Multilayer Perceptrons (MLPs) trained using steady state data. RegnANN estimates the *neighborhood* of each gene (the correlations among one gene and all the others) independently and then it joins these neighborhoods to form the overall network. RegnANN performance is compared with those of top-scoring methods such as KELLER [20], ARACNE [8] and CLR [9]. To improve the general efficiency of RegnANN we implement the algorithm using the GPGPU programming paradigm [21]. The main feature of the novel method presented is that it makes no assumptions about the nature of the relationships between the variables, potentially capturing high-order and non linear dependencies between expression patterns. On the other hand, RegnANN differs greatly from other published methods based on ANN or simple binary perceptron (e.g.: [22–26]): it is a multi-variable regressor (one input variable, many output variables) trained using steady states data for determining gene interactions.

In evaluating the performance of the four different network inference methods, first we settle in a controlled situation with synthetic data and then we focus on a biological setup by analyzing transcriptional subnetworks of *Escherichia coli*.

The general performance of the network inference task is evaluated in terms of Matthews Correlation Coefficient - MCC [27]. MCC is becoming the accuracy measure of choice in many application fields of machine learning and bioinformatics: it is one of the best methods for summarizing into a single value the confusion matrix of a binary classification task, recently adopted also for network topology comparison [28].

The experimental evaluation firstly verifies RegnANN ability of inferring direct and indirect interactions among genes and possible cooperative interaction between putative regulators on a set of toy experiments. Considering only the underlying topology (e.g.: undirected unweighted graph), we then focus on synthetic data mimicking plausible submodules of larger networks generated according to both Barabasi [29] and Erdös-Rényi [30] models. In doing so, we focus on a scenario of reduced search space/extended amount of independent information [2]. To tackle various aspects of the problem of network inference, we analyze the effect of, e.g.: increasing the amount of available data while varying the topology of the network, the number of nodes in the network, the data synthesis method and the inference algorithm applied.

We finally demonstrate our approach on a biological data set consisting of a selected number of subnetworks of *Escherichia coli* including a number of genes ranging from 7 to 104. The expression data consists of 445 microarray expression profiles collected under different experimental conditions.

Results

In order to present coherently the results obtained on synthetic data, we start firstly with an evaluation of RegnANN ability of inferring direct and indirect interactions among genes and possible cooperative interaction between putative regulators (e.g.: transcription factors). This is done on four toy examples considering interaction among four genes.

The second phase of our analysis focuses on the effect of varying the amount of available data in the task of network inference while considering a fixed threshold for the binarization of the inferred adjacency matrix. The performance in terms of MCC obtained with RegnANN is systematically compared with the ones obtained by KELLER. The accuracy of each inference method is firstly evaluated on synthetic data by varying the topology of the network, the amount of data available and the method adopted to synthesize the data. Once the topology of the network is (randomly) selected, the desired amount of data is synthesized according to the generation method of choice, the network inference methods are applied and the MCC score calculated. We consider discrete (in $\{0, 1\}$) symmetric adjacency matrices for fair comparison between the two methods as KELLER does not infer coupling direction, nor the strength of the interaction. To account for intrinsic instability of each inference method, data generation and network inference are repeated 10 times for each given network topology and the MCC score is estimated as the mean of the 10 independent runs. The error of the measurement is expressed as twice the standard deviation. In order to generalize from the selected network topology, the entire procedure is repeated 10 times and the final accuracy score (MCC) is calculated as the mean accuracy for each run. Similarly, the error of the accuracy score is estimated as the mean of the error for each run. Our analysis explores the effect of varying the mean degree of the nodes in the case of Erdös-Rényi networks and the exponent of the power-law for Barabasi networks in network inference. We also test the effect of different data normalization procedures on the accuracy of the two network induction methods considered.

The third phase of the experimental evaluation on synthetic data compares the performance of RegnANN, ARACNE and CLR in terms of AUC (the area under the curve). The curve is constructed by varying the value of the binarization threshold between 0 and the maximum score obtained by the given inference method on the task at hand - in the case of RegnANN the maximum value is bounded to 1, while it is not the case for ARACNE and CLR. As in the previous phase, this is done while varying the topology of the network, the number of nodes and verifying the effect of different mean degrees and power-law coefficients. Also in this phase we consider discrete symmetric adjacency matrices for fair comparison between the methods: ARACNE and CLR do not infer coupling direction. For homogeneity, we introduce the MR (MCC-Recall) curve which express the MCC value for the corresponding Recall value. Although the Precision-Recall (PR) curve is a well known tool in assessing the performance of an induction algorithm, the MR is an equivalent measure that has a straightforward interpretation. The major difference between the two curves is that an induction algorithm scoring a AUC_{PR} value of 0.5 has performance substantially equivalent to chance, the same induction system would score an AUC_{MR} value of 0. In perfect analogy with the previous phase, data generation and network inference are repeated 10 times for each given network topology and the AUC score is estimated as the mean of the 10 independent runs. The error of the measurement is expressed as twice the standard deviation.

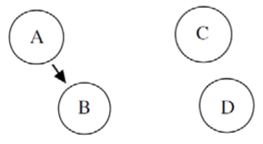
Finally, we compare the results obtained on a selection of Escherichia coli gene subnetworks for the four inference algorithms. We will first start considering a fixed threshold for the binarization of the inferred adjacency matrix. Secondly, we briefly analyze the problem of optimal threshold selection for the binarization of the inferred adjacency matrix in the hypothesis of the presence of gold standard data, which is not available in most realistic biological applications. As for the two phases before, we consider discrete symmetric adjacency matrices.

Toy Examples

In this section we present four different toy experiments to illustrate how RegnANN is capable of inferring direct and indirect interactions among genes and cooperative interaction between putative regulators (e.g.: transcription factors).

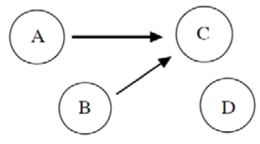
Single Interaction

Let us consider four different genes A, B, C and D, which interact according to Figure 1: gene A regulates B, while C and D do not interact with anyone. We assume that A is a regulator that can be in an active or inactive state. If it is in an active state, it has



	A	B	C	D
A	0	0	0	0
B	0	0	0	0
C	1	0	0	0
D	0	0	0	0

Figure 1. Left: gene *A* regulates gene *B*, while *C* and *D* do not interact with any other. Right: corresponding adjacency matrix.
doi:10.1371/journal.pone.0028646.g001



	A	B	C	D
A	0	0	0	0
B	0	0	0	0
C	1	1	0	0
D	0	0	0	0

Figure 3. Left: gene *A* and *B* cooperatively regulate gene *C*, while *D* does not interact with the other three. Right: corresponding adjacency matrix.
doi:10.1371/journal.pone.0028646.g003

the effect of a linear regulation on B. If it is in an inactive state, B is driven by noise. We generate 100 expression profiles for the four genes as follows: value of A is 0 with probability 50% (A is inactive), and uniformly distributed in [0,1] with probability 50%. The value of gene B is uniformly distributed in the interval $[0,thr]$ if A is inactive (B values are driven by noise), otherwise $B = A + threshold + noise$ (in the case the values of A are different from 0). The term *noise* is a value uniformly distributed in the interval $[0,thr]$. Expression values for genes C and D are uniformly distributed in the interval $[0,thr]$ - C and D are entirely driven by noise. The value for the threshold *thr* is set arbitrarily to 0.05. Before inference, gene expression profiles are linearly rescaled in $[-1,1]$.

The correlation matrix shown in Figure 2 indicates that RegnANN is able to capture the correlation among genes A and B: for interaction $A \rightarrow B$, RegnANN calculates a correlation of 0.93 (and a correlation of 0.88 for the opposite direction: $A \leftarrow B$). On the other hand no correlation (equal or less than 0.1) is recorded among C, D and the other genes. It is interesting to note that, if we select the direction of interaction by identifying the highest correlation value, RegnANN successfully identifies the regulation as in Figure 1.

Cooperative Interaction

Let us consider four different genes A, B, C and D, which interact according to Figure 3: gene A and gene B cooperatively regulate gene C, while D does not interact with the other three.

We synthesize expression profiles very similarly to the SLC method: we start considering a set of 100 *seed* expressions with values uniformly distributed in $[-1,1]$ for each gene A, B, C, D. In order to simulate the presence of an activation threshold, we calculate the expression value for the genes as follows:

$$
\begin{aligned}
gep_a &= seed_a \\
gep_b &= seed_b \\
gep_c &= \begin{cases} seed_a + seed_b + seed_c & \text{if } |seed_a|, |seed_b| > thr \\ seed_c & \text{otherwise} \end{cases} \\
gep_d &= seed_d
\end{aligned}
$$

where gep_a, gep_b, gep_c and gep_d are the gene expression profiles for gene A, B, C and D respectively, while $seed_a$, $seed_b$, $seed_c$ and $seed_d$ are the corresponding seed values. The value for the threshold *thr* is set arbitrarily to 0.05. After generation the profiles are rescaled linearly in $[-1,1]$. To account for intrinsic instability of the inference method, data generation and network inference are repeated 10 times and the adjacency matrix accumulated. Figure 4 shows the mean correlation values inferred by RegnANN (left) and the obtained interaction among genes for correlation greater than 0.50 (right).

The correlation matrix shown in Figure 4 indicates that RegnANN is able to capture the correlation among genes A, B and C: for interaction $A \rightarrow C$, RegnANN calculates a correlation of 0.67 (and a correlation of 0.33 for the opposite direction: $A \leftarrow C$). Similar correlation values are recoded for $B \rightarrow C$ and $B \leftarrow C$. It is interesting to note that no correlation (less than 0.1) is recorded among D and the other genes.

Multiple Interaction

Let us consider again the four different genes A, B, C and D interacting according to Figure 5: gene C regulates gene A and gene B, while D does not interact with the other three.

As in the previous case, we generate the gene expression profiles considering a set of 100 *seed* expressions with values uniformly distributed in $[-1, 1]$ for each gene A, B, C, D. We calculate the expression value for the genes as follows:

$$
\begin{aligned}
gep_a &= \begin{cases} seed_a + seed_c & \text{if } |seed_c| > thr \\ seed_a & \text{otherwise} \end{cases} \\
gep_b &= \begin{cases} seed_b + seed_c & \text{if } |seed_c| > thr \\ seed_b & \text{otherwise} \end{cases} \\
gep_c &= seed_c \\
gep_d &= seed_d
\end{aligned}
$$

where gep_a, gep_b, gep_c and gep_d are the gene expression profiles for gene A, B, C and D respectively, while $seed_a$, $seed_b$, $seed_c$ and $seed_d$ are the corresponding seed values. The value for the threshold *thr* is set arbitrarily to 0.05. Figure 6 shows the mean correlation values for the 10 data generation/inference iterations (left) and the obtained interaction among genes for correlation greater than 0.50 (right).

The correlation matrix in Figure 6 indicates that both $C \rightarrow A$ and $C \rightarrow B$ interactions are discovered (a correlation of about 0.74 in both cases), while a weak interaction $B \leftarrow A$ is recorded

	A	B	C	D
A	0.00	0.88	0.07	0.04
B	0.93	0.00	0.08	0.05
C	0.10	0.10	0.00	0.06
D	0.09	0.09	0.07	0.00

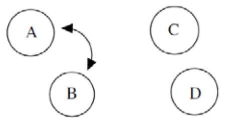


Figure 2. Left: mean correlation values inferred with RegnANN. Right: inferred interaction among the genes for correlation greater than 0.50.
doi:10.1371/journal.pone.0028646.g002

	A	B	C	D
A	0.00	0.06	0.33	0.09
B	0.06	0.00	0.33	0.07
C	0.67	0.64	0.00	0.07
D	0.09	0.06	0.04	0.00

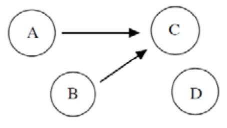
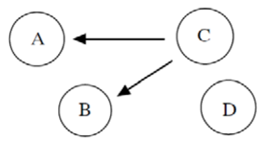


Figure 4. Left: mean correlation values inferred with RegnANN. Right: inferred interaction among the genes for correlation greater than 0.50.
doi:10.1371/journal.pone.0028646.g004



	A	B	C	D
A	0	0	1	0
B	0	0	1	0
C	0	0	0	0
D	0	0	0	0

Figure 5. Left: gene C regulates gene A and gene B, while D does not interact with the other three. Right: corresponding adjacency matrix.
doi:10.1371/journal.pone.0028646.g005

(correlation 0.43). Other two weak interactions are also reported: $A \rightarrow C$ (correlation 0.52) and $B \rightarrow C$ (correlation 0.49). Strictly considering the adjacency matrix in Figure 5 and a discretization threshold of 0.5, one false link is recorded ($A \rightarrow C$) in the inferred interaction matrix.

Indirect Interaction

As a final example, let us consider the four different genes A, B, C and D interacting according to Figure 7: gene A regulates gene B and gene B regulates gene C, while D does not interact with the other three genes.

We generate the gene expression profiles considering a set of 100 *seed* expressions with values uniformly distributed in $[-1,1]$ for each gene A, B, C, D. We calculate the expression values for the genes as follows:

$$\begin{aligned}
gep_a &= seed_a \\
gep_b &= \begin{cases} seed_b + seed_a & \text{if } |seed_a| > thr \\ seed_b & \text{otherwise} \end{cases} \\
gep_c &= \begin{cases} gep_b + seed_c & \text{if } |gep_b| > thr \\ seed_c & \text{otherwise} \end{cases} \\
gep_d &= seed_d
\end{aligned}$$

Figure 8 shows the mean correlation values for the 10 data generation/inference iterations (left) and the obtained interaction among genes for correlation strictly greater than 0.50 (right).

The correlation matrix in Figure 8 indicates that interaction $A \rightarrow B$ and interaction $B \rightarrow C$ are correctly identified with the highest correlation values (correlation 0.76 and correlation 0.72 respectively). A third strong interaction is also indicated by RegnANN: $A \rightarrow C$. The latter is in fact a second order interaction between the the two genes. In Table 1, Table 2 and Table 3 we report for comparison the inferred mutual information matrix inferred by ARACNE, the mean scores obtained applying CLR and the mean adjacency matrix inferred by KELLER on the same exercise.

Strictly considering the adjacency matrix in Figure 7, RegnANN finds 2 false links (Figure 8): $A \rightarrow C$ and $C \rightarrow B$ (if we include correlation value 0.50, the interaction $B \rightarrow A$ would be a third false positive).

	A	B	C	D
A	0.00	0.43	0.74	0.07
B	0.06	0.00	0.74	0.07
C	0.52	0.49	0.00	0.07
D	0.04	0.04	0.07	0.00

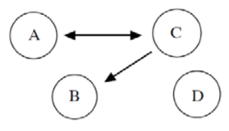
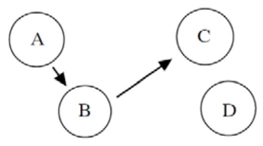


Figure 6. Left: mean correlation values inferred with RegnANN. Right: inferred interaction among the genes for correlation greater than 0.50.
doi:10.1371/journal.pone.0028646.g006



	A	B	C	D
A	0	0	0	0
B	1	0	0	0
C	0	1	0	0
D	0	0	0	0

Figure 7. Left: gene A regulates gene B and gene B regulates gene C, while D does not interact with the other three. Right: corresponding adjacency matrix.
doi:10.1371/journal.pone.0028646.g007

In order to eliminate the ambiguity in the determination of the direction of the interaction among genes, in the following we will consider symmetric adjacency matrices as input for data generation and as the output of the network inference task. As a second possible solution, in the case of RegnANN we could have selected the direction of interaction by identifying the highest correlation value. On the other hand, this choice would have resulted in an unfair comparison with (i.e.) KELLER: the latter discards any information about the direction of the interaction. However, it is important to consider that an exhaustive analysis of the direction of the coupling would require a dedicated procedure to account for the variability of the regression.

Synthetic Networks

In this section we analyze the performance of RegnANN in inferring network topology by applying a fixed discretization threshold of 0.5 on the correlation values obtained. These results are systematically compared with the ones obtained by the KELLER algorithm on the same set of tasks. Data generation and network inference are repeated 10 times for each given network topology and the MCC score is estimated as the mean of the 10 independent runs. The error of the measurement is expressed as twice the standard deviation.

Effect of sample availability

Figure 9 shows accuracy (MCC) scores on synthetic Barabasi networks with 50 nodes while varying data synthesis methodology and *data ratio*: the ratio of the number of expression profiles to the number of nodes. We rescale linearly the synthetic gene expression values in $[-1,1]$. Moreover, we generate Barabasi networks with power-law equal to 1 and Erdös-Rényi networks with mean degree equal to 1. Figure 9 indicates that increasing the available data is beneficial to the performance of all the inference methods tested irrespective of the data synthesis method applied. RegnANN-SLC - Figure 9(b) - shows MCC scores ranging between 0.21 ± 0.09 and 0.43 ± 0.09 for data ratio 25% and data ratio 200% respectively. For the same values of data ratio, KELLER-SLC scores 0.19 ± 0.08 and 0.38 ± 0.07. Similar behavior is recorded for GES data synthesis - Figure 9(a).

Figure 10 shows accuracy scores on synthetic Erdös-Rényi networks with 100 nodes while varying data synthesis methodology and data ratio. In the case of GES data synthesis, RegnANN

	A	B	C	D
A	0.00	0.50	0.40	0.05
B	0.76	0.00	0.66	0.07
C	0.70	0.72	0.00	0.07
D	0.04	0.04	0.03	0.00

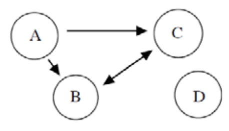


Figure 8. Left: mean correlation values inferred with RegnANN. Right: inferred interaction among the genes for correlation greater than 0.50.
doi:10.1371/journal.pone.0028646.g008

Table 1. Mean mutual information values over 10 runs obtained applying ARACNE for inferring indirect interactions.

	A	B	C	D
A	0.00	0.48	0.00	0.00
B	0.48	0.00	0.63	0.00
C	0.00	0.63	0.00	0.00
D	0.00	0.00	0.00	0.00

doi:10.1371/journal.pone.0028646.t001

scores 0.54 ± 0.08 for data ratio 150%. Similarly, KELLER scores 0.44 ± 0.04 for data ratio 150%. In the case of SLC and for data ratio 200%, RegnANN scores 0.91 ± 0.05, KELLER scores 0.84 ± 0.07.

Figure 11 shows accuracy scores on synthetic Erdös-Rényi and Barabasi networks with 200 nodes while varying the data ratio. The figure shows very good performance of RegnANN on both Barabasi and Erdös-Rényi topology. The method we propose shows accuracy scores constantly above the MCC values obtained with the other method. It is interesting to note that the two different topologies have significative influence on the accuracies of the methods tested. Considering data ratio 150%, RegnANN scores 0.49 ± 0.08 and 0.92 ± 0.03 on Barabasi and Erdös-Rényi networks respectively.

Figure 12 summarizes the accuracy scores on synthetic Erdös-Rényi and Barabasi networks obtained with Data Ratio 100% while varying number of nodes and data generation method.

Effect of mean degree and power-law coefficient

In the following we show results obtained by varying the mean degree of the nodes between 1 and 4 for Erdös-Rényi networks We also test the performance of the two inference algorithms by varying the exponent of the power-law in the range [1,4] for Barabasi networks. Here we consider synthetic networks of 100 nodes, SLC data generation and expression linearly rescaled in $[-1,1]$.

Figure 13(a) shows accuracy scores that decrease as the mean degree/power-law coefficient increases.

In the case of Erdös-Rényi topology with mean degree equal to 1, RegnANN scores 0.90 ± 0.04; in the case of mean degree equal to 4, the same algorithm scores 0.39 ± 0.04. Also KELLER shows an accuracy curve for the Erdös-Rényi topology that decreases as the mean degree of the network increases, although this behavior is less marked.

In the case of Barabasi topology, the accuracy of both methods drops quickly to a value of 0 as the exponent of the power-law increases.

Table 2. Mean scores over 10 runs obtained applying CLR for inferring indirect interactions.

	A	B	C	D
A	0.00	1.87	0.38	1.01
B	1.87	0.00	2.73	0.90
C	0.38	2.73	0.00	0.58
D	1.01	0.90	0.58	0.00

doi:10.1371/journal.pone.0028646.t002

Table 3. Mean adjacency matrix obtained applying KELLER for inferring indirect interactions.

	A	B	C	D
A	0.00	1.00	0.30	0.00
B	1.00	0.00	1.00	0.00
C	0.30	1.00	0.00	0.00
D	0.00	0.00	0.00	0.00

Value 0.3 indicates that the link has been detected 3 times in 10 runs.
doi:10.1371/journal.pone.0028646.t003

It is interesting to note that for both topologies, KELLER tends to outperform RegnANN for mean degree values bigger and equal to 3, when we consider a data ratio of 150.

Figure 13(b) shows the accuracy score of Erdös-Rényi networks with mean degree equal to 2. As in the case of Erdös-Rényi networks with mean degree equal to 1 - Figure 10(b) - the performance of the two methods tends to increase as the amount of data available increases.

Effect of data normalization

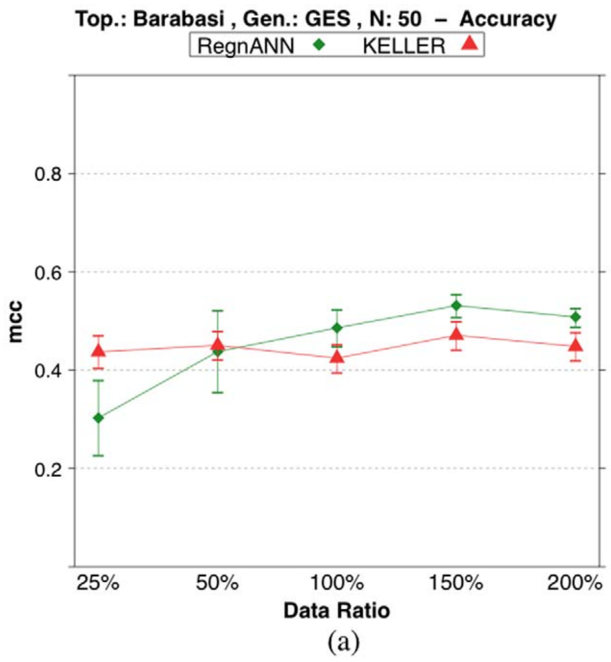
In microarray experiments, the analysis of the raw data is often hampered by a number of technical and statistical problems. The possible remedies usually lie in appropriate preprocessing steps, proper normalization of the data and application of statistical testing procedures in the derivation of differentially expressed genes [31]. Although many of the real-world issues in data preprocessing and normalization do not apply here, we are interested in verifying how discretization and rescaling - some of the most common (and possibly simple) steps taken to normalize the raw data - can impact the accuracy of the network inference algorithms here considered. Full details of the normalization procedures applied are given in the Material and methods of he paper.

Figure 14 shows the accuracy (MCC) of RegnANN and KELLER while varying the data normalization applied to the synthetic levels. The performance of KELLER significantly depends on the data normalization applied. In the case of data discretization, KELLER scores 0.21 ± 0.03, while an MCC value of 0.42 ± 0.04 is recorded in the case of linear rescaling. Finally, if statistical normalization is applied to the data, KELLER scores 0.43 ± 0.07. On the contrary, the accuracy of RegnANN is invariant (taking into account the error of the measure) with regards to the data normalization schema applied.

Effect of variable binarization threshold

In this section we compare the performance of ARACNE, CLR and RegnANN varying the threshold applied to discretize the inferred adjacency matrix in terms of the the AUC (area under the curve) value.

Figure 15 shows a sample Precision-Recall curve (continuous line) and MCC-Recall curve (dashed line) for RegnANN on a synthetic Barabasi network (left) and on a synthetic Erdös-Rényi network (right). In the given example we considered 100 nodes, 100% data ratio and SLC data generation. The AUC values for in the case of Barabasi network are: $AUC_{PR} = 0.43$ and $AUC_{MR} = 0.32$. For the given Erdös-Rényi network the AUC values are: $AUC_{PR} = 0.84$ and $AUC_{MR} = 0.59$. The figure shows curves that are in substantial agreement with the results reported in the previous section: the accuracy of the inference task strongly depends on the topology of the network.



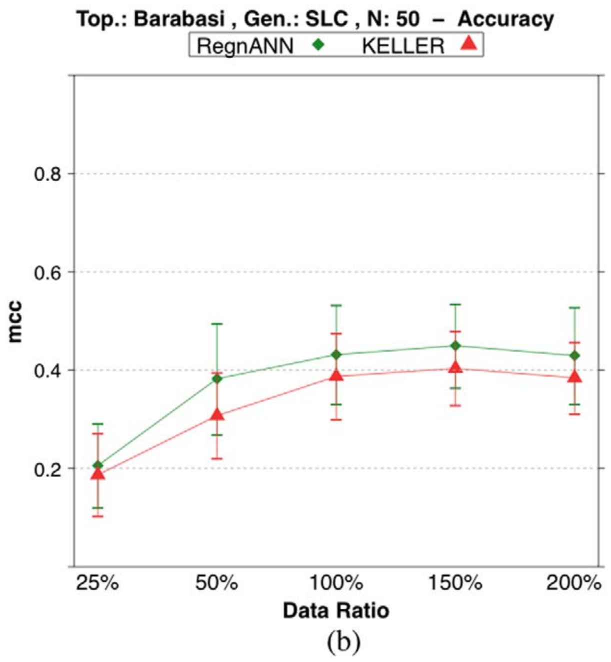
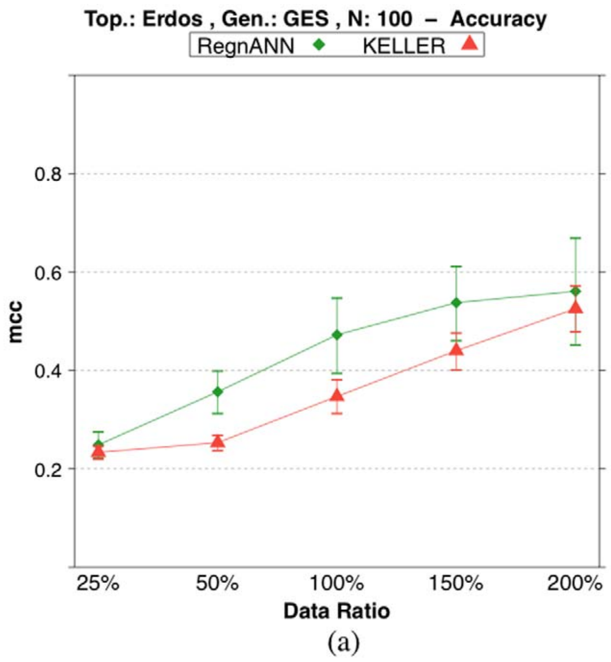


Figure 9. Accuracy (MCC) scores of RegnANN and KELLER on synthetic Barabasi networks with 50 nodes. (a) Results obtained with GES data generation. (b) Results obtained with SLC data generation.
doi:10.1371/journal.pone.0028646.g009

Figure 16 shows the mean AUC_{MR} scores for ARACNE, CLR and RegnANN on a synthetic Barabasi network and on a synthetic Erdös-Rényi network, 100 nodes and SLC data generation, varying data ratio.

The figure indicates that the mean performance (AUC_{MR}) of the three methods obtained varying the discretization threshold are equivalent considering the confidence intervals, e.g.: in the case of data ratio 100% and Barabasi networks ARACNE scores 0.34 ± 0.06, CLR scores 0.42 ± 0.07 and RegnANN scores 0.39 ± 0.08. In the case of Erdös-Rényi networks, data ratio 150% ARACNE scores 0.60 ± 0.07, CLR scores 0.61 ± 0.03 and RegnANN scores 0.63 ± 0.02.

Figure 17 summarizes the AUC_{MR} scores obtained by the three inference algorithms varying the number of nodes in the synthetic



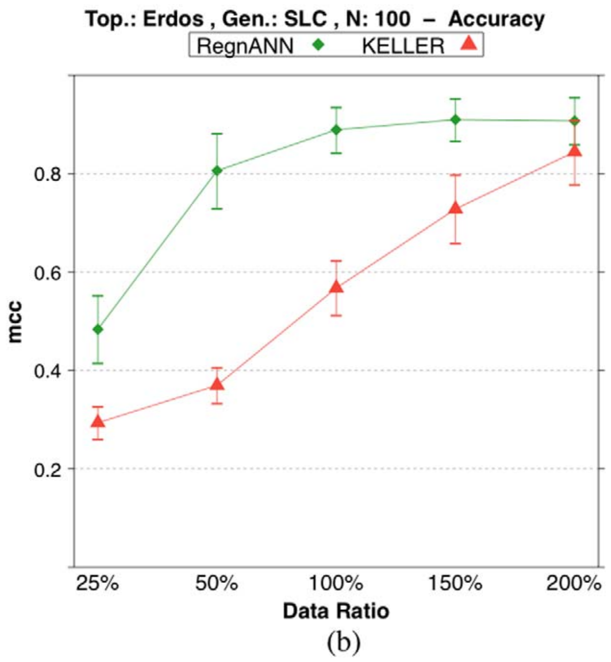


Figure 10. Accuracy (MCC) scores of RegnANN and KELLER on synthetic Erdös-Rényi networks with 100 nodes. (a) Results obtained with GES data generation. (b) Results obtained with SLC data generation.
doi:10.1371/journal.pone.0028646.g010

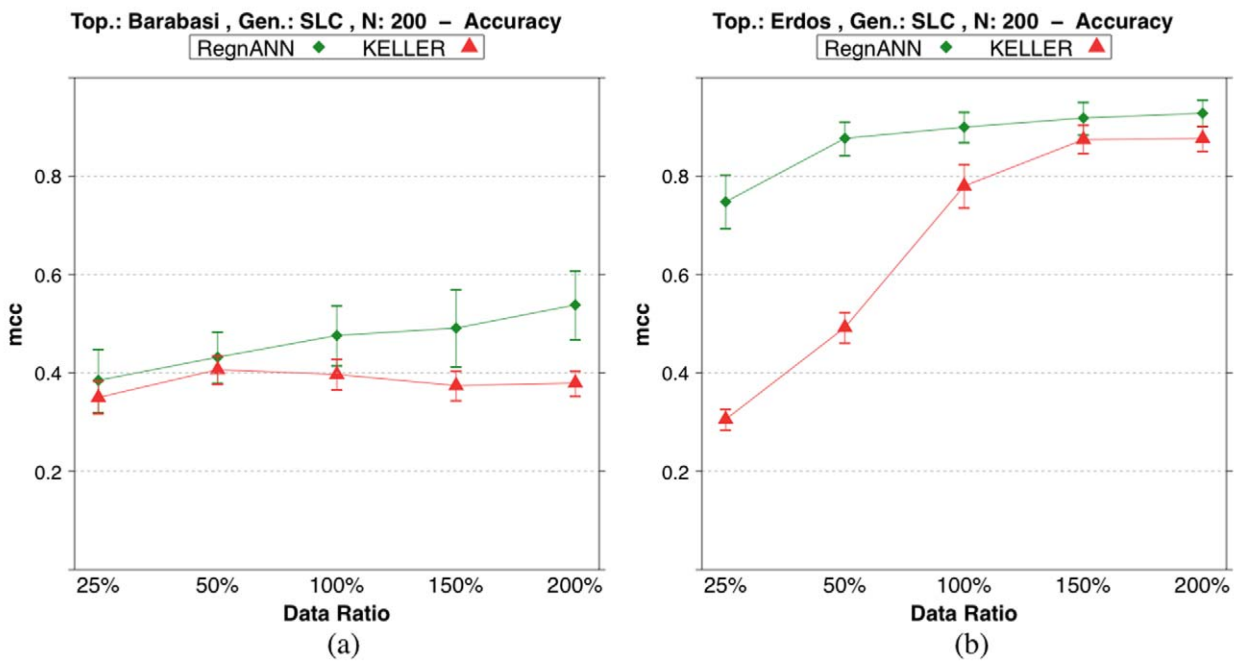


Figure 11. Accuracy (MCC) scores of RegnANN and KELLER on synthetic networks with 200 nodes, SLC data generation. (a) Results obtained for Barabasi topology. (b) Results obtained for Erdös-Rényi topology.
doi:10.1371/journal.pone.0028646.g011

network. In the case of Barabasi networks, GES data generation, CLR tends of perform better than the other two methods. However, this phenomenon may not be strictly statistically significant, e.g.: with 200 nodes, ARACNE scores 0.32 ± 0.06, CLR scores 0.42 ± 0.03, while RegnANN scores 0.33 ± 0.7. In the case of Erdös-Rényi networks, the three methods are equivalent, e.g.: considering 200 nodes, ARACNE scores 0.62 ± 0.02, CLR scores 0.63 ± 0.02 while RegnANN scores 0.64 ± 0.02.

Figure 18 shows the mean AUC_{MR} scores for ARACNE, CLR and RegnANN on a synthetic Barabasi network generated using a power-law exponent equal to 2 and on a synthetic Erdös-Rényi network with mean degree equal to 2. We consider 100 nodes and

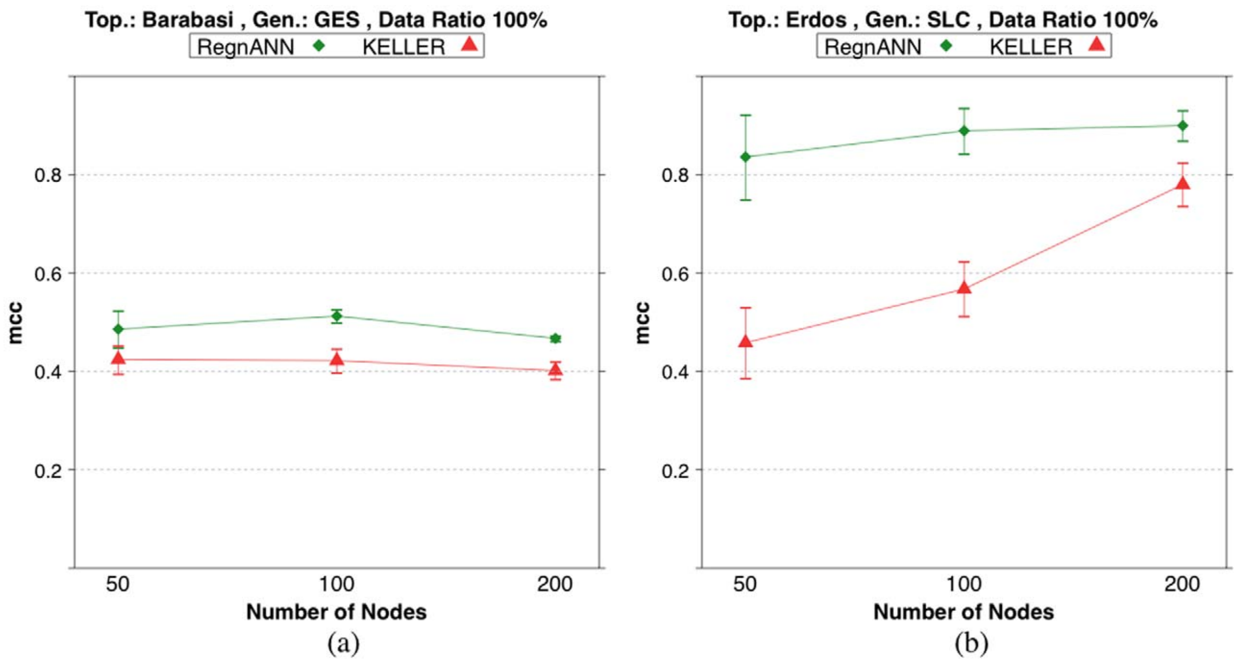


Figure 12. Accuracy (MCC) scores of RegnANN and KELLER on synthetic networks with Data Ratio 100% while varying number of nodes. (a) Results obtained for Barabasi topology, GES data generation. (b) Results obtained for Erdös-Rényi topology, SLC data generation.
doi:10.1371/journal.pone.0028646.g012

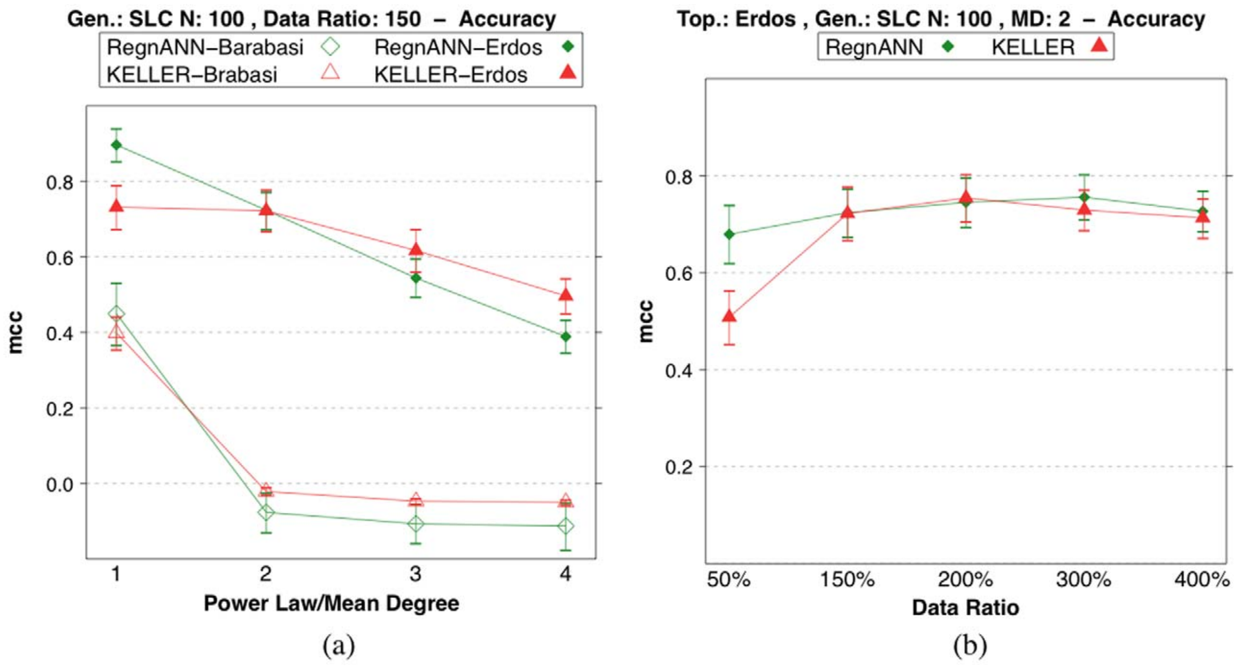


Figure 13. Accuracy (MCC) scores of RegnANN and KELLER on synthetic networks with 100 nodes and SLC data generation. (a) Results obtained on Barabasi networks varying the power-law coefficient and Erdös-Rényi topology while varying the mean degree of the network. (b) Results obtained for Erdös-Rényi (mean degree value equal to 2) while varying the data ratio.
doi:10.1371/journal.pone.0028646.g013

SLC data generation, while we vary the data ratio. In agreement with the results shown in Section Mean Degree and Power-law Coefficient, the figure indicates that increasing the power-law coefficient for the synthetic Barabasi network is detrimental to the performance of all the inference algorithm tested: ARACNE, CLR and RegnANN show mean AUC_{MR} scores of 0, random chance. On the other hand, the three methods are less affected by increasing the mean degree of the synthetic Erdös-Rényi network, as also noticed previously.

Escherichia coli transcriptional subnetworks

Table 4 summarizes the results obtained on a selection of *Escherichia coli* gene subnetworks [32] for the four inference algorithms. In absence of a gold standard data (not available in most realistic biological applications) for the estimation of the threshold for the binarization of the adjacency matrix, we arbitrarily set such threshold to 0.001 for ARACNE and CLR. This is done in the hypothesis that a value different from zero indicates meaningful interaction. It is important to stress the fact that the direction of the interaction is discarded as only symmetric matrices are considered. In the case of RegnANN, we set this threshold to 0.5 (we hypothesize that correlation values bigger than 0.5 indicate meaningful interaction). In Section Selecting the Binarization Threshold we briefly discuss a possible strategy to infer the optimal threshold and the related shortcomings.

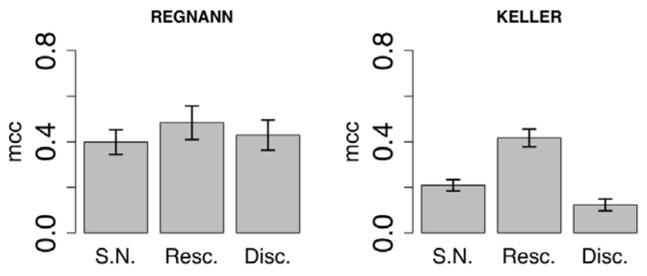


Figure 14. Accuracy (MCC) scores of RegnANN (left) and KELLER (right) on synthetic Erdös-Rényi networks with 100 nodes, SLC data generation and data ratio equal to 150, while varying data normalization.
doi:10.1371/journal.pone.0028646.g014

While ARACNE, CLR and KELLER are deterministic algorithms - given a particular input, the algorithm will always produce the same output, always passing through the same sequence of states - RegnANN may produce different results depending on the random initialization of the weights in the ensemble of multi-layer perceptrons. Thus, in order to smooth out possible local minima, we adopted a majority voting schema: for each network module, the RegnANN algorithm is applied 10 times and the inferred adjacency matrices accumulated. The final topology is obtained by selecting those links that appeares with a frequency higher than 7. The entire procedure is repeated 10 times, the final prediction is estimated as the mean and the associated error as twice the standard deviation of the 10 independent runs. Gene expression values are linearly rescaled in $[-1,1]$.

Table 4 indicates great variability of the MCC scores across the different network modules for all the inference methods tested. ARACNE scores range from 0.78 (module 81) to 0.00 (module 88). CLR values range between 0.45 and 0.02 for module 81 and 96 respectively. KELLER scores range between 0.63 and -0.12 (module 12 and module 81 respectively). Finally RegnANN scores range between 0.32 ± 0.00 (module 12, in this case the error associated to the measure is 0: the very same result is obtained for all repetitions) and -0.05 ± 0.02 (module 88). It is interesting to note that the MCC score varies unevenly for the reference inference algorithms with respect to the module network density (the ratio of the number of links to the square of the number of

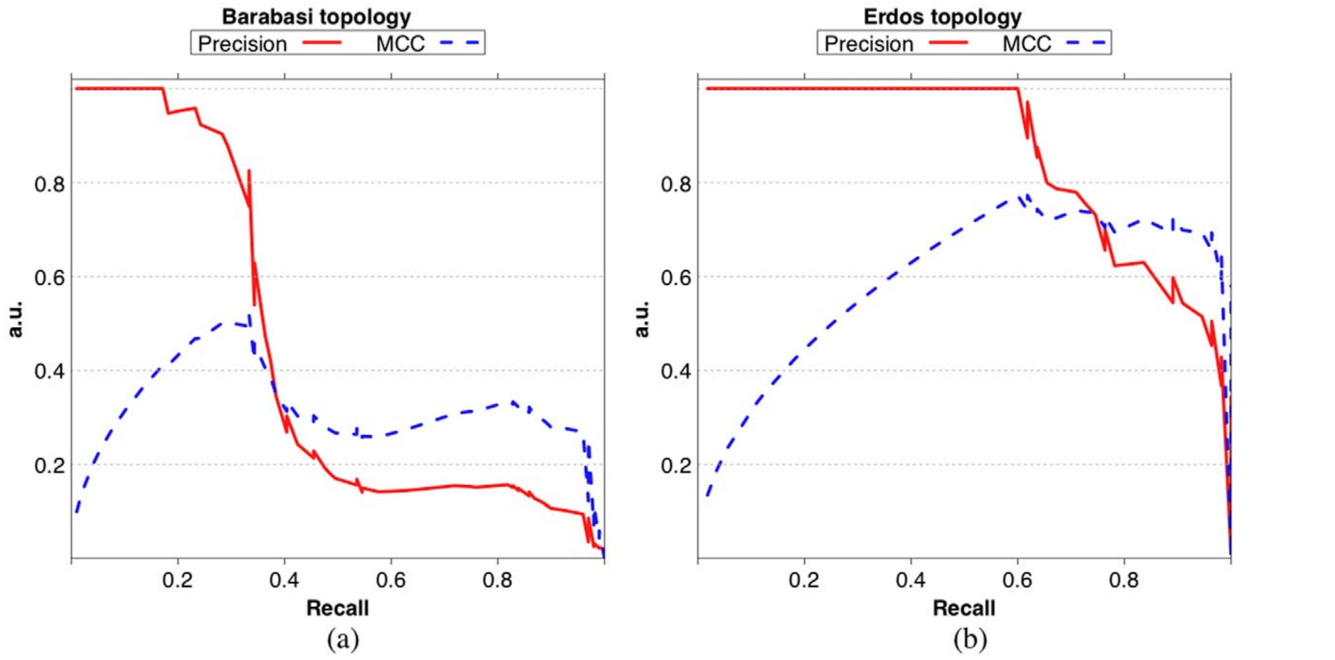


Figure 15. Precision-Recall curve (continuous line) and MCC-Recall curve (dashed line) for RegnANN on a synthetic Barabasi network (a) and on a synthetic Erdös-Rényi network (b), 100 nodes, 100% data ratio and SLC data generation.
doi:10.1371/journal.pone.0028646.g015

nodes), e.g.: ARACNE scores 0.13 on module 6 (density $D=0.189$) and scores 0.43 on module 12 (density $D=0.189$). On the same two modules, CLR scores 0.29 and 0.39 respectively while KELLER scores 0.02 and 0.63. On the other hand, RegnANN scores are more homogeneous: they read 0.3 ± 0.1 and 0.32 ± 0.00 on module 6 and module 12 respectively. These results suggest that the correctness of the inferred network depends on the topological properties of the modules (the very same expression values are used to infer the different gene sub-networks), in accordance to findings in [4].

Selecting the binarization Threshold

In this section we analyze the problem of optimal threshold selection for the binarization of the inferred adjacency matrix.

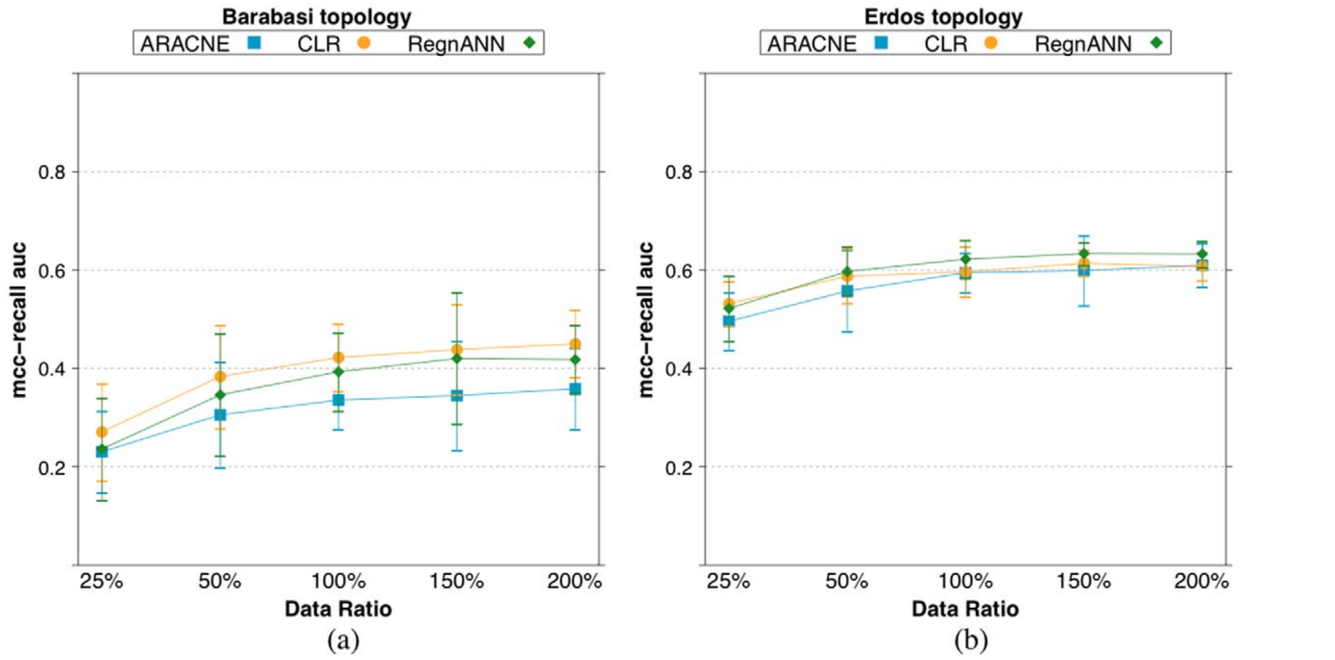
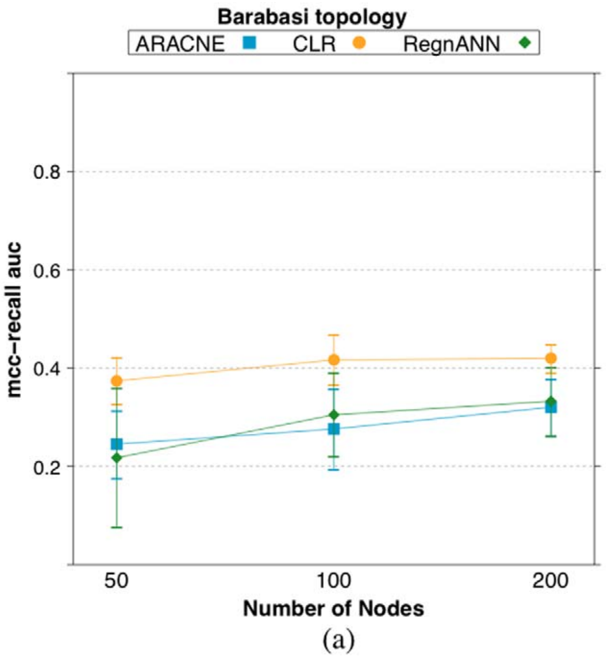


Figure 16. AUC_{MR} scores for ARACNE, CLR and RegnANN on a synthetic Barabasi network (a) and on a synthetic Erdös-Rényi network (b), 100 nodes and SLC data generation, varying data ratio.
doi:10.1371/journal.pone.0028646.g016



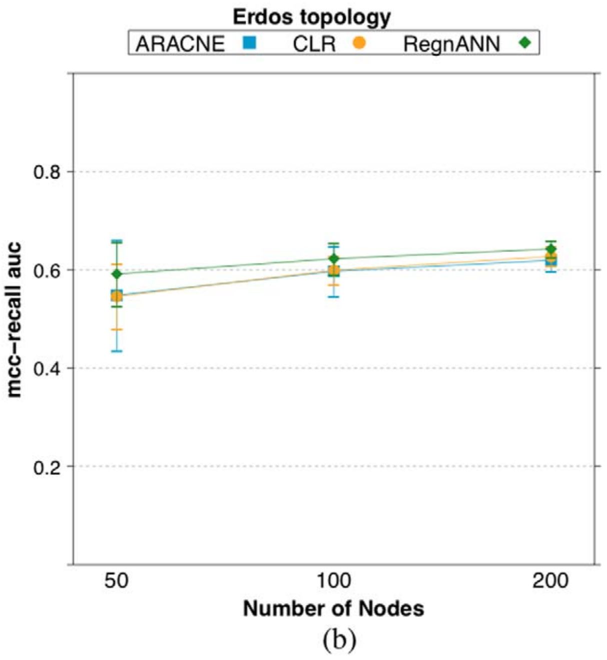
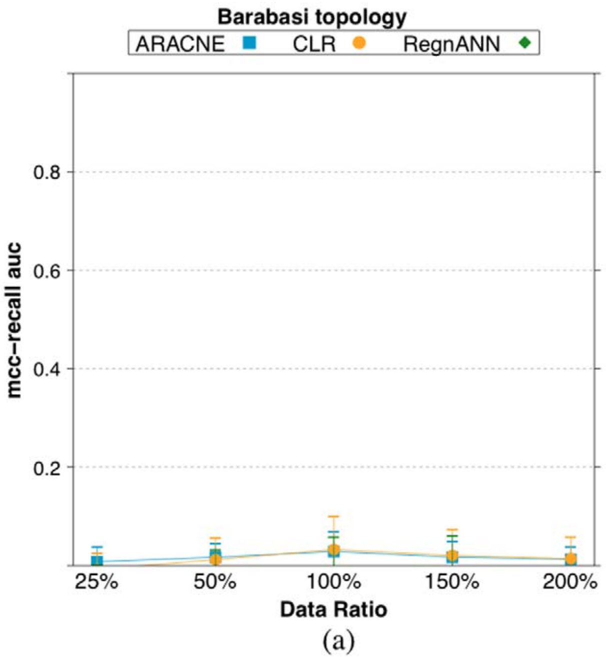


Figure 17. AUC_{MR} scores for ARACNE, CLR and RegnANN on synthetic networks varying the number of nodes, while keeping constant the data ratio (100%). (a) Synthetic Barabasi topology, GES data generation. (b) Synthetic Erdös-Rényi network, SLC data generation.
doi:10.1371/journal.pone.0028646.g017

Generally, the threshold cutoff value should be estimated for each data-set (network) or it should be derived from a gold standard data, which is not available in most realistic biological applications. An obvious solution to this problem is to adopt a training/validation schema: ground-truth data is used to infer the optimal threshold value while external data is used to verify the reconstruction accuracy. Here, for each module in Table 4 and for three inference algorithms (ARACNE, CLR and RegnANN), we partition the *Escherichia coli* data-set in training data (70%) and validation data (30%). The training data is used to infer the best threshold cutoff value: we perform a grid search for threshold values in [0,1] and we pick the value granting the best



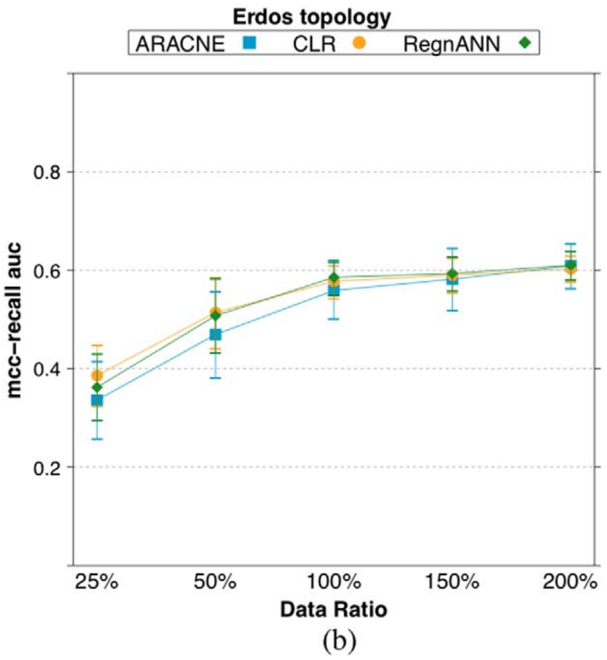


Figure 18. AUC_{MR} scores for ARACNE, CLR and RegnANN on a synthetic Barabasi network with power-law coefficient equal to 2 (a) and on a synthetic Erdös-Rényi network (b) with mean degree equal to 2. Here we consider 100 nodes and SLC data generation, while we vary the data ratio.
doi:10.1371/journal.pone.0028646.g018

Table 4. MCC scores of the different network inference algorithms on the selected *Escherichia coli* network modules.

ID	Density	N.N.	N.L.	ARACNE	CLR
81	0.245	7	12	0.78	0.45
6	0.189	13	32	0.13	0.29
12	0.180	10	18	0.43	0.42
75	0.133	16	34	0.10	0.24
88	0.100	19	36	0.00	0.17
96	0.001	104	18	0.08	0.02
94	0.000	81	2	0.09	0.02
ID	**Density**	**N.N.**	**N.L.**	**KELLER**	**RegnANN (Err)**
81	0.245	7	12	−0.12	0.4 (0.1)
6	0.189	13	32	0.02	0.3 (0.1)
12	0.180	10	18	0.63	0.32 (0.00)
75	0.133	16	34	0.10	0.23 (0.08)
88	0.100	19	36	−0.07	−0.05 (0.02)
96	0.001	104	18	0.00	0.00 (0.01)
94	0.000	81	2	0.15	0.026 (0.001)

Column ID indicates the id of the network module as in [32], column Density the density of the module is the ratio of the number of links to the square of the number of nodes. Column N.N. indicates the nuber of nodes in the subgraph, while N.L. the number of links.
doi:10.1371/journal.pone.0028646.t004

reconstruction score. The validation set is used to verify the accuracy (MCC) of the inference algorithms. This procedure is repeated 50 times randomly partioning the *Escherichia coli* data-set each time. Scores are expressed as the mean value, while the error as twice the standard deviation.

Table 5 shows the optimal threshold cutoff values for the three inference methods (ARACNE, CLR and RegnANN) and for the *Escherichia coli* submodules as in Table 4.

The table indicates that the optimal threshold value depends on the algorithm adopted and the submodule considered.

Table 6 shows the accuracy (MCC) obtained on the training set and on the validation set with the optimal threshold cutoff value. Scores are obtained varying inference method and *Escherichia coli* submodule. Table 6 shows values that vary considerably depending on the subnetwork selected and the inference algorithm adopted. It is interesting to note that although the variance of the MCC scores is small (the column Error in the table) the accuracy on the validation set is often higher than the corresponding the training set accuracy score, e.g.: module 81, 6, 12, 94 in the case of ARACNE; module 81 and module 6 in the case of CLR; module 12 in the case of RegnANN. On the other hand, there are cases where good training accuracy scores are not matched with comparable validation accuracies, e.g.: module 75, 88, 96 in the case of ARACNE; module 75, 88 in the case of CLR. This results indicates both a good stability in replicating the results using the estimated optimal threshold (small error), and a tendency to both poorly fit and over fit the data.

Table 5. Mean values of the optimal threshold cutoff scores for the three inference methods (ARACNE, CLR and RegnANN) varying the *Escherichia coli* submodules.

ID	ARACNE	Err.	CLR	Err.	RegnANN	Err.
81	0.04	0.01	0.61	0.07	0.01	0.01
6	0.35	0.06	0.15	0.01	0.22	0.03
12	0.12	0.01	0.4	0.2	0.36	0.01
75	0.42	0.02	0.5	0.2	0.59	0.03
88	0.39	0.01	0.33	0.06	0.43	0.03
96	0.87	0.01	0.74	0.01	0.66	0.01
94	0.81	0.01	0.62	0.05	0.95	0.01

doi:10.1371/journal.pone.0028646.t005

Although outside the scope of this work, this preliminary evaluation indicates that in the case of biological data, learning the optimal threshold value via standard machine leaning methods is not straightforward: presence of noise in the data and the high complexity of the domain often cause selection bias. This is the key point that lead us focus on estimating the structures of the interaction between genes rather than the detailed strength of these interactions.

Discussion

In this work we presented a novel method for network inference based on an ensemble of multi-layer perceptrons configured as multi-variable regressor (RegnANN). We compared its performance to the performance of three different network inference algorithms (ARACNE, CLR and KELLER) on the task of reverse engineering the gene network topology, in terms of the associated MCC score. The proposed method makes no assumptions about the nature of the relationships between the variables, capturing high-order dependencies between expression patterns and the direction of the interaction, as shown on selected synthetic toy examples. Our extensive evaluation indicates that the newly introduced RegnANN shows accuracy and stability scores that compare very favorably with all the other inference methods tested, often outperforming the reference algorithm in the case of fixed binarization threshold. On the other hand, considering all the possible thresholds for the binarization of the inferred adjacenci matrix (the AUC score) the differences among the tested methods tend to become irrelevant. Our evaluation on

Table 6. Mean accuracy (MCC) obtained on the training set and on the validation set for the different inference methods and network submodules.

ID	Training	Error	Validation	Error
ARACNE				
ID	**Training**	**Error**	**Validation**	**Error**
81	0.21	0.01	0.57	0.03
6	0.04	0.01	0.19	0.02
12	0.26	0.01	0.32	0.03
75	0.54	0.02	0.14	0.01
88	0.75	0.01	0.03	0.01
96	0.71	0.01	0.27	0.02
94	0.05	0.01	0.40	0.10
	CLR			
ID	**Training**	**Error**	**Validation**	**Error**
81	0.16	0.01	0.57	0.02
6	0.03	0.01	0.31	0.01
12	0.36	0.01	0.36	0.01
75	0.46	0.01	0.13	0.01
88	0.71	0.01	0.08	0.01
96	0.03	0.01	0.04	0.01
94	0.11	0.01	0.03	0.01
	RegnANN			
ID	**Training**	**Error**	**Validation**	**Error**
81	0.24	0.01	0.23	0.01
6	0.34	0.01	0.10	0.01
12	0.22	0.01	0.31	0.02
75	0.40	0.01	0.17	0.01
88	0.23	0.01	0.08	0.01
96	0.03	0.01	0.02	0.01
94	0.12	0.01	0.00	0.01

doi:10.1371/journal.pone.0028646.t006

synthetic data demonstrates that various factors influence the performance of the inference algorithms adopted: the topology of network, its size and its complexity, the amount of data available, the normalization procedure adopted. Generally, these are only a few of the factors that may influence the outcome of a network inference algorithm; they may not be limited to the relative small set of parameters explored here.

Results on the biological data confirm that the correctness of the inferred network depends on the topological properties of the modules: very different accuracy results are obtained on the different submodules of *Escherichia coli*, although the very same expression values are used to infer the different gene sub-networks. Our experiments indicate great variability of the scores of the reference inference algorithms across the different *Escherichia coli* sub-modules. On the other hand, RegnANN scores are more homogeneous, decreasing as the density of the module decreases.

Finally, we tested the possibility of applying standard machine leaning methods to learn the optimal binarization threshold value. Our preliminary evaluation indicates that this is not a straightforward task: presence of noise in the data and the high complexity of the biological domain often cause selection bias.

The robustness of RegnANN performance recorded across the board and the agreement between results obtained with this new methods on both synthetic and biological data are promising and they stimulate its application to a wider range of problems.

Materials and Methods

RegnANN: Network Inference Using ANN

To infer gene regulatory networks we adopt an ensemble of feed-forward multilayer perceptrons. Each member of the ensemble is essentially a multi-variable regressor (one to many) trained using an input expression matrix to learn the relationships (correlations) among a target gene and all the other genes. Formally, let us consider the multilayer perceptron as in Figure 19 (right): 1 input neuron I, 1 layer of H hidden units and 1 layer of K output units. Indicating with g the activation function of each unit and $w_{h,k}$ the weights associated with the links between the output layers and the hidden layer and with $\hat{w}_h$ the weights of the links between input neuron and hidden layer, the value O_k for the output unit k can be calculated as follows:

$$O_k = g(\sum_{h=1}^{H} w_{h,k} \cdot g(\hat{w}_h \cdot I)) \qquad (1)$$

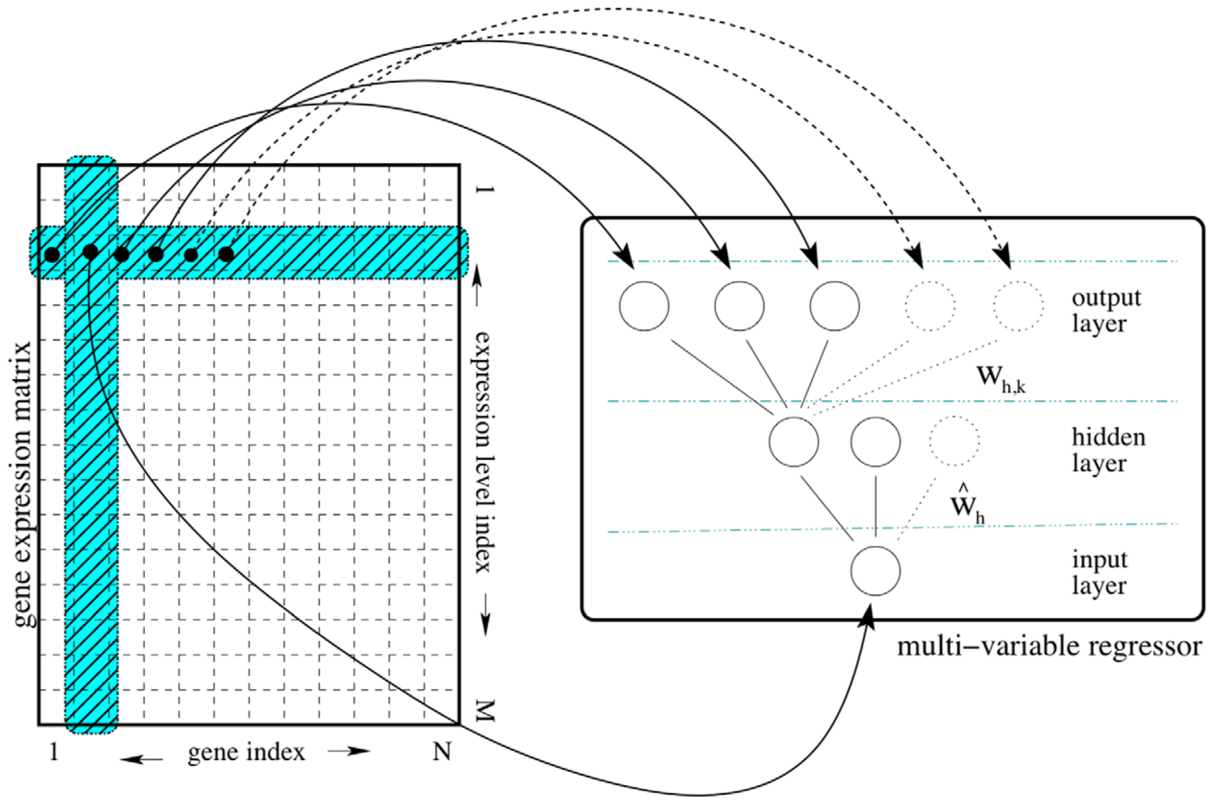


Figure 19. The *ad hoc* procedure proposed to build the training input/output patterns starting from a gene expression matrix. Each input pattern corresponds to the expression value for the selected gene of interest. The corresponding output pattern is the vector of expression values for all the other genes for the given row in the gene expression matrix. The right part of the figure schematizes the multi-variable regressor: a feed-forwardad multilayer perceptron with 1 input neuron, 1 layer of H hidden units and 1 layer of K output units; $w_{h,k}$ are the weights associated with the links between the output layers and the hidden layer and $\hat{w}_h$ the weights of the links between the input neuron and the hidden layer.
doi:10.1371/journal.pone.0028646.g019

The value O_k is the inferred interaction between the corresponding gene k and the gene associated with the input neuron I. We proceed in determining the interactions among genes separately and then we join the information to form the overall gene network. From each row of the gene expression matrix we build a set of input and output patterns used to train with back-propagation [33] a selected multilayer perceptron. Each input pattern corresponds to the expression value for the selected gene of interest. In this work we consider gene expression matrices of dimension $M \times N$, i.e. N genes whose expression levels are recorded M times; expression levels are normalized in the interval $[-1,1]$. The output pattern is the row-vector of expression values for all the other genes for the given row in the gene expression matrix (Figure 19). By cycling through all the rows in the matrix, each regressor in the ensemble is trained to learn the correlations among one gene and all the others. Repeating the same procedure for all the columns in the expression matrix, the ensemble of multi-variable regressors is trained to learn the correlations among all the genes. The procedure of learning separately the interactions among genes is very similar to the one presented in [20], where the authors propose to estimate the *neighborhood* of each gene (the correlations among one gene and all the others) independently and then joining these neighborhoods to form the overall network, thus reducing the problem to *a set of identical atomic optimizations*.

We build N (one for each of the N genes in the network) multilayer perceptrons with one input node, one layer of hidden nodes and one layer of $N-1$ output nodes, adopting the hyperbolic tangent as activation function. The input node takes the expression value of the selected gene rescaled in $[-1,1]$. The number of hidden nodes is set to the square root of the number of inputs by the number of outputs. This value is to be considered a rule of thumb granting enough hidden units to solve the regression problem and allowing dynamical adaptation of the structure of RegnANN to the size of the biological network under study. The output layer provides continuous output values in the range $[-1,1]$.

The algorithm of choice for training each multi-layer perceptron is the back-propagation algorithm [33]. The back-propagation is a standard algorithm for learning feed-forward multilayer perceptrons that essentially looks for the minimum of the error function in the weight space using the method of gradient descent. The error function is defined as the difference between the output of each neuron in the multilayer perceptron and its expected value. The back-propagation algorithm starts with the forward-propagation of the input value in the multilayer perceptron, followed by the backward propagation of the errors from the output layer toward the input neuron. The algorithm corrects the weight values according to the amount of error each unit is responsible for. Formally, the weight values at learning epoch τ are updated as follows:

$$\Delta w^{(\tau)} = -\eta \nabla E + \mu \Delta w^{(\tau-1)} \qquad (2)$$

To keep the notation simple w refers to both the weights associated with the links between the output layers and the hidden layer and with the weights of the links between input neuron and hidden layer. ∇E refers to the gradient of the error in weight space. η is the learning rate; μ is the momentum.

Although back-propagation is essentially a heuristic optimization method and alternatives such as Bayesian neural network learning [34] have more sound theoretical basis, in the proposed multi-variable regression schema the simple back-propagation algorithm allows us to design a far less complex system. This is due to how Bayesian neural network learning handles the regression problem. As indicated in [35]: "Networks are normally used to define models for the conditional distribution of a set of target values given a set of input values.[...]. For regression and logistic regression models, the number of target values is equal to the number of network outputs." This implies that in the case of Bayesian learning an extra procedure is required to discretize the target values from the continuous range $[-1,1]$ and that for each ensemble member the layer of output neurons ($N-1$ in the case of back-propagation) has to be translated into a matrix of neurons of size $(N-1)\times T$, where T is the number of desired target values. Accordingly, also the hidden layer becomes a matrix of neurons, each one with its own set of parameters. Thus, in the context of multivariable regression, adopting back-propagation allow us to design a lower complexity inference system limiting issues related to high dimensional settings.

The learning parameters we use to train each multi-layer perceptron are as follows: learning rate equal to 0.01; momentum equal to 0.1, learning epochs equal to 1000. These values are evaluated empirically during preliminary tests on synthetic data. In section R̈egnANN: varying learning parametersẅe show how the performance of the proposed method depends on the choice of the learning parameters.

Once the ensemble is trained, the topology of the gene regulatory network is obtained by applying a second procedure. Considering each gene in the network separately, we pass a value of 1 to the input neuron of the correspondent multilayer perceptron, consequently recording its output values. The continuous output values in the range $[-1,1]$ represent the expected normalized expression values for the other genes (its neighborhood). This procedure basically aims at verifying the correlation between the input gene and all the others: assuming the input gene maximally expressed (the value 1), an output value of (i.e.) 1 indicates that the correspondent gene will be also maximally expressed, thus indicating perfect correlation between the two genes. An output value of (i.e.) -1 indicates that the correspondent gene will be maximally under-expressed: perfect anti-correlation of the two genes. Thus, the continuous output values in the range $[-1,1]$ are interpretable in terms of positive correlation (>0), anti-correlation (<0) and no-correlation (0). By cycling this procedure through all the ensemble members in the regression system, we obtain N (one for each of the N genes in the network) vectors of length $N-1$ of continuos values in $[-1,1]$. The correlation matrix is obtained by correctly joining the N vectors. It is important to note that all the values of the diagonal of the adjacency matrix are equal to 0 by construction: this procedure does not allow discovering of gene self correlation (regulation) patterns, but only correlation patterns among different genes. Finally the adjacency matrix of the sought gene network is obtained by thresholding the correlation coefficients.

To improve the general efficiency of the algorithm and thus to allow a systematic comparison of its performance with the other gene network reverse engineering methods tested, we implemented the ANN based regression system using the GPGPU programming paradigm. A reference implementation of the RegnANN algorithm is available at http://sourceforge.net/projects/regnann/files/. The source code is distributed according to the GPLv3 license (open-source).

Alternative inference methods

As reference methods we select three alternative algorithms widely used in literature: ARACNE, CLR and KELLER.

KELLER is a kernel-reweighted logistic regression method [20] introduced for reverse engineering the dynamic interactions among genes based on the time series of their expression values. It estimates the neighborhood of each gene separately and then it joins the neighborhoods to form the overall network. The approach aims at reducing the network inference problem to a set of identical atomic optimizations. KELLER makes use of the l_1-regularized logistic regression algorithm and it operates modeling the distribution of interactions between genes as a binary pair-wise Markov Random Field.

With respect to other inference methodologies, KELLER adopt a fixed threshold to discretize the inferred adjacency matrix while it performs an optimization of the regularization weight lambda controlling the sparsity of the solution by maximizing a Bayesian Information Criterion (BIC). The authors apply a grid search on a selection of possible parameter values. In our work, we adopt the very same procedure: we use the same fixed discretization threshold for the binarization of the adjacency matrix, while we select the optimal solution by maximizing the BIC via a grid search for the optimal value of lambda (the very same value range used in [20]). This is done for each inference task.

As indicated in [20], the KELLER algorithm approximates the dynamic rewiring of the gene networks topology by borrowing "information across time by reweighting the observations from different time points and then treating them as if they were i.i.d. observations. Intuitively, the weighting should place more emphasis on observations at or near time point t with weights becoming smaller as the observations move further away from time point t. " Thus, this procedure relies in fact on a sequence of static topologies that do not differ greatly one another (one of the KELLER algorithmic assumptions is that the time-evolving network varies smoothly across time). In this work we consider only one fixed (static) topology for each inference task subsequently measuring the correctness of the result.

ARACNE is a general method able to address a wide range of network deconvolution problems - from transcriptional [8] to metabolic networks [10] - that was originally designed to scale up to the complexity of regulatory networks in mammalian cells. The method makes use of an information theoretic approach to eliminate the majority of indirect interactions inferred by co-expression methods. ARACNE removes the vast majority of indirect candidate interactions by applying a well-known information theoretic property: the data processing inequality [36].

Here we use the reference implementation of the algorithm provided in [37] with default value for the data processing inequality tolerance parameter. In Section "ARACNE: varying learning parameters" we explore the influence of the tolerance parameter on a sample testbed.

As many other methods, ARACNE relies on the definition of a threshold for the binarization of the adjacency matrix. In absence of a good heuristic for defining such threshold, on the synthetic data-sets we will adopt the area under the curve (AUC) as performance metric.

CLR is an extension of the relevance networks class of algorithms [9], which predicts regulations between transcription factors and genes by applying the mutual information score. CLR proposes an adaptive background correction step that is added to the estimation of mutual information. For each gene, the statistical likelihood of the mutual information score is computed within its network context. Then, for each transcription factor-target gene pair, the mutual information score is compared to the context likelihood of both the transcription factor and the target gene, and turned into a z-score. We adopt the reference implementation of the algorithm provided in [37]. As in the case of ARACNE, in absence of a good heuristic for defining a binarization threshold for the inference of the adjacency matrix, on the synthetic data-sets we will adopt the area under the curve (AUC) as performance metric.

Performance Metric

When the performance of a network inference method is evaluated, it is common practice to adopt two metrics: precision and recall. Recall indicates the fraction of true interactions correctly inferred by the algorithm, and it is estimated according to the following equation:

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

where TP indicates the fraction of *true positives*, while FN indicates the fraction of *false negatives*.

On the other hand, precision measures the fraction of true interactions among all inferred ones, and it is computed as:

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

where FP indicates the ratio of *false positives*.

In this work we adopt the Matthews Correlation Coefficients - MCC [27,28]: this is a measure that takes into account both true/false positives and true/false negatives and it is generally regarded to as a balanced measure, useful specially in the case of unbalanced classes (i.e.: different number of positive and negative examples).

The MCC is in essence a correlation coefficient between the observed and predicted binary classifications: it returns a value between -1 and $+1$. A coefficient value equal to $+1$ represents a perfect prediction, 0 indicates an average random prediction while -1 an inverse prediction [27,38]. In the context of network topology inference the observed class is the true network adjacency matrix, while the predicted class is the inferred one.

The Matthews Correlation Coefficient is calculated as:

$$\frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{5}$$

Recently MCC has also been used for comparing network topologies [28,39].

Data

We benchmark the reverse engineering algorithms here considered using both synthetic and biological data.

Synthetic Data

The synthetic data sets are obtained starting from an adjacency matrix describing the desired topology of the network. Here we consider two different network topologies: Barabasi-Albert [29] and Erdös-Rényi [30]. Network graphs are generated using the *igraph* extension package to the GNU R project for Statistical Computing [40]. Figure 20 shows two sample network topologies. In this work we focus our analysis on undirected and unweighted graphs: we are interested in estimating the structures of interaction between nodes/genes, rather than the detailed strength or the direction of these interactions. Thus, we consider only symmetric and discrete adjacency matrices, representing with a value of 1 the presence of a link between two nodes. A value equal to 0 in the adjacency matrix indicates no interaction.

Once the topology of the network is (randomly) generated, the output profiles of each node are generated according to two different approaches: the first one considers only linear correlation among selected genes (SLC), the second one is based on a gene network/expression simulator recently proposed to assess reverse engineering algorithms (GES [41]). In order to account for the intrinsic underdetermination of the task of network inference, we focus on synthetic data mimicking plausible submodules of larger networks: relatively small networks with a number of nodes

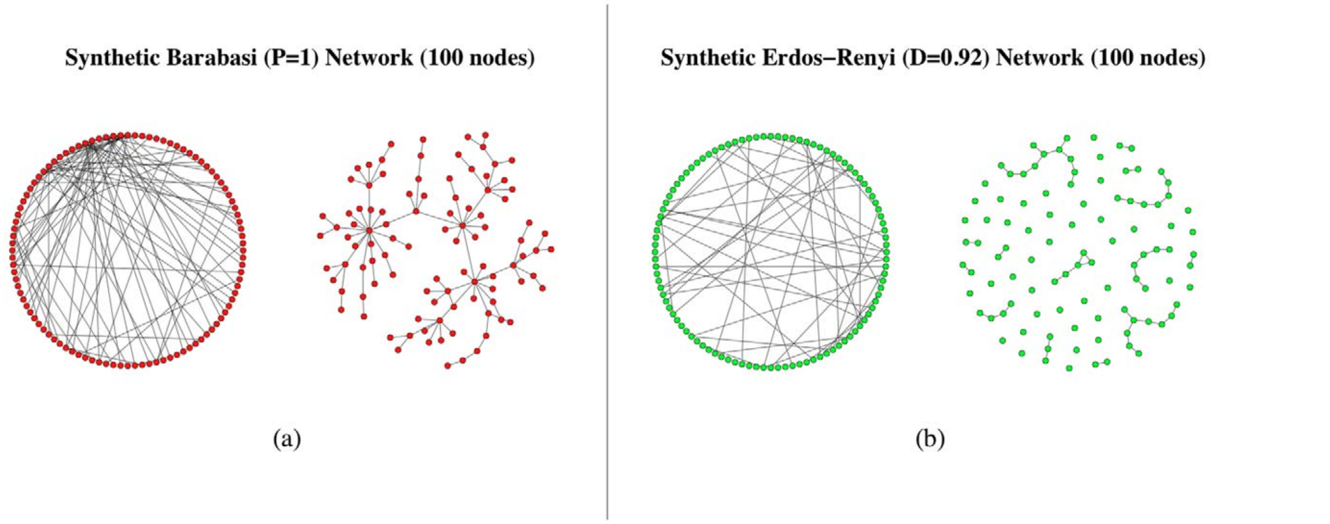

Synthetic Barabasi (P=1) Network (100 nodes)
(a)
Synthetic Erdos–Renyi (D=0.92) Network (100 nodes)
(b)

Figure 20. Sample network topologies: (a) Barabasi Network with 100 nodes (power-law exponent *P* equal to 1); (b) Erdös-Rényi network, 100 nodes and average degree (*D*) equal to 0.92.
doi:10.1371/journal.pone.0028646.g020

ranging between 50 and 200 and a number of expression profiles ranging from 10 to 400. Thus, we settle our analysis in a scenario of reduced search space/extended amount of independent information.

Simple Linear Correlation (SLC): similarly to the simulation of gene expression data presented in the supplementary material of [42], we consider a set of *seed* expressions (a matrix $M \times N$ - N genes which expression profiles are recoded M times - with values uniformly distributed in $[-1,1]$) and the desired topology expressed by the adjacency matrix $adjM$ ($N \times N$). The matrix $adjM$ contains only zeros and ones: a value of one indicates a link between the corresponding genes. The gene expression profiles (gep, a matrix $M \times N$) are calculated as:

$$gep = seed + seed \star adjM \tag{6}$$

where the symbol '$+$' indicates element-wise summation and the symbol '$\star$' indicates row-column matrix multiplication. With this method, the *seed* expression columns are linearly correlated (correlation equal to 1) with the columns of the same matrix as described by the discrete input adjacency matrix *adjM*.

Gene Expression Simulator (GES): this second methodology is based on a gene network simulator recently proposed to assess reverse engineering algorithms [41]. Given an input adjacency matrix, the network simulator uses fuzzy logic to represent interactions among the regulators of each gene and it adopts differential equations to generate continuous data. As in [8], we obtain synthetic expression values of each gene n ($n = 1, \ldots, N$) by simulating its dynamics until the expression value reaches its steady state. We obtain M different values for each gene by repeating the process M times and recording the expression value at steady state. The synthesis of each gene profile is randomly initialized by the simulator.

Escherichia coli Transcriptional Network

The task for the biological experiments is the inference of a few transcriptional subnetworks of the model organism *Escherichia coli* starting from a set of steady state gene expression data. The data are obtained from different sources and they consist of three different elements, namely the whole *Escherichia coli* transcriptional network, the set of the transcriptional subnetworks and the gene expression profiles to infer the subnetworks from. The *Escherichia coli* transcriptional network is extracted from the RegulonDB (http://regulondb.ccg.unam.mx/) database, version 6.4 (2010) and it consists of 3557 experimentally confirmed regulations between 1442 genes, amongst which 172 transcription factors. The 117 subnetworks are defined in [43]: in our experiments we use 7 of these subnetworks, with a number of genes ranging from 7 to 104. Information about number of genes and number of links for each subnetwork is reported in Table 4, Section Results. The expression data have been originally used in [9] and they consist of 445 *Escherichia coli* Affymetrix Antisense2 microarray expression profiles for 4345 genes, collected under different experimental conditions such as pH changes, growth phases, antibiotics, heat shock, varying oxygen concentrations and numerous genetic perturbations. MAS5 preprocessing is chosen among the available options (MAS5, RMA, gcRMA, DChip).

Data Discretization

A number of sources of noise can be introduced into the microarray measurements, e.g. during the stage of hybridization, digitization and normalization. Therefore, it is often preferred to consider only the qualitative level of gene expression rather than its actual value [20]: gene expression is modeled as either being up-regulated ($+1$) or down-regulated (-1) by comparing the given value to a threshold. For example, in [44] it is shown that binarizing gene expression data leads to classification outcomes very similar to the results obtained on real-valued data.

In this work we compute the discrete value of the expression for each of the N genes at each of the M steps as the sign of the difference of the expression values of the given gene at step m and step $m-1$.

Data Rescaling

Generally, when a scaling method is applied to the data, it is assumed that different sets of intensities differ by a constant global factor [31]. It may also happen that the rescaling is a necessary step due to the inference method adopted, as in the case of SVM (Support Vector Machine) or ANN (Artificial Neural Network) classification/regression.

In this work we test two different data rescaling methods:

- *linear rescaling*: each gene expression column-vector is linearly rescaled between $[-1,1]$;
- *statistical normalization*: each gene expression column-vector is rescaled such that its mean value is equal to 0 and the standard deviation equal to 1.

We consider gene expression matrices of dimension $M \times N$: N genes whose expression levels are recorded M times.

Experimenting with learning parameters

RegnANN: varying learning parameters. In this section we show how the tuning parameters of RegnANN impact its performance on a selected testbed: Barabasi networks with 100 nodes and SLC data generation. Accuracy scores (MCC) are calculated as mean of 10 iterations. Error bars are omitted for clarity. Gene expression profiles are rescaled linearly in $[-1,1]$.

Figure 21 summarizes the MCC score of RegnANN obtained varying training epochs for fixed momentum and learning rate values.

Figure 22 summarizes the MCC score of RegnANN obtained varying momentum and learning rate for fixed training epochs value.

The two set of figures indicate that the values for the learning parameters adopted in the evaluation of the performance of RegnANN (momentum = 0.1, learning rate = 0.01, training epochs = 1000) are chosen prudently.

ARACNE: varying learning parameters. In this section we explore the influence of the tolerance parameter (EPS) on a sample testbed. The reference implementation of ARACNE provided by [37] as R package [40] sets its value to 0.0.

We adopt the AUC_{MR} score (the AUC value for the MCC-Recall curve) as performance metric. AUC_{MR} values are calculated as mean of 10 iterations. Error bars are calculated as twice the standard deviation.

Figure 23 shows the mean AUC_{MR} (the AUC value for the MCC-Recall curve) varying the eps value in [0.0,1.0], the topology of the network and the number of nodes. Gene expression profiles are rescaled linearly in $[-1,1]$. We consider fixed data ratio equal to 100%.

The figure indicates that for Barabasi networks no statistically relevant difference in the performance of ARACNE is recoded varying the EPS parameter, e.g.: considering network size 100 nodes, with EPS value 0.0 ARACNE scores 0.36 ± 0.09; with EPS value 0.4 ARACNE scores 0.3 ± 0.1; with EPS value 0.8 ARACNE scores 0.4 ± 0.1; with EPS value 1.0 ARACNE scores 0.4 ± 0.1. In the case of Erdös-Rényi, setting the EPS parameter value to 0.4 is

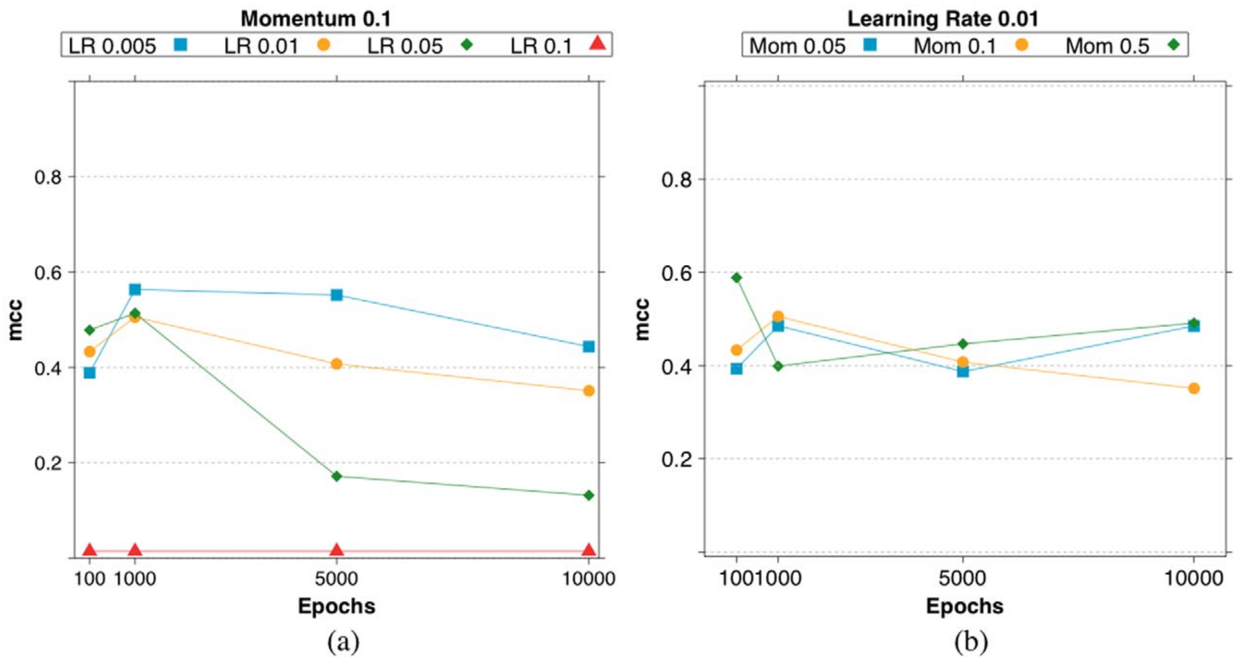


Figure 21. Accuracy (MCC) scores for RegnANN on synthetic Barabasi networks with 100 nodes and SLC data generation. (a) Fixed momentum (0.1), varying learning rate and training epochs. (b) Fixed learning rate (0.01), varying momentum and training epochs.
doi:10.1371/journal.pone.0028646.g021

detrimental for the mean accuracy of ARACNE, while all the other values tested result in equivalent performances, e.g.: considering network size 100 nodes, with EPS value 0.0 ARACNE scores 0.61 ± 0.04; with EPS value 0.4 ARACNE scores 0.51 ± 0.08; with EPS value 0.8 ARACNE scores 0.62 ± 0.03; with EPS value 1.0 ARACNE scores 0.62 ± 0.03.

The figure indicates that the value 0.0 is a sound default for the eps parameter.

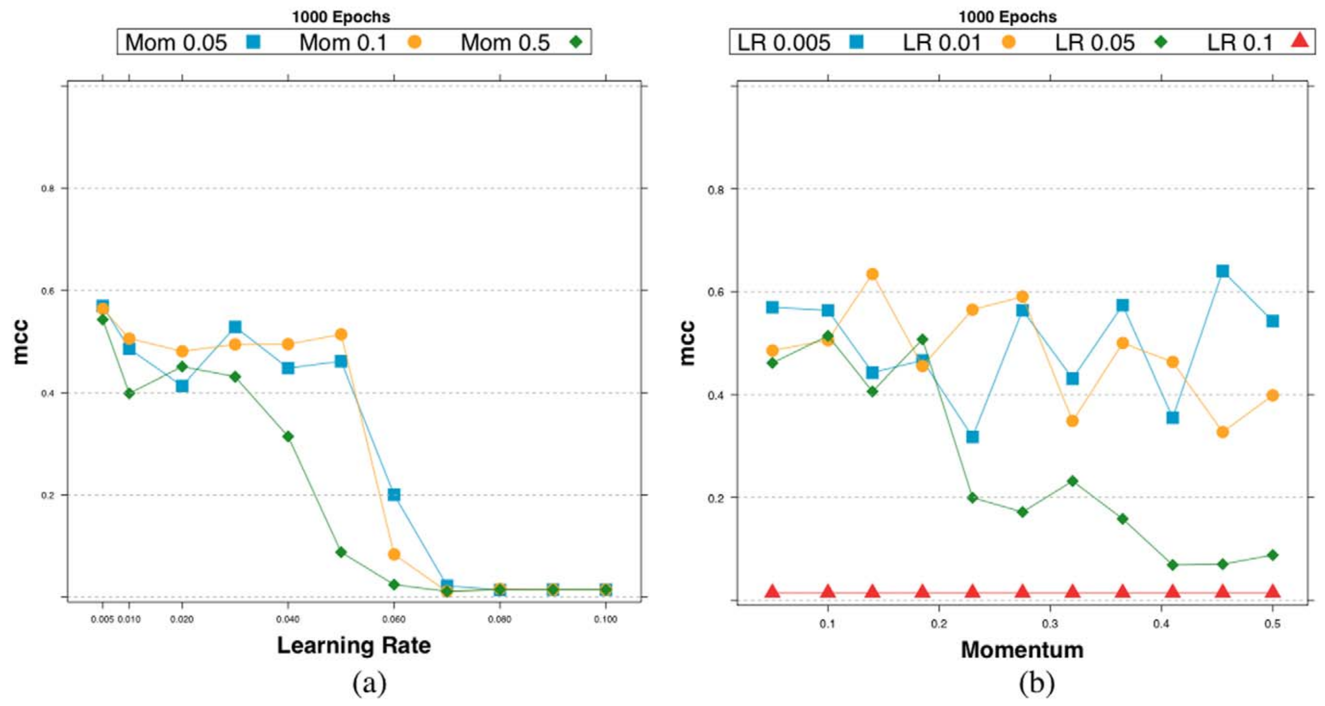


Figure 22. Accuracy (MCC) scores for RegnANN on synthetic Barabasi networks with 100 nodes, SLC data generation. We consider fixed training epochs (1000) while varying (a) learning rate and (b) momentum.
doi:10.1371/journal.pone.0028646.g022

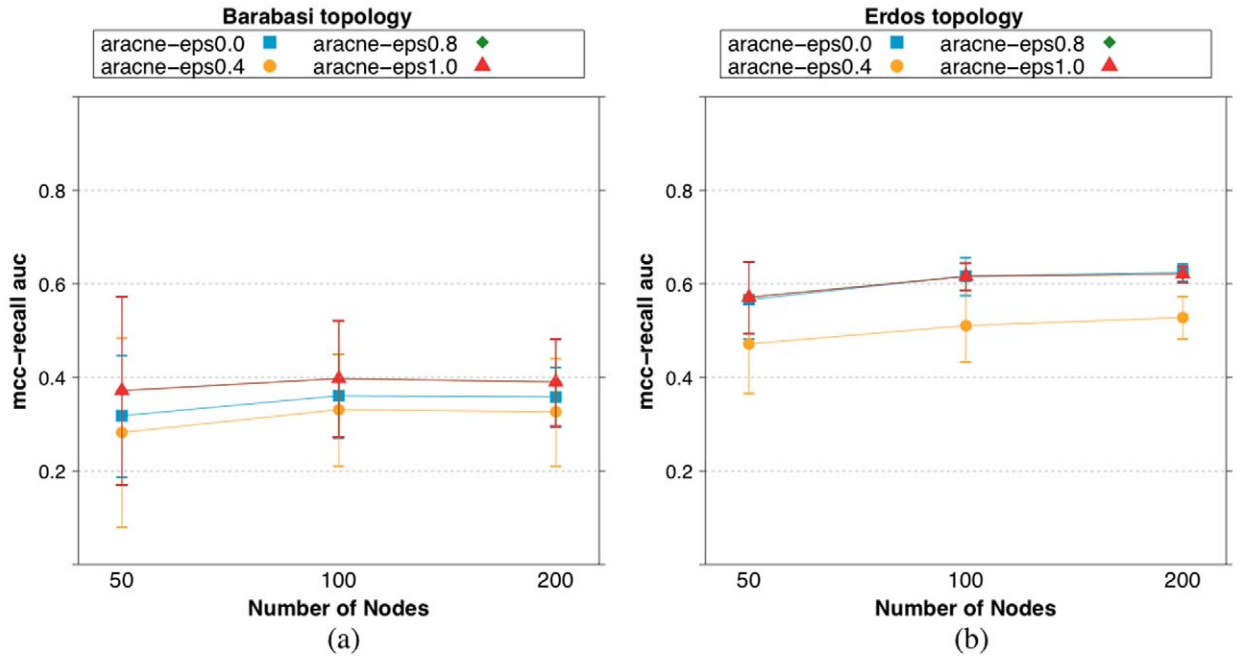


Figure 23. Mean AUC_{MR} varying the eps value, the topology of the network and the number of nodes. Gene expression profiles are rescaled linearly in $[-1,1]$. (a) Barabasi topology. (b) Erdös-Rényi networks.
doi:10.1371/journal.pone.0028646.g023

Acknowledgments

The authors want to thank the anonymous reviewers for their invaluable remarks and suggestions.

Author Contributions

Conceived and designed the experiments: MG RV GJ. Performed the experiments: MG RV GJ. Analyzed the data: MG RV GJ. Contributed reagents/materials/analysis tools: MG RV GJ. Wrote the paper: MG RV GJ.

References

1. Baralla A, Mentzen W, de la Fuente A (2009) Inferring Gene Networks: Dream or Nightmare? Ann NY Acad Sci 1158: 246–256.
2. De Smet R, Marchal K (2010) Advantages and limitations of current network inference methods. Nat Rev Microbiol 8: 717–729.
3. Krishnan A, Giuliani A, Tomita N (2007) Indeterminacy of reverse engineering of gene regulatory networks: The curse of gene elasticity. PLoS ONE 2: e562.
4. Altay G, Emmert-Streib F (2010) Revealing differences in gene network inference algorithms on the network level by ensemble methods. Bioinformatics 26: 1738–1744.
5. Marbach D, Prill R, Schaffter T, Mattiussi C, Floreano D, et al. (2010) Revealing strenghts and weaknesses of methods for gene network inference. PNAS 107: 6286–6291.
6. Eisen MB, Spellman PT, Brown PO, Botstein D (1998) Cluster analysis and display of genome-wide expression patterns. PNAS 95: 14863–14868.
7. Huynh-Thu VA, Irrthum A, Wehenkel L, Geurts P (2010) Inferring regulatory networks from expression data using tree-based methods. PLoS ONE 5: e12776.
8. Margolin A, Nemenman I, Basso K, Wiggins C, Stolovitzky G, et al. (2006) ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics 7: S7.
9. Faith J, Hayete B, Thaden J, Mogno I, Wierzbowski J, et al. (2007) Large-Scale Mapping and Validation of *Escherichia coli* Transcriptional Regulation from a Compendium of Expression Profiles. PLoS Biol 5: e8.
10. Nemenman I, Escola G, Hlavacek W, Unkefer P, Unkefer C, et al. (2007) Reconstruction of Metabolic Networks from High-Throughput Metabolite Profiling Data. Ann NY Acad Sci 1115: 102–115.
11. Altay G, Emmert-Streib F (2010) Inferring the conservative causal core of gene regulatory networks. BMC Syst Biol 4: 132.
12. Mordelet F, Vert JP (2008) SIRENE: supervised inference of regulatory networks. Bioinformatics 24: i76–i82.
13. Bansal M, Belcastro V, Ambesi-Impiombato A, di Bernardo D (2007) How to infer gene networks from expression profiles. Mol Syst Biol 122: 78.
14. Dimitrova E, Jarrah A, Laubenbacher R, Stigler B (2007) A Gröobner fan method for biochemical network modeling. In: Wang D, ed. Proceedings of ISSAC 2007. pp 122–126.
15. Kauffman S (1993) The Origins of Order: Self-Organization And Selection in Evolution. Oxford: OUP.
16. Friedman N, Linial M, Nachman I, Péer D (2000) Using Bayesian networks to analyze expression data. J Comput Biol 7: 601–620.
17. Markowetz F, Spang R (2007) Inferring cellular networks - a review. BMC Bioinformatics 8(S6): S5.
18. Karlebach G, Shamir R (2008) Modelling and analysis of gene regulatory networks. Nat Rev Mol Cell Biol 9: 770–780.
19. He F, Balling R, Zeng AP (2009) Reverse engineering and verification of gene networks: Principles, assumptions, and limitations of present methods and future perspectives. J Biotechnol 144: 190–203.
20. Song L, Kolar M, Xing E (2009) KELLER: estimating time-varying interactions between genes. Bioinformatics 25. pp i128–i136.
21. Lahabar S, Agrawal P, Narayanan P (2008) High Performance Pattern Recognition on GPU. In: Proceedings of NCVPRIPG 2008: pp 154–159.
22. Bailly-Bechet M, Braunstein A, Pagnani A, Weigt M, Zecchina R (2010) Inference of sparse combinatorial-control networks from gene-expression data: a message passing approach. BMC Bioinformatics 11: 355.
23. Braunstein A, Pagnani A, Weigt M, Zecchina R (2008) Inference algorithms for gene networks: a statistical mechanics analysis. Journal of Statistical Mechanics P12001.
24. Braunstein A, Pagnani A, Weigt M, Zecchina R (2008) Gene-network inference by message passing. Journal of Physics: Conference Series 95: 012016.
25. Keedwell E, Narayanan A (2005) Discovering gene networks with a neural-genetic hybrid. IEEE/ACM Trans Comput Biol Bioinformatics 2: 231–242.
26. Zhang Y, Xuan J, de los Reyes B, Clarke R, Ressom H (2008) Network motif-based identification of transcription factor-target gene relationships by integrating multi-source biological data. BMC Bioinformatics 9: 203.
27. Matthews B (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta 405: 442–451.
28. Stokic D, Hanel R, Thurner S (2009) A fast and efficient gene-network reconstruction method from multiple over-expression experiments. BMC Bioinformatics 10: 253.
29. Barabasi A, Albert R (1999) Emergence of scaling in random networks. Science 286: 509–512.
30. Erdös P, Renyi A (1959) On Random Graphs. Publ Math Debrecen 6: 290–297.
31. Steinhoff C, Vingron M (2006) Normalization and quantification of differential expression in gene expression microarrays. Brief Bioinform 7: 166–177.

32. Peregrin-Alvarez J, Xiong X, Su C, Parkinson J (2009) The Modular Organization of Protein Interactions in Escherichia coli. PLOS Comput Biol 5: e1000523.
33. Bishop C (1995) Neural Networks for Pattern Recognition. New York: OUP.
34. Neal R (1996) Bayesian Learning for Neural Networks (Lecture Notes in Statistics). New York: Springer-Verlag.
35. Neal R (1995) Bayesian learning for neural networks. Ph.D. thesis, Department of Computer Science, University of Toronto.
36. Cover T, Thomas J (1991) Elements of information theory. New York: Wiley-Interscience.
37. Meyer P, Lafitte F, Bontempi G (2008) minet: A r/bioconductor package for inferring large transcriptional networks using mutual information. BMC Bioinformatics 9: 461.
38. Baldi P, Brunak S, Chauvin Y, Andersen C, Nielsen H (2000) Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 16: 412–424.
39. Supper J, Spieth C, Zell A (2007) Reconstructing linear gene regulatory networks. In: Proc of EvoBIO2007. Springer-Verlag. pp 270–279.
40. R Development Core Team (2011) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org. ISBN 3–900051–07–0. Accessed 2011 Nov 26.
41. Di Camillo B, Toffolo G, Cobelli C (2009) A Gene Network Simulator to Assess Reverse Engineering Algorithms. Ann NY Acad Sci 1158.
42. Langfelder P, Horvath S (2007) Eigengene networks for studying the relationships between coexpression modules. BMC Syst Biol 1: 1–54.
43. Marr C, Theis F, Liebovitch L, Hütt MT (2010) Patterns of Subnet Usage Reveal Distinct Scales of Regulation in the Transcriptional Regulatory Network of Escherichia coli. PLoS Comput Biol 6: e1000836.
44. Tuna S, Niranjan M (2009) Cross-Platform Analysis with Binarized Gene Expression Data. In: PRIB. pp 439–449.