scientific reports

OPEN

Learning complex dependency structure of gene regulatory networks from high dimensional microarray data with Gaussian Bayesian networks

Catharina E. Graafland & José M. Gutiérrez

Reconstruction of Gene Regulatory Networks (GRNs) of gene expression data with Probabilistic Network Models (PNMs) is an open problem. Gene expression datasets consist of thousand of genes with relatively small sample sizes (i.e. are large-*p*-small-*n*). Moreover, dependencies of various orders coexist in the datasets. On the one hand transcription factor encoding genes act like hubs and regulate target genes, on the other hand target genes show local dependencies. In the field of Undirected Network Models (UNMs)—a subclass of PNMs—the Glasso algorithm has been proposed to deal with high dimensional microarray datasets forcing sparsity. To overcome the problem of the complex structure of interactions, modifications of the default Glasso algorithm have been developed that integrate the expected dependency structure in the UNMs beforehand. In this work we advocate the use of a simple score-based Hill Climbing algorithm (HC) that learns Gaussian Bayesian networks leaning on directed acyclic graphs. We compare HC with Glasso and variants in the UNM framework based on their capability to reconstruct GRNs from microarray data from the benchmarking synthetic dataset from the DREAM5 challenge and from real-world data from the *Escherichia coli* genome. We conclude that dependencies in complex data are learned best by the HC algorithm, presenting them most accurately and efficiently, simultaneously modelling strong local and weaker but significant global connections coexisting in the gene expression dataset. The HC algorithm adapts intrinsically to the complex dependency structure of the dataset, without forcing a specific structure in advance.

The reconstruction of Gene Regulatory Networks (GRNs) of gene expression data is an open problem and has attracted great deal of interest for decades. GRNs model interaction structure of genes in a network and are important to understand the biological functions in living organisms as well as the regulation of diseases in them. The increasing availability and improved systematic storage of gene expression data obtained with DNA microarrays[1] revolutionized the incorporation of mathematical and computational models to model GRNs in the past two decades[2–4]. Mathematical models range from more complex models relying on sets of differential equations that directly describe dynamic changes in GRNs[5,6] to simpler models that describe GRNs building on a graph presentation (graphical models or network models). The latter attract attention due to their capacity of visualization of the complex interaction structure of microarray data in a graph. The most simple and widely used example of a graphical model are pairwise correlation networks[7], but lately more advanced Probabilistic Network Models (PNMs) that make use of machine learning algorithms to comprehensively model conditional and/or partial dependencies have gained in popularity.

One of the first applications of PNMs to reconstruct GRNs was done two decades ago in a seminal work[8] using Bayesian Networks (BNs)—a subclass of PNMs relying on directed acyclic graphs— to analyze *S. cerevisiae* cell-cycle measurements using two approaches: multinomial BNs and Gaussian BNs. The former relies on discrete data and therefore requires the discretization of continuous microarray data; the latter can directly handle continuous microarray data assuming a multivariate Gaussian distribution function. However, the subsequent use of BNs in this field has mainly focused on adapting and improving algorithms for multinomial BNs[8–12], while the application of Gaussian BNs have been left untreated (with some exceptions in a comparison study[13]). The

Instituto de Física de Cantabria, CSIC-Universidad de Cantabria, Avenida de Los Castros, 39005 Santander, Spain. email: catharina.graafland@unican.es

Gaussian case though has been widely investigated in the subclass of PNMs that relies on undirected graphs, i.e., Undirected probabilistic Network Models (UNMs, also known as undirected Markov random fields or pairwise Markov networks)[14]. The most outstanding algorithm of the past decade to learn Gaussian UNMs was first applied on cell-signaling data from proteomics[15]. This algorithm is called the Graphical lasso (Glasso) and estimates the inverse covariance matrix, also concentration or precision matrix, of the Gaussian distribution function. It was successfully applied in the high dimensional setting of gene interaction networks[16,17] and in network clustering in bioinformatics[18], but also in psychology networks[19], risk management[20,21], and climate[22].

The application of Gaussian UNMs to continuous microarray data has faced two main challenges. First, the large-p-small-n nature of the data. Microarray experiments contain expression levels of thousands of genes at the same time, but have relative small sample size. Glasso deals with high dimensional data by imposing sparsity on the precision matrix and thus on the undirected graph. Secondly, degrees of interaction among genes in gene networks are not uniform, but higher order. In GRNs, Transcription Factor encoding genes (TFs) regulate the expression of many target genes. However, only a few genes encode TFs, which can be seen as gene hubs[23,24] that confer GRNs with scale-free network characteristics, as suggested by previous research on the degree distribution of GRNs[25]. Numerous modifications of the Glasso algorithm distribution have been proposed in recent years to better model this complex structure of interactions in microarray data (and other real-world datasets). Two of them were developed especially for gene regulatory networks that are expected to be scale-free[26] or to consist of hubs[27]. We will refer to them as the Scale-Free Glasso (SFGlasso) and the Hub Glasso (HGlasso). Both modifications of the Glasso algorithm force beforehand the expected structure in the estimation of the precision matrix and were shown to outperform Glasso on simulated data that had these characteristics.

As an alternative, in this paper we build on the initial work using Bayesian Networks[8] and propose the use of Gaussian Bayesian Networks (GBNs) to address the complex structure problem in GRNs. Our motivation stems from a recent paper showing the suitability of GBNs for modelling the complex (hub) interaction structure in complex high dimensional data, revealing the underlying probabilistic backbone[28]. Here we use the score-based Hill Climbing (HC) GBN learning algorithm used in that work, which shows good performance when compared with other GBN learning algorithms[29], and assumes no specific structure beforehand. We compare HC with the default approach for learning sparse UNMs, the Graphical lasso (Glasso) and with its modifications SFGlasso and HGlasso using two different datasets. On the one hand, we use simulated data from an in silico gene regulatory network allowing for a comprehensive evaluation of the algorithm. On the other hand, we use (the most-up-to date version of) the dataset[30] from the *Escherichia coli* (*E. coli*) genome, which is entitled to be the best known/encoded organism on earth and from which information on TFs is well documented.

This simultaneous study and intercomparison of direct and undirect structure learning algorithms for high dimensional complex data is, to our knowledge, new. First, we place the algorithms in the broader perspective of PNMs and analyse their learning method, the statistical criterion and form of presentation (directed or undirected network). We then evaluate the algorithms in terms of accuracy and data explanation capabilities, paying attention to the compactness and sparsity of the learned networks and to the truthful balancing of dependencies of different orders.

Results

BNs and PNs in the general framework of Gaussian probabilistic network models. Gaussian probabilistic network models[31] consist of graph and parameterset. The parameterset fully determines the associated Gaussian Joint Probability Density (JPD) function. The type of parameterset determines the presentationform of the graphical model, i.e. the 'meaning' of the edges in the graph. A Gaussian JPD function can be determined by different types of parametersets such as the covariance matrix ($\mathbf{\Sigma}$), its inverse, the precision or concentration matrix ($\mathbf{\Sigma}^{-1}$), or linear regression coefficients in combination with local variation coefficients ($\boldsymbol{\beta}, \boldsymbol{\nu}$). Figure 1 shows two learning methods and their associated presentationforms of the Gaussian JPD function: Precision Networks or pairwise Markov networks (PN ($\mathbf{\Sigma}^{-1}$)) and Bayesian Networks (BN ($\boldsymbol{\beta}, \boldsymbol{\nu}$)). In this study we leave out the analysis of correlation networks that rely on the ($\mathbf{\Sigma}$) presentation generally generated from a thresholded sample correlation matrix, because they cannot properly regulate high dimensional data. We refer the interested reader to Graafland et al.[28] for an extensive comparison study between BNs and correlation networks. The representations of the Gaussian JPD function in terms of ($\mathbf{\Sigma}^{-1}$) and ($\boldsymbol{\beta}, \boldsymbol{\nu}$) are described in "Methods" section "Gaussian probabilistic network models" and subsections "Probabilistic PN models" and "Probabilistic BN models".

As illustrated in Fig. 1, the algorithms under subject of study, Glasso (and modifications) and Hill climbing, differ in the parameterset (and hence the presentationform) they have as an objective to learn. They do however coincide on their machine learning spirit; both algorithms attempt to find a parameterset that optimizes a score function in which a statistical criterion (penalization) is integrated. The score functions and their statistical criteria are described in more detail in "Methods" sections "Learning PN structure from data" and "Learning BN structure from data". Learning methods (the steps in the algorithms) are chosen to efficiently execute this optimization task and are described in "Methods" section "Learning with hill climbing and glasso". In the case of Hill Climbing the search space is restricted to that of Directed Acyclic Graphs (DAGs) and in the case of Glasso to that of symmetric positive definite precision matrices, which, converted to binary format, are translated into undirected graphs. These constraints procure that the edges in the networks respectively are associated with the regression coefficients $\boldsymbol{\beta}$ in the ($\boldsymbol{\beta}, \boldsymbol{\nu}$) parametrization of the JPD function and with the entries of the precision matrix $\mathbf{\Sigma}^{-1}$ in the ($\mathbf{\Sigma}^{-1}$) parametrization.

Once learned, a PNM is in general not bound to its initial parametrization/presentationform. At some cost, an analytical transformation of the parameterset can transform the presentationform of the PNM. In Fig. 1 the purple arrow represents a direct transformation between parametersets from BNs to PNs for which an analytical

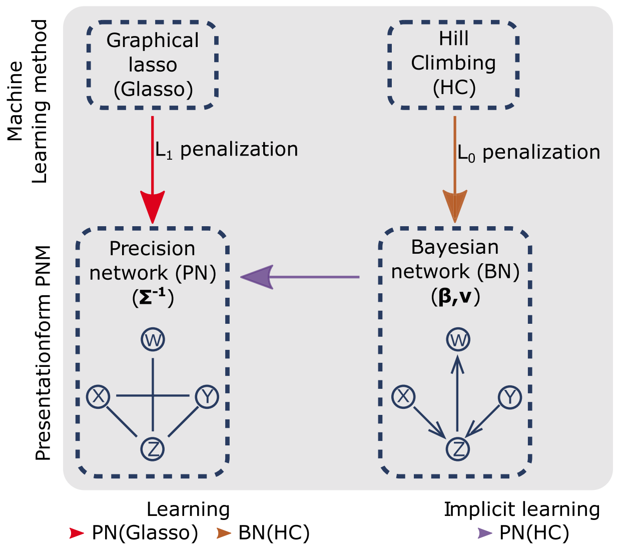

Figure 1. Schematic illustration of the presentationforms of Gaussian probability density function P in terms of conditional parameters ($\boldsymbol{\beta}, \boldsymbol{\nu}$) and precision matrix ($\boldsymbol{\Sigma}^{-1}$), associated with Bayesian Networks (BNs) and Precision Networks (PNs), respectively. Brown and red arrows refer to the associated learning algorithms (brown; score-based Hill Climbing algorithm, red; Graphical lasso) together with the type of integrated statistical criterion (l_0 or l_1 penalization). The purple arrow refers to the analytical transformation of the parameterset from BNs (learned with Hill Climbing) to PNs.

formula exists. The analytical transformation is described in "Methods" section "Transformation of probabilistic BN model to probabilistic PN model". The transformation makes the initial BN loosing some information on its independency structure (see how the graphs of BNs and PNs encode independence statements about the JPD function in "Methods" section "Dependencies in BN and PN structure"), but allows us to compare Hill Climbing and Graphical lasso under the same UNM framework.

Data, reference networks and algorithm settings. In this study we use two gene expression datasets, a synthetic benchmark dataset and a real-world dataset. The former is a gene expression dataset generated from an in silico network provided in one of the DREAM (Dialogue on Reverse Engineering Assessment and Methods) challenges (https://dreamchallenges.org). DREAM is a community-driven initiative to promote open, reproducible, and collaborative scientific research in biomedicine, supplying challenges that address a broad set of questions in healthcare and the life sciences. A significant part of the challenges is oriented to train machine learning models that could predict gene expression from sequences. The in silico network in the DREAM5 challenge consists of a total amount of 1548 genes of which 178 genes encode TFs and count as the origin of all transcriptional regulation interactions. Since all interactions go from TF encoding genes to target genes, the in silico network is by nature a directed network consisting of 4012 edges (unique interactions) without self-loops. This gold standard was used to generate a simulated gene expression dataset (with sample size $n = 805$) using the Gene Net Weaver[32], including random expression levels for 95 additional genes (17 decoy TFs and 78 decoy target genes).

The real-world dataset is constructed from two public available datasets for the *Escherichia coli* (*E. coli*) genome. On the one hand we use the *E. coli* microarray dataset that is available from the Many Microbe Microarrays database (M3D)[1]. This dataset contains the uniformly normalized expression levels of 4297 genes in *E. coli* measured under 466 experimental conditions using Affymetrix arrays. Expression levels from the same experimental conditions are averaged and their mean expression provides one of the 466 sample points in the dataset. In particular we use the latest version at this moment: E_coli_v4_Build_6. On the other hand, a reference network to evaluate the learned PNMs is constructed from a dataset containing evidence about transcriptional regulations (i.e. interactions that arise from TF encoding genes and their target genes). For the *E. coli* genome, a number of studies generated information on transcriptional regulations. The Regulon Data Base (RegulonDB) is the primary database[33] in which this information is gathered. We use the most complete file in the RegulonDB called network_tf_gene.txt.

We quire the algorithms to construct networks using only the expression levels in M3D of the 1683 genes from which evidence is reported in the RegulonDB. The known interactions in the RegulonDB will then really act like a reference network that enables us to evaluate the accuracy of the learned GRNs from the microarray data. A similar strategy is applied in[30]. The RegulonDB network has similar characteristics as the in silico network in DREAM5. However, in contrast to the in silico network, the RegulonDB network is not 'perfect', i.e. there are 'false negatives' in the gold standard due to undiscovered relationships. The 1683 genes in the reduced M3D dataset contain a total of 173 TF encoding genes of which 172 count as the origin of all transcriptional regulation interactions in the reference network. Together they are good for 3381 unique interactions without

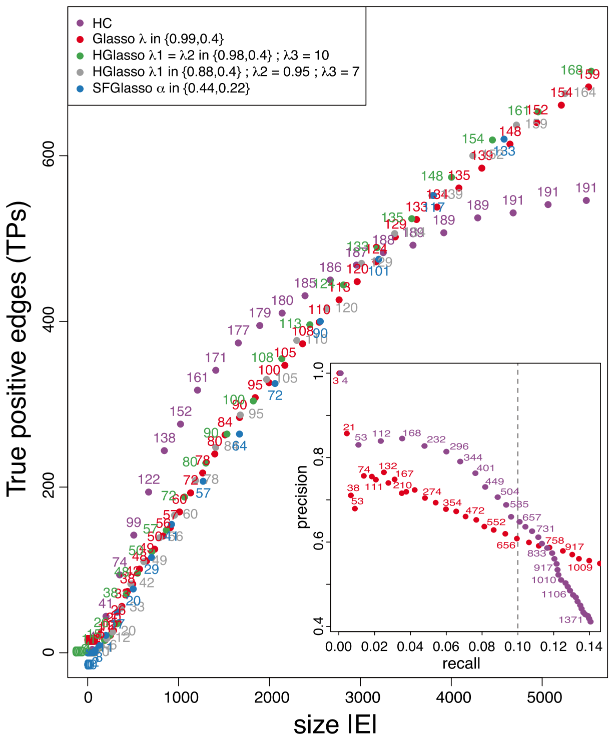

Figure 2. True positives (dots) and amount of transcription factors (numbers) w.r.t. the networksize |E| as encountered from the in silico dataset by HC (purple), Glasso with $\lambda \in \{0.98, 0.6\}$ (red), HGlasso with $\lambda_1 = \lambda_2 \in \{0.98, 0.6\}, \lambda_3 = 10$ (green), HGlasso with $\lambda_1 = 0.9, \lambda_2 \in \{0.61, 0.55\}, \lambda_3 = 14$ (orange), HGlasso with $\lambda_1 \in \{0.98, 0.6\}, \lambda_2 = 0.95, \lambda_3 = 7$ (grey), and SFGlasso with $\alpha \in \{0.43, 0.3\}$ (blue). The inset shows the precision vs the recall values for HC and Glasso with numbers indicating the network size; note that the the 0.1 recall value is indicated with a dashed line.

self-loops. As interactions go from TF encoding genes to target genes, the reference RegulonDB network is by nature a directed network.

The number of edges that is needed to construct a PNM from the microarray dataset plays an important role in the quality and the practical possibilities of a structure learning algorithm. With this in mind we generate networks of different sizes for all algorithms. To vary the amount of edges |E| we vary the initial parameters λ, and $\lambda_1, \lambda_2, \lambda_3$, and α for Glasso, HGlasso, and SFGlasso, respectively (see "Methods" section "Learning PN structure from data"). For Hill Climbing we obtain networks of different amount of iterations while using the standard BIC score (an action that give similar results as application of the BIC$_\gamma$ score and varying the parameter γ[29]). The directed BNs learned by Hill Climbing are transformed to UNMs for evaluation. This process, described in "Methods" section "Transformation of probabilistic BN model to probabilistic PN model", has a small penalty (extra, unnecessary parameters/edges will be added), but allows for a fair comparison with other algorithms (SF-, H- and Glasso) as the transformed edges will encode the same type of parameters.

Results for in silico benchmark network in DREAM5. We first assess the performance of the algorithms using the amount of True Positive edges (TPs) according to the benchmark in silico network. To this aim, an undirected edge in the learned network is a TP if there exists an associated directed edge in the in silico network. Moreover, the amount of connected Transcription Factors (TFs) in the estimated network is measured as an indication of the hub structure of the learned networks. These measures are analyzed as a function of the network size |E| to assess the capability of the methods to produce accurate, though compact and sparse networks.

Figure 2 shows the results on TPs and TFs with respect to the number of total estimated edges |E| for networks learned with the four algorithms. The dots indicate the amount of TPs and the dot labels, in the form of numbers, indicate the amount of TFs found in the corresponding network. This figure shows that the UNMs obtained with HC exhibits the best results (highest amount of TPs and TFs) for sparse networks up to 3000 edges. The biggest gain of TPs with respect to the amount of edges |E| occurs in the range from 500 to 1500 edges. In this range also the amount of recovered TFs is highest (more than 10 new TFs per 100 added edges). This simultaneous increase of TPs and TFs is intuitive as all 4012 edges in the in silico network consist of at least one TF. The standard Glasso algorithm recovers less TP edges than HC in sparse networks, with recovering rates particularly lower than HC in the range of 500–1500 edges; this could be related to the fact that in this range Glasso only recovers between 1 and 5 new TFs per 100 edges. In bigger networks consisting of more than 3000 edges the amount of TPs exceeds HC values. Note that all Glasso variants exhibit a similar performance as the standard Glasso. However, their search strategy is clearly different: the TPs of SFGlasso are originated by less TFs than Glasso, whereas the TPs of HGlasso are originated by more TFs than Glasso.

In addition to this analysis, we also evaluate learned networks using their precision and recall. The precision of a network shows how many of the predicted TF–target-gene links (edges in the learned network) are true positives (relative to all predicted positives), whereas the recall shows the rate of TPs relative to the total amount of positives. The use of the Precision-Recall curve (PR-curve) is in concordance with the methodology used in DREAM5 (see "Methods" section "Evaluation measures" for the formal definition of these measures). To benchmark our results in the framework of the DREAM5 challenge, we eliminated all links between two target-genes from the networks' edgelist before calculating their precision and recall. Note that participants in DREAM5 were provided with the names of the TFs in the set of genes and algorithms in DREAM5 were queried to search only for TF–target-gene relationships in the gene expression data; in this work we quire the algorithms to learn PNMs consisting of undirected edges in which, despite TF–target-gene relationships, also target-gene–target-gene dependencies are included. These edges are found in gene expression data due the existence of co-regulated genes by the same TFs[2]. The inset of Fig. 2 displays the PR-curve for the HC and Glasso algorithms. Each network in the curve is labeled with its associated amount of retrieved (predicted positives) TF–target-gene relationships. The figure shows a trade-off between precision and recall for all learned networks. For a recall below 10%, HC finds sparse networks of high precision (i.e. much qualitative information in sparse networks). On the other hand, Glasso finds networks of lower precision for a recall below 10%, but shows a steadier decay in precision for recall above 10%. Variants of Glasso behave equally to Glasso and are not displayed in the inset.

Although many GRN inference algorithms that participated in DREAM5 used (lasso) regression techniques, none of these methods are comparable to Glasso and HC algorithms used in this work, because they mainly focus on the regression of TFs on individual target genes. HC and Glasso perform in the top 15 network inference methods used in DREAM5 for recall values below 10% as judged by Fig. S3a of Supplementary Note 4[2]. This benchmarking is not perfect, as our algorithms did not receive information about the TFs in advance and are therefore at a disadvantage compared to the algorithms that participated in DREAM5.

Results for real-world gene expression data. The evaluation results for the in silico network highlight HC as an promising exploratory algorithm (finding sparse networks of high precision for recalls below 10 per cent), whereas Glasso and variants seem to have less exploratory ability (exhibiting lower precision for sparse networks) but to perform more steady after an initial exploration phase (medium precision for networks of growing size and higher recall). In this section we evaluate the algorithms on real-world M3D data using evaluation measures chosen in alignment with the nature of real-world datasets and the focus of this work: complex gene expression datasets of low sample size without background knowledge on which we want to chart the exploratory capability of the algorithms. We do not consider false positives in this case (and thus do not consider the PR-curve) since the networks are learned from gene expression data that also contain target-gene–target-gene dependencies that might not be in conflict with the evidence in the true network due to the existence of genes co-regulated by the same TFs. As these dependencies are not reported in the true network, they would be regarded as false positives, although they might be correct. Moreover, in the case of real-world data, the algorithms may discover new TF–target-gene interactions that are currently missing in the gold standard. Again, these dependencies would be regarded as false positives. This problem has been already discussed in the literature[2] suggesting to use the false positive and true negative measures with caution for this reason.

Similarly to the previous analysis for the in silico dataset, Fig. 3 shows the results on TPs and TFs for the RegulonDB with respect to the number of total estimated edges $|E|$ in the network for the four algorithms. Dots indicate the amount of TPs and the dot labels, in the form of numbers, indicate the amount of TFs found in the belonging network. Figure 3 shows that the UNMs obtained with HC contain the highest amount of TPs and TFs for sparse networks up to 5000 edges. The biggest gain of TPs with respect to the amount of edges $|E|$ occurs in the range from 1000 to 1800 edges. In this range also the amount of recovered TFs is highest (around 20 new TFs per 100 added edges). The standard Glasso recovers less TP edges than HC in sparse networks. The recovering rate is especially low with respect to HC in the range of 1000–1800 edges and is probably related to the fact that in this range Glasso only recovers between 1 and 2 new TFs per 100 edges. Similarly to the case of the in silico dataset, the amount of TPs found with Glasso exceeds HC values for big networks (consisting of more than 5000 edges). In the discussion we consider in more detail the strategy of Glasso and HC to discover TPs.

In contrast to the in silico network, Glasso and variants discover relatively few TFs in sparse networks. Moreover, the Glasso variants do not all perform equally well with respect to the evidence of the RegulonDB network. The performance of HGlasso (and whether it can outperform Glasso or not) depends on the network size and the combination of the values of the parameters λ_1, λ_2, and λ_3. Take for example the combination $\lambda_1 = 0.9$, λ_2 = variable, and $\lambda_3 = 14$ (orange dots). These networks consist of more TFs than standard Glasso networks, but less TPs are found. The impact of the relaxed penalty term in the score function for hubs is thus visible, but not realized through the caption of more TPs by the HGlasso algorithm. The best results of HGlasso are obtained with λ_1 = variable, $\lambda_2 = 0.95$, and $\lambda_3 = 7$ (grey dots); medium networks (up from 3000 edges) with these parameters not only consist of more TFs, but also outperform the standard Glasso on TPs. Finally, we see that SFGlasso performs worse than the Glasso and HGlasso on both the amount of TPs and TFs. The discovery rate of TFs is slow and, in case of real-world data, this results in few added TPs. It seems that the power-law degree distribution imposed by the SFGlasso algorithm does not fit well to the dependency structure determined by TFs in the M3D microarray data.

The difference in sample size (n = 805 for in silico and n = 466 for RegulonDB) and/or the difference in underlying dependency structure of the in silico and RegulonDB could have caused the differences in behaviour of Glasso algorithms on the two datasets. Note that, in the end, data is our principle resource for analysing new

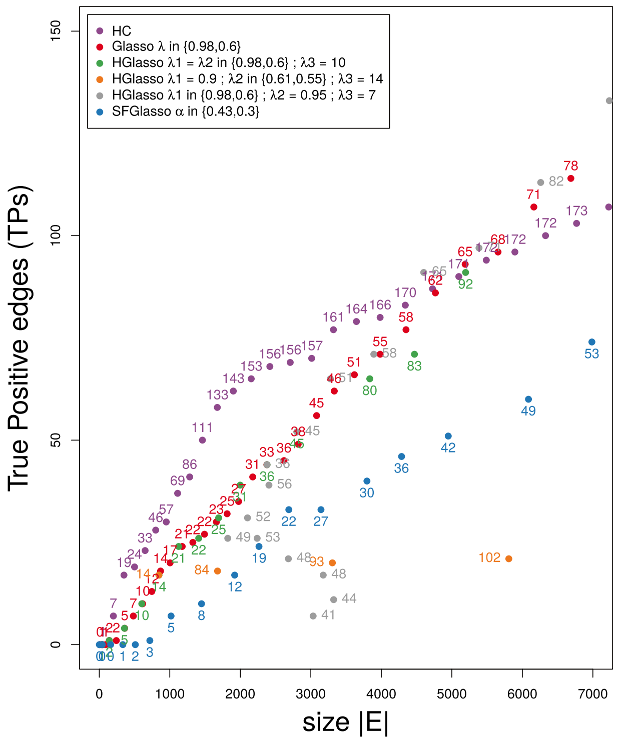

Figure 3. True positives (dots) and amount of transcription factors (numbers) w.r.t. the networksize |E| as encountered from the M3D microarray dataset by HC (purple), Glasso with $\lambda \in \{0.98, 0.6\}$ (red), HGlasso with $\lambda_1 = \lambda_2 \in \{0.98, 0.6\}$, $\lambda_3 = 10$ (green), HGlasso with $\lambda_1 = 0.9$, $\lambda_2 \in \{0.61, 0.55\}$, $\lambda_3 = 14$ (orange), HGlasso with $\lambda_1 \in \{0.98, 0.6\}$, $\lambda_2 = 0.95$, $\lambda_3 = 7$ (grey), and SFGlasso with $\alpha \in \{0.43, 0.3\}$ (blue).

unknown gene regulatory networks. This type of resource brings forward the need for computational methods that can extract a generalizable model from (a single) realisation(s) and, at the same time, faithfully reveal the various emergent phenomena like co-expression, that are relevant in the system.

To chart networks on their data information content, in the next, we implement a simple probabilistic evaluation measure that appreciates faithfulness of algorithms to observational data sources. We measure the relative probabilistic accuracy of a single learned UNM with respect to another learned UNM by their difference in log-likelihood. The log-likelihood quantifies how well the model explains the data in terms of the likelihood of the available data given the particular probabilistic model (see "Methods" section "Log-likelihood definition and calculation"). Figure 4 displays the log-likelihood of the networks versus the amount of total links in the networks. In the case of Gaussian PNMs the number of parameters in the probabilistic model grows linearly in function of the amount of edges. Hence, the log-likelihood relative to the number of edges of Gaussian PNMs also provides information on models' parsimony. For easy comparison with Fig. 3, we again placed the number of integrated TFs as dotlabels.

Figure 4 shows that the log-likelihood curve of HC rapidly improves until 2000 edges and establishes between 2000 and 7000 edges. Note that the same is true for the number of found TFs, growing rapidly until 2000 edges, reaching 153 TFs, and than slowing down until 7000 edges, reaching the total amount of 173 TFs. The true positives curve in Fig. 3 also begins to flatten around 2000 edges. Hence, in the case of HC, the validation measures TPs and TFs can be directly associated with the amount of information that is extracted from the microarray data and captured in the network.

In the case of H-, SF-, and Glasso the relation between log-likelihood and evaluation measures is more complex. The Glasso log-likelihood curve remains far apart from the HC curve for small and medium networks. From Fig. 3 we learn that Glasso values of evaluation measures finally approach HC values when including more and more edges. Figure 4 illustrates that this is not true (or in any case not true for regularized networks) for log-likelihood values. In the discussion we go into the relation between log-likelihood and TPs for Glasso and HC.

All HGlasso networks score better on log-likelihood values than the standard Glasso. For HGlasso networks that improve Glasso networks on TPs (e.g., HGlasso networks up from 3000 edges with $\lambda_2 = 0.95$ and $\lambda_3 = 10$; grey dots) this betterment is even more pronounced. Still, however, HC log-likelihood values are not reached neither at small nor at medium edge size by the adapted HGlasso algorithm. Similar to the case of TPs, SFGlasso scores worse than Glasso on log-likelihood values for small to medium networksizes. We do observe that the more edges are added to the SFGlasso network, the closer log-likelihood values become to Glasso values.

Illustration of network structure. To illustrate the above results on the performance of the methods, we zoom into a part of the reference RegulonDB network that contains the two TF encoding genes hyfR and fhlA—

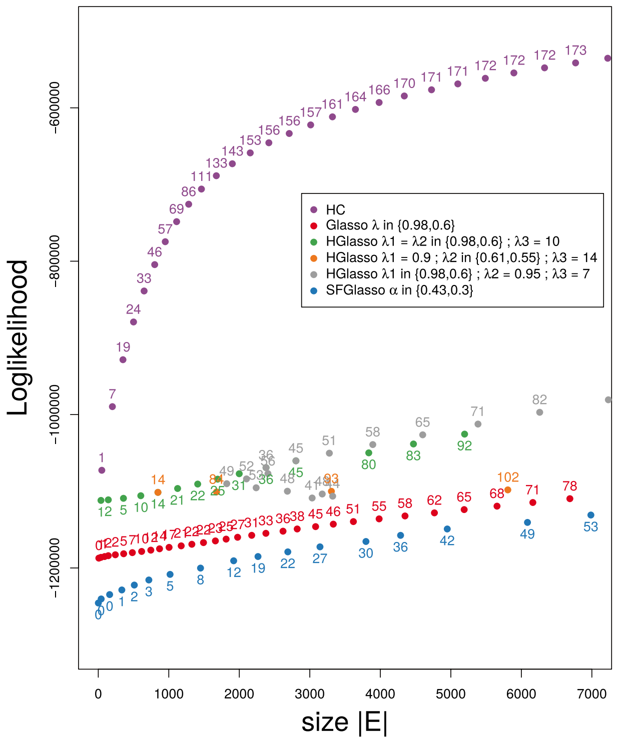

Figure 4. Log-likelihood values (dots) and amount of transcription factors (numbers) w.r.t. the networksize |E| as encountered from the M3D microarray dataset by HC (purple), Glasso with $\lambda \in \{0.98, 0.6\}$ (red), HGlasso with $\lambda_1 = \lambda_2 \in \{0.98, 0.6\}, \lambda_3 = 10$ (green), HGlasso with $\lambda_1 = 0.9, \lambda_2 \in \{0.61, 0.55\}, \lambda_3 = 14$ (orange), HGlasso with $\lambda_1 \in \{0.98, 0.6\}, \lambda_2 = 0.95, \lambda_3 = 7$ (grey), and SFGlasso with $\alpha \in \{0.43, 0.3\}$ (blue).

there is a similar performance in other portions of the network—and we compare with the networks learned from M3D with the four structure learning algorithms. Panel (a) in Fig. 5 shows the two TF encoding genes hyfR and fhlA (coloured in red) and their direct neighbours in the RegulonDB network (among the direct neighbours are three other TFs that are also colored in red: crp, fnr and nsrR).

The RegulonDB is a directed network and symmetric dependencies that could exist between children of hyfR and/or fhlA due to their common parent are not documented in the RegulonDB and hence are not displayed in panel (a). Panels (b–e) show the same subset of nodes as panel (a) and form part of networks that are learned from M3D by respectively HC, Glasso, HGlasso, and SFGlasso. These networks are all (converted to, in the case of HC) UNMs in which edges are undirected and represent direct dependency between two variables (the edges represent non zero entries in the symmetric precision matrix). The exact encoding of dependencies in the networks is described in the "Methods" section "Dependencies in PN". The grey nodes indicate genes that are integrated in the network and red edges indicate edges that find their direct counterpart in the reference network in panel (a), i.e. edges that are denominated TPs. The edge-labels display the weight of the edges, the estimated partial variation. In panel (d) blue edges and blue vertex closure indicate edges and vertex that are included in the HGlasso network but not in the Glasso network in panel (c) of similar size, whereas grey edges and black vertex closures indicate edges and vertices that exist in both networks.

The results in the above subsections are illustrated in the subnetworks. In a sparse network, Hill Climbing (panel (b)) discovers both TF encoding genes, hyfR and fhlA and with them two TP edges between respectively hyfR and hyfJ and fhlA and hypE. Glasso, HGlasso, and SFGlasso (panels (c)–(e)) discover only one TF, hyfR, and find one TP related to hyfR.

Despite finding the most TPs, HC also discovers the most unique child–child dependencies between the target genes that are regulated by the transcription factors HyfR and FhlA. Child–child dependencies are not considered as TPs in the RegulonDB network, but do improve quality of UNM network structure. For example, the community $\{hyfA, hyfB, \ldots, hyfJ\}$ and the TP between hyfR and hyfJ can be an indication that the set $\{hyfA, hyfB, \ldots, hyfJ\} \setminus \{hyfJ\}$ is also progeny of HyfR.

Glasso finds less variety of child–child dependencies between nodes that are regulated by HyfR and FhlA. Instead, the sub groups of children are interconnected with more edges between them. These edges are probably true, but, from an information theoretic perspective, are redundant in the sense that they lower entropy of the community structure in the network (see also[28]). Moreover, the values of the parameters, the partial variation, in Glasso differ in magnitude with the values of the parameters in HC. The low partial variation as estimated by (H)Glasso lower log-likelihood values with respect to HC. In the discussion we come back to the estimation of the magnitude of the parameters and its impact on log-likelihood values.

In panel (d), HGlasso improves the network structure with respect to Glasso with an additional connected gene hyfG while using the same amount of edges. Also, parameter estimation seems more accurate as HGlasso

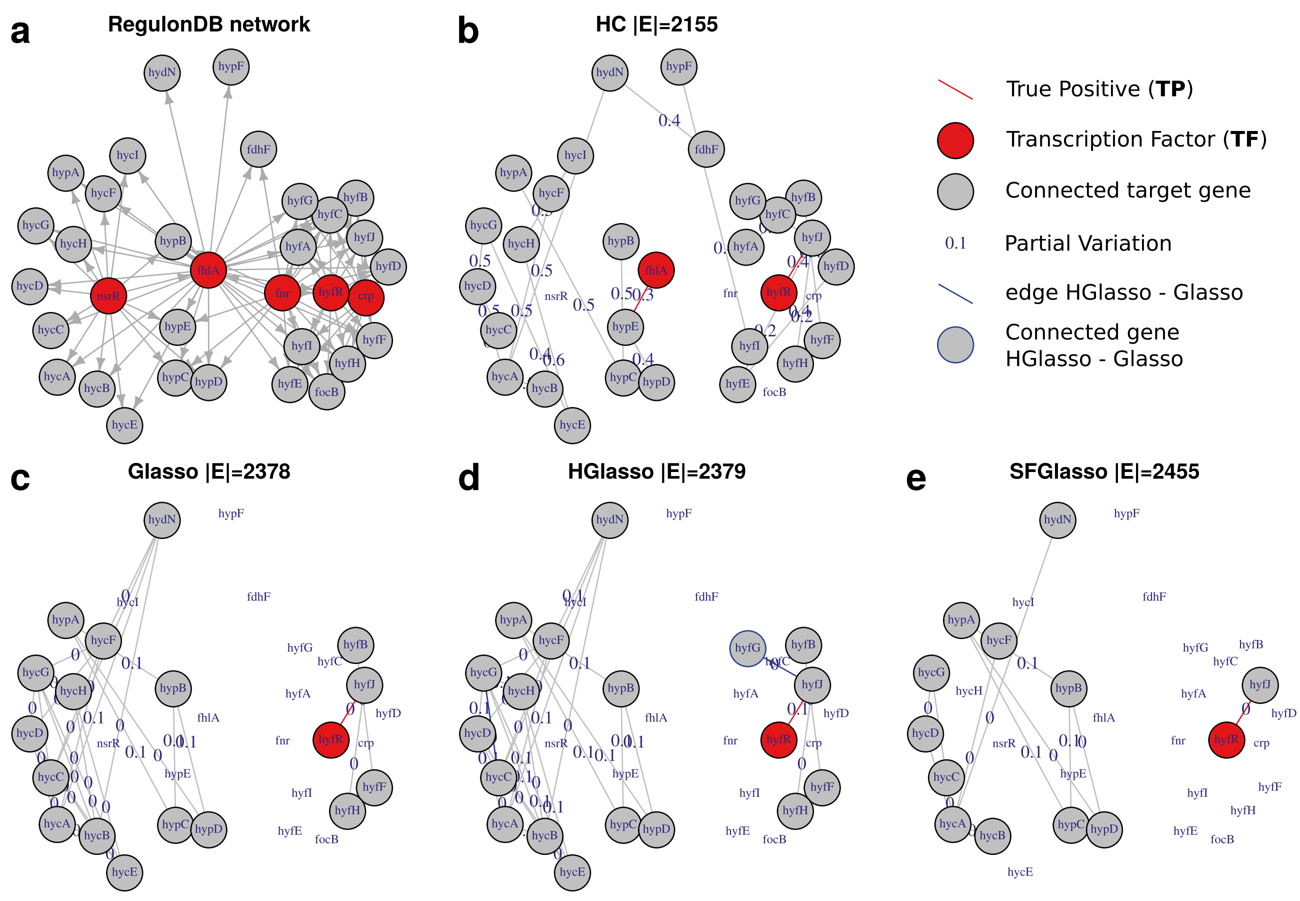

Figure 5. Reference and estimated networks. Panel (**a**) is the part of the reference RegulonDB network that contains the two TF encoding genes hyfR and fhlA and their direct neighbours. Panel (**b**–**e**) are the corresponding sub-networks of networks that are learned with respectively HC, (H)Glasso, and SFGlasso. In panel (**a**) TF encoding genes are coloured in red. In (**b**–**e**) TF encoding genes are coloured in red only when included in the network and connected genes are coloured grey. Red edges in (**b**–**e**) have a direct counterpart in panel (**a**) (i.e. red edges are TPs). In panel (**d**) blue edges and blue vertex closure indicate edges and vertex that are included in the HGlasso network but not in the Glasso network of the same size, whereas grey edges and black vertex closures indicate edges and vertices that exist in both networks.

includes higher partial variation values. In general, the inclusion of more significant edges (for some combinations of $\lambda_1, \lambda_2, \lambda_3$) and more precise parameter estimation alter the log-likelihood values of HGlasso with respect to Glasso, showed in Figs. 3 and 4.

The SFGlasso in panel (e) on the other hand excludes four more genes than the Glasso. Not many dependencies are found in this part of the sub network. Thereby, the estimation of parameters is worse in this part of the sub network. Partial variations are of even smaller magnitude with respect to Glasso. These results are in accordance with Fig. 3 showing that SFGlasso networks are less complete than (H)Glasso networks and with Fig. 4 showing that the least amount of information can be induced from SFGlasso networks.

Discussion

Our results on sparse and medium-size networks showed high log-likelihood values for HC with respect to the standard Glasso and its modifications. Moreover, the results for sparse networks (up to 5000 edges) show that the HC algorithm finds most interactions for which evidence exists (in the form of TPs). We first discuss the log-likelihood and TP results in the light of the topology of the learned networks and the score functions that are integrated in the algorithms. Then, we analyze the results of HC in the framework of two additional partial-variation learning algorithms (SPACE and ESPACE) that are evaluated in the literature[30] using the same *E. coli* dataset and an similar evaluation procedure. Finally, we discuss some generalities about GBNs that are learned with HC and applied on high dimensional real-world data[28].

When analyzing log-likelihood values, we should first keep in mind that not all interactions that can be inferred from the M3D are equally informative and thus not contribute equally to the log-likelihood. It is intuitive that TF–target-gene interactions determine the backbone structure of the M3D data, whereas local interaction structure between children of the same TFs is less informative with respect to the complete M3D database. Therefore, the HC algorithm produces network topologies that contribute to high log-likelihood values, finding relatively more TPs and much more TFs than Glasso and its modifications in sparse networks, and just enough links to determine local interaction structure. Secondly, accurate estimation of the incorporated parameters (or the weight of the edges) alters log-likelihood values. The HC algorithm is able to accurately estimate the parameters, thanks to the l_0 penalty used in the score function that only penalizes the *amount* of parameters and not their values. The l_1 regularization in Glasso, on the other hand, while penalizing the amount of parameters, also penalizes their values and, thus, imposes a structural bias on the selected parameters in the Gaussian JPD function, resulting in inaccurate parameter estimation and substantially lower log-likelihood values. The score functions including l_1 or l_0 penalization are described in detail in "Methods" sections "Learning PN structure from data" and "Learning BN structure from data".

The reasons for HC being more efficient than Glasso and modifications in sparse networks have to do with the integrated score functions and the steps in the learning algorithms. In every iteration, HC incorporates a parameter/edge maximizing the score function composed by the log-likelihood and the l_0 penalization terms. On the one hand, there are only a few high informative links (from TFs) that alter the log-likelihood substantively. HC is designed to optimize log-likelihood with the minimum amount of parameters and, thus, includes first TFs and informative local links that enlarge communities. On the other hand, the Glasso score function includes the log-likelihood part and the l_1 penalization term. Local links generally have a stronger direct correlation (see Fig. 5) and Glasso includes more of them, whether informative or uninformative, as local links 'remain' under application of the statistical criterion. More informative links between TFs and target genes have weaker direct correlation and in sparse networks Glasso shrinks those links to zero.

The differences in results between Glasso and modifications can be explained using similar arguments. Due to the adapted l_1 regularization, HGlasso performs better on log-likelihood values than the standard Glasso for every combination of initial parameters $\lambda_1, \lambda_2, \lambda_3$. The new score function alleviates the penalization on parameters that are related to hubs, resulting in higher log-likelihood values with respect to the standard Glasso. HGlasso, however, still depends heavily on λ_1 (the penalty term for non hubs) to introduce overall sparsity, and thus can not reduce the mean parameter bias sufficiently to reach HC values for small and medium networks. With respect to the measure of true positives, the performance of HGlasso (including whether it outperforms Glasso) depends on the combination of tuning parameters and on the network size. A successful combination of tuning parameters leads HGlasso to finding more TPs. These networks generally contain more TFs than the associated Glasso network. However, the incorporation of more TFs does not always lead to more true positives in the network. We found no standard method to discover combinations of initial parameters with which Glasso could be outperformed. Finding such a combination is a time consuming trial and error process.

The algorithm SFGlasso does incorporate a hub structure, but the networks generally contain less TFs than Glasso and the combinations of HGlasso. As resulting from the low log-likelihood values, the algorithm thus selects 'wrong' hubs and also places great bias on the estimates of informative links. These results may have to do with the poor adjustment of SFGlasso to high dimensional data. Generally, in scale-free networks, the coefficient α in the power-law with which the degree distribution decays lies between 2 and 3. However, we had to reduce α to values below 0.5 as otherwise the resulting networks resulted empty. For $\alpha \leq 1$ the degree distribution is not generally well-defined and requires the existence of a maximum degree k_{max} to be normalizable. In this work we only display results obtained with two iterations of the algorithm, because using more than two iterations the Glasso algorithm took extraordinary long time to converge. In this sense, one must be careful with the interpretation of the power-law resulting from SFGlasso.

The analysis of HGlasso and SFGlasso with respect to Glasso shows that the intrinsic complexity structure of the complex system is not (fully) represented by their ad hoc procedure to impose a hub structure and a power-law degree distribution in the network. Therefore, we can conclude that the GRN underlying *E. coli* is either not fully characterized by a power-law degree distribution and/or hub nodes or the algorithms fail to faithfully model these characteristics from high dimensional complex data.

Besides the standard Glasso and variants used in this paper, other advanced algorithms such as SPACE and ESPACE[30] have been been applied on an older version of the microarray data. The algorithm SPACE uses sparse covariance selection techniques and was especially designed for the large-*p*-small-*n* data in the framework of GRNs and for powerful identification of hubs[34]. The ESPACE algorithm is an extension of the SPACE method and explicitly accounts for prior information on hub genes, which, in the case of *E. coli* data, yields knowing in advance that TFs are the highly connected nodes in the true *E. coli* GRN. We can loosely compare the HC algorithm with the performance of the algorithms SPACE and ESPACE. In particular, SPACE reports a precision of 4.35% in a network of 386 edges, and ESPACE reports a precision of 12.89% in a network of 349 edges[30]. In our work, HC shows a precision of 4.85% in a network of 350 edges. We may conclude with some precaution, as both M3D and the RegulonDB has been extended and updated meanwhile, that HC outperforms the exploratory algorithm SPACE, but can not compete with ESPACE that starts with prior information about the true hubs, i.e. information about the transcription factors.

Finally, the great power of HC when exploring *E. coli* data with unknown structure is that one can extract information of the true deterministic underlying network structure from every edge, as was illustrated at hand of the graphs in Fig. 5. The fact that no edge is redundant with respect to the dataset enables us to learn from every 'false' positive edge, whereas in the case of Glasso and variants this is not true: not every edge is data-significant neither tells us more about the network structure. The same conclusion was drawn for high dimensional climate data for which we found that HC provided an informative community structure that can be analyzed well with centrality measures (i.e. betweenness centrality, see[28]). The HC approach prioritizes an efficient edge distribution by favouring heterogeneous selection of edges between communities over uniform selection of edges that lie in communities. The resulting network topology allows for analysis with complex network measures; in the case of climate data these measures revealed spatially distant communities interconnected by teleconnections. Complex network centrality measures also have high potential applied on the PNMs learned with HC from real-world microarray data, especially when TF encoding genes are not known (i.e. no database similar to the RegulonDB exists); in this case to reveal candidate TFs and distinguish regulatory interactions from target-gene–target-gene interactions.

Conclusions

The use of the Hill Climbing algorithm that arises in the context of Gaussian Bayesian networks offers a sound approach for the reconstruction of gene regulatory networks of high-dimensional-low-sample-size microarray data when no initial information is at hand about the underlying complex dependency structure. The HC algorithm picks only the most significant dependencies from the Gaussian data and in this manner naturally includes the complex dependency structure of the complex GRN that may consist of hubs, a scale-free degree distribution, outliers, a combination of the former or of other real-world characteristics. The algorithm naturally leaves out uninformative dependencies and variables.

The Bayesian network can easily be transformed to an undirected Gaussian probabilistic network model (i.e. a Gaussian Markov network or precision network), paying the price of some loss of information on the independence structure with respect to its initial directed Bayesian network. If one prefers an undirected PNM over a directed PNM—for easy interpretation due to symmetric links or for sparse estimation of the inverse covariance matrix—this study shows that the transformation from BN to PN is worth this loss of information for sparse networks as the PNs obtained by Hill Climbing outperform PNs obtained by state of the art PN-structure learning algorithms when applied to high-dimensional-low-sample-size (microarray) data that contains an unknown complex dependency structure.

This conclusion is drawn with respect to state-of-the-art PN structure algorithms that arise in the context of undirected Gaussian network models: the Glasso algorithm and variants of Glasso that are developed to integrate complex dependency structure. These algorithms model unnecessary dependencies at the expense of the probabilistic information in the network and of a structural bias in the probability function that can only be relieved including many parameters. In the case of the *E. coli* gene expression data used in this work, unnecessary dependencies also go at the expense of the amount of true positive edges, the last as judged by a reference network compounded of evidence gathered in the RegulonDB.

Methods

Gaussian probabilistic network models. The term refers to the choice of a multivariate Gaussian Joint Probability Density (JPD) function to associate graph edges with model parameters in a given PNM, such that the probabilistic model encodes in the JPD function a large number of random variables that interact in a complex way with each other by a graphical model. The multivariate Gaussian JPD function can take various representations in which dependencies between the variables are described by different types of parameters. The best known representation of the Gaussian JPD function is in terms of marginal dependencies, *i.e.* dependencies of the form $X_i, X_j|\emptyset$ as present in the covariance matrix $\mathbf{\Sigma}$. Let $\mathbf{X}$ be a N-dimensional multivariate Gaussian variable then its probability density function $\mathrm{P}(\mathbf{X})$ is given by:

$$\mathrm{P}(\mathbf{X}) = (2\pi)^{-N/2}\det(\mathbf{\Sigma})^{-1/2}\exp\{-1/2(\mathbf{X}-\boldsymbol{\mu})^\top\mathbf{\Sigma}^{-1}(\mathbf{X}-\boldsymbol{\mu})\}, \tag{1}$$

where $\boldsymbol{\mu}$ is the N-dimensional mean vector and $\mathbf{\Sigma}$ the $N \times N$ covariance matrix. In the following we describe in some detail two types of Gaussian PNMs, in which parameters reflect respectively direct dependencies $X_i, X_j|\mathbf{X}\backslash\{X_i, X_j\}$ and general conditional dependencies $X_i|\mathscr{S}$ with $\mathscr{S} \subseteq \mathbf{X}$ (direct dependencies are the least restrictive case of conditional dependencies).

Probabilistic PN models. The Gaussian JPD function in Eq. (1) can be formulated more generally using a set of factors $\mathbf{\Phi} = \{\phi_1(\mathscr{S}_1), \dots, \phi_k(\mathscr{S}_k)\}$ that describe dependencies between arbitrary (overlapping) subsets of variables $\mathscr{S}_1, \dots, \mathscr{S}_k$ which comply with $\cup_k \mathscr{S}_k = \mathbf{X}$. This representation of the JPD is called the Gibbs function and written as[31]

$$\mathrm{P}(X_1, \dots, X_N) = \frac{1}{Z}\tilde{\mathrm{P}}(X_1, \dots, X_N), \tag{2}$$

with

$$\tilde{\mathrm{P}}(X_1, \dots, X_N) = \prod_{i=1}^{k}\phi_i(\mathscr{S}_i) \qquad \text{and} \qquad Z = \sum_{i=1}^{N}\tilde{\mathrm{P}}(X_1, \dots, X_N). \tag{3}$$

The Gibbs distribution where all of the factors are over subsets of single variables or pairs of variables is called a pairwise Markov network. The factors in a pairwise Markov network correspond to direct dependencies, i.e., dependencies of the form $X_i, X_j|\mathbf{X}\backslash\{X_i, X_j\}$. In a Gaussian distribution these dependencies are present in the inverse covariance matrix or precision matrix $\mathbf{\Sigma}^{-1}$. The information form of the Gaussian JPD function in terms of the precision matrix $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$,

$$\mathrm{P}(\mathbf{X}, \mu = 0, \mathbf{\Theta}) = (2\pi)^{-N/2}|\mathbf{\Theta}|^{1/2}\exp\{-1/2\sum_i \theta_{ii}X_i^2 - \sum_{i<j}\theta_{ij}X_iX_j\}, \tag{4}$$

is equivalent to the Gibbs function in Eq. (2) with factors defined on every variable and every pair of variables, i.e. $\mathbf{\Phi} = \{\mathbf{\Phi}^n, \mathbf{\Phi}^e\}$ with $\phi_i^n = \exp\{-\frac{1}{2}\theta_{ii}X_i^2\}$ and $\phi_{ij}^e = \exp\{\theta_{ij}X_iX_j\}$.

The corresponding Gaussian PNM in which the notion of variables and pairs of variables is extended to the notion of nodes and undirected edges in a graph is called the probabilistic PN model. The graph of a PN encodes the probability function in Eq. (4) as follows. Each node corresponds to a variable $X_i \in \mathbf{X}$, the presence of an edge $X_i - X_j$ implies the presence of the factor ϕ^e_{ij} in $\mathrm{P}(\mathbf{X})$ and direct dependency of X_i and X_j. Moreover, the absence of an arc between X_i and X_j in the graph implies the absence of the factor ϕ^e_{ij} in $\mathrm{P}(\mathbf{X})$ and, thus, the existence of a set of variables $\mathscr{S} \subseteq \mathbf{X}\backslash\{X_i, X_j\}$ that makes X_i and X_j conditionally independent in probability[31].

The graph structure of PNs in this work is estimated simultaneously with the values of the parameters in $\mathbf{\Theta}$ that define this structure. This simultaneous learning process is explained in "Methods" section "Learning PN structure from data".

Probabilistic BN models. Alternatively, the $\mathrm{P}(\mathbf{X})$ in Eq. (1) can be characterized with conditional dependencies of the form $X_i|\mathscr{S}$ with $\mathscr{S} \subseteq \mathbf{X}$. The representation of the JPD is then a product of conditional probability densities:

$$\mathrm{P}(X_1, \ldots, X_N) = \prod_{i=1}^{N} \mathrm{P}_i(X_i|\Pi_{X_i}) \tag{5}$$

with

$$\mathrm{P}(X_i|\Pi_{X_i}) \sim \mathcal{N}\left(\mu_i + \sum_{j|X_j \in \Pi_{X_i}} \beta_{ij}(X_j - \mu_j),\ \frac{1}{\nu_i}\right) \tag{6}$$

whenever the set of random variables $\{X_i|\Pi_{X_i}\}_{i\in N}$ is independent[35]. In this representation $\mathcal{N}$ is the normal distribution, μ_i is the unconditional mean of X_i, ν_i is the conditional variance of X_i given the set Π_{X_i} and β_{ij} is the regression coefficient of X_j, when X_i is regressed on Π_{X_i}. We call Π_{X_i} the parentset of variable X_i.

The corresponding Gaussian PNM in this case is the probabilistic BN model. The graph of a BN model is a DAG encoding the corresponding probability distribution as in Eq. (5). Each node corresponds to a variable $X_i \in \mathbf{X}$, the presence of an arc $X_j \rightarrow X_i$ implies the presence of the factor $\mathrm{P}_i(X_i|\ldots X_j\ldots)$ in $\mathrm{P}(\mathbf{X})$, and thus conditional dependence of X_i and X_j. Moreover, the absence of an arc between X_i and X_j in the graph implies the absence of the factors $\mathrm{P}_i(X_i|\ldots X_j\ldots)$ and $\mathrm{P}_j(X_j|\ldots X_i\ldots)$ in $\mathrm{P}(\mathbf{X})$ and, thus, the existence of a set of variables $\mathscr{S} \subseteq \mathbf{X}\backslash\{X_i, X_j\}$ that makes X_i and X_j conditionally independent in probability[31,36].

The graph structure of the BN identifies the parentset Π_{X_i} in Eq. (5). With this structure available, one easily learns the corresponding parameter set $(\boldsymbol{\beta}, \boldsymbol{\nu})$; in our case parameters β_{ij} and ν_i are a maximum likelihood fit of the linear regression of X_i on its parentset Π_{X_i}. To estimate parameter values from graph structure we use the appropriate function in the R-package BNLEARN[37]. The challenge of learning the graph structure is explained in "Methods" section "Learning BN structure from data".

Learning PN structure from data. A Precision Network (PN) is learned with the help of a structure learning algorithm that estimates the inverse covariance matrix, i.e. the precision matrix $\mathbf{\Sigma}^{-1}$ of the underlying Gaussian distribution. Converted into binary format, the estimate $\mathbf{\Theta}$ of $\mathbf{\Sigma}^{-1}$ provides the undirected adjacency matrix $\boldsymbol{A}$ of a pairwise Markov Network. From the adjacency matrix $\boldsymbol{A}$ of the graph of a PN the structure of the factor-set $\mathbf{\Phi} = \{\mathbf{\Phi}^n, \mathbf{\Phi}^e\}$ of the associated Gaussian JPD function (outlined in Eq. (4)) can be directly read off. In the "Methods" section "Probabilistic PN models" is explained how pairwise Markov networks encode the corresponding Gaussian PNM.

The Graphical lasso (Glasso) can be regarded as the default structure learning algorithm learning PNs for large-p-small-n datasets. Glasso is a score-based algorithm based on a convex score. This score is basically made up by the maximum likelihood estimate of the precision matrix of a Gaussian probability function to which an l_1 penalty term is added[15]:

$$Score(\mathbf{\Theta}, \mathbf{S}, \lambda) = \log(\det \mathbf{S\Theta}) - tr(\mathbf{S\Theta}) - \lambda\|\mathbf{\Theta}\|_1. \tag{7}$$

Here $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$, $\mathbf{S}$ is the sample covariance matrix calculated directly from the data $\mathscr{D}$ and λ a scalar, the penalization coefficient. Networks of different sizes can be generated by varying the penalization parameter λ. In this work we generate networks with zero edges to complete networks by varying λ from 1 to 0. A short outline of the steps in the Graphical lasso algorithm is given in "Methods" section "Learning with hill climbing and glasso". The Glasso function is implemented in the R-package GLASSO[15].

The Hub Graphical lasso (HGlasso) learns a PN that consist of hub nodes combining a lasso (l_1) penalty and a sparse group lasso (l_2) penalty[38]. The estimated inverse covariance matrix $\mathbf{\Theta}$ can be decomposed as $\mathbf{\Theta} = \mathbf{Z} + \mathbf{V} + t(\mathbf{V})$, where $\mathbf{Z}$ is a sparse matrix and $\mathbf{V}$ is a matrix that contains hub nodes. The belonging score is

$$\begin{aligned} Score(\mathbf{\Theta}, \mathbf{S}, \lambda_1, \lambda_2, \lambda_3) = {} & \log(\det \mathbf{S\Theta}) - tr(\mathbf{S\Theta}) \\ & - \lambda_1\|\mathbf{Z}\|_1 - \lambda_2\|\mathbf{V} - \mathrm{diag}(\mathbf{V})\|_1 \\ & - \lambda_3 \sum_{j=1}^{N} \|(\mathbf{V} - \mathrm{diag}(\mathbf{V}))_j\|_2, \end{aligned} \tag{8}$$

with $\mathbf{\Theta}$ restricted to $\mathbf{\Theta} = \mathbf{V} + \mathbf{V}^T + \mathbf{Z}$. In this score λ_3 controls the selection of hub nodes, and λ_2 controls the sparsity of each hub nodes' connections to other nodes. We obtain networks of different sizes by varying λ_1, λ_2, and λ_3. The HGlasso function is implemented in the R-package HGLASSO[38].

The Scale-Free Graphical lasso (SFGlasso) aims to include even more structural information than mere sparsity or hubs[26]. Hubs are expected in this type of network but the focus lies on learning models that possess the so-called scale-free property; a property often claimed to appear in real-world networks. This feature is mathematically expressed by a degree distribution $p(d)$ that follows a powerlaw: $p(d) \propto d^{-\alpha}$ (up from a certain degree d). In the score function of the standard Glasso, the l_1 edge regularization is replaced with a power law regularization. The objective score function is:

$$\begin{aligned} Score(\mathbf{\Theta}, \mathbf{S}, \alpha, \beta) = {} & \log(\det \mathbf{S\Theta}) - tr(\mathbf{S\Theta}) \\ & - \alpha \sum_i \log(\|\mathbf{\Theta}\neg i\|_1 + \varepsilon_i) - \beta \sum_i |\theta_{ii}|, \end{aligned} \tag{9}$$

with $\mathbf{\Theta}_{\neg i} = \{\theta_{ij} | j \neq i\}$. This score function is not convex, a requirement to use Glasso, however can be proven to be monotone increasing. The score $Score(\mathbf{\Theta}, \mathbf{S}, \alpha, \beta)$ is sequentially improved by elements of the sequence $\mathbf{\Theta}^n$ that iteratively maximize the following re-weighted convex l_1 regularization problems:

$$\begin{aligned} Score(\mathbf{\Theta}^{n+1}, \mathbf{S}, \lambda_{ij}) = {} & \log(\det \mathbf{S\Theta}^{n+1}) - tr(\mathbf{S\Theta}^{n+1}) \\ & - \sum_{i \neq j} \lambda_{ij} |\theta_{ij}^n| - \beta \sum_i |\theta_{ii}^n|, \end{aligned} \tag{10}$$

where $\lambda_{ij} = \alpha(\frac{1}{\|\mathbf{\Theta}^n_{\neg i}\|_1 + \varepsilon_i} + \frac{1}{\|\mathbf{\Theta}^n_{\neg j}\|_1 + \varepsilon_j})$. This re-weighting reduces regularization coefficients of nodes with high degree, encouraging the appearance of hubs with high degree.

Following the set up in the experiment section of Ref.[26] we take $\beta_i = 2\alpha/\varepsilon_i$ and ε_i equal to θ_{ii} estimated in the last iteration, in this way ε_i is of the same magnitude of $\|\mathbf{\Theta}^n_{\neg j}\|_1$. To optimize Eq. (10) and find $\mathbf{\Theta}^{n+1}$ we iteratively use the glasso function in the R-package GLASSO with $\lambda = \lambda(\mathbf{\Theta}^n)$, defined above.

Learning BN structure from data. The graph of a BN is estimated with the help of a structure learning algorithm that finds the conditional dependencies between the variables and encodes this information in a DAG. Graphical (dis)connection in the DAG implies conditional (in)dependence in probability (see "Methods" section "Dependencies in BN"). From the structure of a BN a factorization of the underlying JPD function $\mathrm{P}(\mathbf{X})$ of the multivariate random variable $\mathbf{X}$ (as given by Eq. (5)) can be deduced. In the "Methods" section "Probabilistic BN models" is explained how networks can be extended to their corresponding probabilistic network model.

In general there are three types of structure learning algorithms: constraint-based, score-based, and hybrid structure learning algorithms—the latter being a combination of the first two algorithms.

Constraint-based algorithms use conditional independence tests of the form $Test(X_i, X_j | \mathscr{S}; \mathscr{D})$ with increasingly large candidate separating sets $\mathscr{S}_{X_i,X_j}$ to decide whether two variables X_i and X_j are conditionally independent. All constraint-based algorithms are based on the work of Pearl on causal graphical models[39] and its first practical implementation was found in the Principal Components algorithm[40]. In contrast, score-based algorithms apply general machine learning optimization techniques to learn the structure of a BN. Each candidate network is assigned a network score reflecting its goodness of fit, which the algorithm then attempts to maximise[41]. In Ref.[29] we compared structure learning algorithms belonging to the three different classes on accuracy and speed for high dimensional complex data. We found that score-based algorithms perform best. Algorithms in this class are able to handle high-variable-low-sample-size data and find networks of all desired sizes. Constraint-based algorithms can only model complex data up to a certain size and, as a consequence, for climate data they only reveal local network topology. Hybrid algorithms perform better than constraint-based algorithms on complex data, but worse than score-based algorithms.

In this work we use a simple score-based algorithm, the Hill Climbing (HC) algorithm[41], to learn BN structure. The HC algorithm starts with an empty graph and iteratively adds, removes or reverses an edge maximizing the score function. This algorithm is formalized in "Methods" section "Learning with hill climbing and glasso". HC is implemented in the R-package BNLEARN.

We used the Bayesian Information Criteria (BIC) (corresponding to BIC_0 in Ref.[29]) score, which is defined as:

$$\mathrm{BIC}(\mathscr{G}; \mathscr{D}) = \sum_{i=1}^{N} \left[\log \mathrm{P}(X_i | \Pi_{X_i}) - \frac{|\mathbf{\Theta}_{X_i}|}{2} \log n \right], \tag{11}$$

where $\mathscr{G}$ refers to the graph (DAG) for which the BIC score is calculated, P refers to the probability density function that can be deduced from the graph (see "Methods" section "Probabilistic BN models"), Π_{X_i} refer to the parents of X_i in the graph (*i.e.* nodes Y with relation $Y \rightarrow X_i$ in the graph) and $|\mathbf{\Theta}_{X_i}|$ is the amount of parameters of the local density function $\mathrm{P}(X_i | \Pi_{X_i})$.

Dependencies in BN and PN structure. In the following we describe how a BN (or DAG) and a PN (or pairwise Markov network) encode conditional dependencies. New nomenclature is indicated with an asterisk and illustrated in Fig. 6a–c.

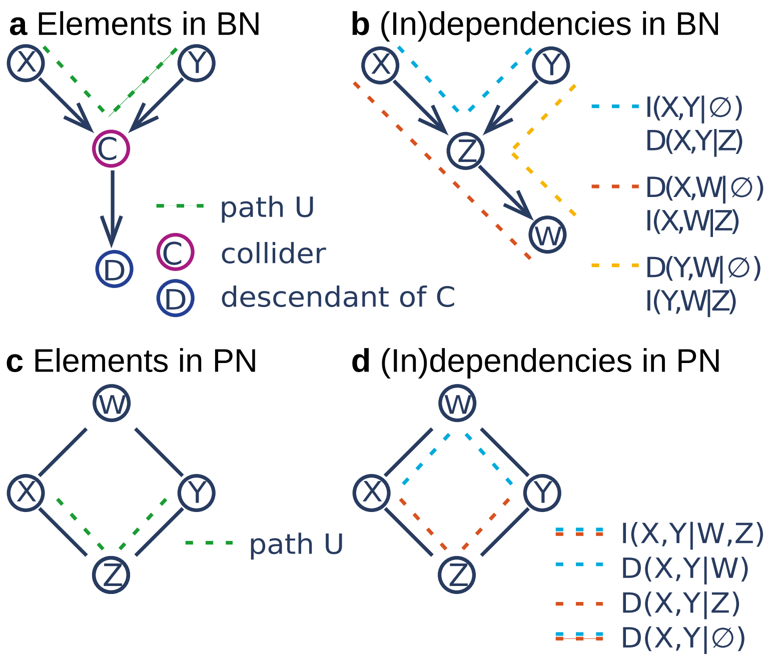

Figure 6. (**a**) and (**c**): Nomenclature of elements in respectively a Bayesian Network (BN) and a Precision Network (PN). (**b**) and (**d**): Some (in)dependencies in simple BN and PN consisting of four nodes X, Y, Z and W. In (**b**) two sets of nodes are dependent given a third if conditions (1) and (2) in the main text are fulfilled. On the one hand, the conditional relationship $X, Y|Z$ and the marginal relationships $X, W|\emptyset$ and $Y, W|\emptyset$ satisfy conditions (1) and (2), so that we have $\mathrm{D}(X, Y|Z)$, $\mathrm{D}(X, W|\emptyset)$, and $\mathrm{D}(Y, W|\emptyset)$. On the other hand, the marginal relationship $X, Y|\emptyset$ violates condition (1) and the conditional relationships $X, W|Z$ and $Y, W|Z$ violate condition (2), so that we have $\mathrm{I}(X, Y|\emptyset)$, and $\mathrm{I}(X, W|Z)$ and $\mathrm{I}(Y, W|Z)$. In (**d**) the conditional relationships $X, Y|W$, $X, Y|Z$, and $X, Y|\emptyset$ satisfy the condition of graphical dependence in a PN, and hence the statements $\mathrm{D}(X, Y|W)$, $\mathrm{D}(X, Y|Z)$, and $\mathrm{D}(X, Y|\emptyset)$ hold. On the other hand the conditional relation $X, Y|W, Z$ does not satisfy the condition of graphical dependence; there does not exist a path U, such that neither W nor Z is not on U. Thus $\mathrm{I}(X, Y|W, Z)$ holds. In the following we give a formal proof of $\mathrm{D}(X, Y|Z)$ in (**b**): The conditioning set $\mathscr{S}$ exists of Z. The only path between X and Y is the blue path. Hence we declare the blue path U. Z is a collider and Z is in $\mathscr{S}$. There are no other colliders on U. Hence condition (1) is satisfied. Z is the only variable on U. And Z is a collider. Thus, U does not contain non-colliders. Hence condition (2) is satisfied. As condition (1) and (2) are satisfied we have that X and Y are dependent given Z, i.e. $\mathrm{D}(X, Y|Z)$.

Dependencies in BN. In a BN two nodes X and Y are conditionally dependent given a set $\mathscr{S}$ (denoted by $\mathrm{D}(X, Y|\mathscr{S})$) if and only if they are graphically connected, that is, if and only if there exists a path U* between X and Y satisfying the following two conditions:

- *Condition (1)*: for every collider* C (node C such that the part of U that goes over C has the form of a V-structure, *i.e.* $\rightarrow C \leftarrow$) on U, either C or a descendant* of C is in $\mathscr{S}$.
- *Condition (2)*: no non-collider on U is in $\mathscr{S}$.

If the above conditions do not hold we call X and Y conditionally independent given the set $\mathscr{S}$ (denoted by $\mathrm{I}(X, Y|\mathscr{S})$). Marginal dependency between two nodes can be encoded by any path U with no V-structures. In Fig. 6b six conditional (in)dependence statements are highlighted in a simple DAG. In the caption of Fig. 6 one of the statements is proved at the hand of conditions (1) and (2).

Dependencies in PN. In a PN two nodes X and Y are conditionally dependent given a set $\mathscr{S}$ (denoted by $\mathrm{D}(X, Y|\mathscr{S})$) if and only if there exists a path U* between X and Y satisfying: No node $Z \in \mathscr{S}$ is on U. If the above condition do not hold we call X and Y conditionally independent given the set $\mathscr{S}$ (denoted by $\mathrm{I}(X, Y|\mathscr{S})$). Marginal dependency between two nodes can be encoded by any path U. In Fig. 6d four conditional (in)dependence statements are highlighted in a simple pairwise Markov network.

Learning with hill climbing and glasso.

```
Algorithm 1 Hill Climbing[29]
Input: a data set $\mathscr{D}$ from $\mathbf{X}$, an initial (usually empty) DAG $\mathscr{G}$ and a score function $Score(\mathscr{G},\mathscr{D})$ as given in equation (11).
Output: the DAG $\mathscr{G}_{max}$ that maximises $Score(\mathscr{G},\mathscr{D})$.
  1. Compute the score of $\mathscr{G}$, $S_{\mathscr{G}} = Score(\mathscr{G},\mathscr{D})$, and set $S_{max} = S_{\mathscr{G}}$ and $\mathscr{G}_{max} = \mathscr{G}$.
  2. Repeat as long as $S_{max}$ increases:
     (a) for every (or some; simple hill climbing) possible arc addition, deletion or reversal in $\mathscr{G}_{max}$ resulting in a DAG:
         i. compute the score of the modified DAG $\mathscr{G}^*$, $S_{\mathscr{G}^*} = Score(\mathscr{G}^*,\mathscr{D})$:
         ii. if $S_{\mathscr{G}^*} > S_{max}$ and $S_{\mathscr{G}^*} > S_{\mathscr{G}}$, set $\mathscr{G} = \mathscr{G}^*$ and $S_{\mathscr{G}} = S_{\mathscr{G}^*}$.
     (b) if $S_{\mathscr{G}} > S_{max}$, set $S_{max} = S_{\mathscr{G}}$ and $\mathscr{G}_{max} = \mathscr{G}$.
```

```
Algorithm 2 Graphical Lasso
Input: The sample correlation matrix $\mathbf{S}$ generated from a data set $\mathscr{D}$ from $\mathbf{X}$ and the penalization coefficient $\lambda$.
Output: The estimated precision matrix $\mathbf{\Theta}$ (in binary format the undirected PN graph) that maximises the $Score(\mathbf{\Theta},\mathscr{D})$ as given in equation (7).
  1. Start with $\boldsymbol{W} = \mathbf{S} + \lambda \boldsymbol{I}$. The diagonal of $\boldsymbol{W}$ remains unchanged in what follows.
  2. Repeat until convergence:
     For each $j = 1,2,\ldots,N,1,2,\ldots,N,\ldots$:
     (a) Reorganize the matrix $\boldsymbol{W}$ in $\boldsymbol{W}_{11}$ (all but the jth row and column of $\boldsymbol{W}$), and $\boldsymbol{w}_{12}$ and $\boldsymbol{w}_{21}$, the jth row and column without the diagional element $w_{22}$. Do the same for $\mathbf{S}$.
     (b) Solve the lasso regression problem $\min_{\boldsymbol{\beta}}\{\frac{1}{2}\|\boldsymbol{W}_{11}^{1/2}\boldsymbol{\beta} - \boldsymbol{b}\|^2 + \lambda\|\boldsymbol{\beta}\|_1\}$ where $\boldsymbol{b} = \boldsymbol{W}_{11}^{-1/2}\boldsymbol{s}_{12}$, this gives a $(N-1)$ -vector solution $\boldsymbol{\beta}$:
     (c) Fill in the corresponding row and column of $\boldsymbol{W}$ using $\boldsymbol{w}_{12} = \boldsymbol{\beta}\boldsymbol{W}_{11}$.
  3. Finally, using the notation of step 2(a) for $\mathbf{\Theta}$, for each $j$, first recover $\theta_{22}$ from the equation $1/\theta_{22} = w_{22} - \boldsymbol{w}_{12}\top\boldsymbol{\beta}$ and then recover $\boldsymbol{\theta}_{12}$ from $\boldsymbol{\theta}_{12} = -\boldsymbol{\beta}\theta_{22}$.
```

At the hand of Algorithms 1 and 2 we outline Hill Climbing and Graphical Lasso. For a more detailed description—and explanation of the equalities in Glasso—we respectively refer the reader to Refs.[15,41]. The input of both algorithms consists of the dataset $\mathcal{D}$ (the sample correlation matrix $\mathbf{S}$ in Algorithm 2 is just $(1/(n-1))\mathcal{D}^\top\mathcal{D}$ for standardized variables) consisting of n independent samples of the multi Gaussian variable $\mathbf{X}$ and a score function to optimize. The output of HC is a DAG, whereas the output of Glasso is the estimated precision matrix $\mathbf{\Theta}$, which, in binary format, is the adjacency matrix of the associated undirected graph.

Hill Climbing simply visits all (or some; 'simple' Hill Climbing) neighbouring networks that differ on one edge of the current network (step 1) and moves then to the network with highest score—or directly to the first network found with better score in the case of simple HC (step 2). The algorithm stops when no neighbouring network has higher score than the current network (this could be at a local optimum). The optimized implementation of HC in BNLEARN that was used in this work uses score caching, score decomposability and score equivalence to reduce the number of duplicated tests. When datasets of higher dimensionality are addressed, other efficient versions of HC can be considered[42].

Glasso transforms the initial score function (Eq. 7) to a lasso problem and applies a coordinate descent approach to solve the problem: the algorithm fixes all dependencies in the current estimate of the correlation matrix $\boldsymbol{W}$ except those of one variable (coordinate), i.e. except one column and row (step 2a). Then it estimates the dependencies of this variable that best solves the element wise lasso regression problem (step 2b) and fills in the corresponding row and column in the updated correlation matrix $\boldsymbol{W}$ (step 2c). Next, it moves to the next coordinate and solves the same problem, this time with the former solution integrated in the fixed coordinates (integrated in $\boldsymbol{W}_{11}$). This process (step 2) is repeated until convergence. Finally, in the last cycle, the row $\boldsymbol{\theta}_{12}$ and diagonal element θ_{22} in $\mathbf{\Theta}$ are recovered from $\boldsymbol{W}$ and $\boldsymbol{\beta}$ (step 3).

Transformation of probabilistic BN model to probabilistic PN model. The purple arrow in Fig. 1 indicates that BNs can be transformed into PNs. The transformation of the DAG into an undirected graph is done applying *moralization*, i.e. including an undirected edge between any two nodes with a common child, and subsequently dropping edge directions. Thus, each set of parents and children $\{X_i, \Pi_i\}$ is a fully connected subset in the moral graph. The mapping may not be perfect and the dependencies of the resulting undirected moral graph may typically be an I-map of the original DAG (i.e. the moral graph is a minimum superset of the dependencies of the DAG meaning that there is no other undirected graph with less edges including all DAG dependencies but also that the inclusion of extra links and parameters in the transformation process slightly distorts the *in*-dependence structure of the network).

The moral graph of a BN can be asociated with more than one undirected PNM. To associate the moral graph with the special case of a probabilistic PN model that encodes the JPD formulated in Eq. (4), i.e. a *pairwise* Markov network, the parameterset of the initial BN has to be transformed in order to obtain the precision matrix. The following equality between the precision matrix $\boldsymbol{\Theta}$ and the parameters $(\boldsymbol{\beta}, \boldsymbol{\nu})$ of a Gaussian Bayesian Network holds[43]:

$$\boldsymbol{\Theta} = \boldsymbol{\Theta}(\boldsymbol{\beta}, \boldsymbol{\nu}) = (\boldsymbol{I} - \boldsymbol{B})\boldsymbol{\nu}^{-1}(\boldsymbol{I} - \boldsymbol{B})^{\top}. \tag{12}$$

The new weights of the edges and parameters of the *pairwise* Markov Network are the entries of the precision matrix:

$$\theta_{ij} = \theta_{ji} = -\frac{\beta_{ij}(1-\beta_{jj})}{\nu_j} - \frac{\beta_{ji}(1-\beta_{ii})}{\nu_i} + \sum_{k\neq i,j} \frac{\beta_{ik}\beta_{jk}}{\nu_k}. \tag{13}$$

The entry θ_{ij} is zero if there is no edge i – j in the moral graph. Occasionally, θ_{ij} can take the value of zero as a result of the matrix summation at the right hand side of Eq. (13).

In this work we moralize and obtain the precision matrix of all BNs that were learned with the Hill Climbing algorithm. In practice we use the R-packages BNLEARN for the process of moralization and SPARSEBNUTILS[44] for the extraction of the precision matrix.

Evaluation measures. Following[2], we define the recall of a PNM with edge size |E| as

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{P},$$

where TP is the number of true positives in the PNM, FN the number of false negatives in the PNM and P is the number of positives/edges in the directed RegulonDB network or the in silico network, i.e. $P = 3387$ and $P = 4012$, respectively. Moreover, we define the precision of a PNM with edge size |E| as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{|\text{E}|},$$

where FP is the number of False Positives in the network.

Log-likelihood definition and calculation. The likelihood of the data $\mathscr{D}$, given a model $\mathscr{M}$ is the density of the data under the given model $\mathscr{M}$: $\text{P}(\mathscr{D}\,|\mathscr{M})$. For discrete density functions the likelihood of the data equals the probability of the data under the model. The likelihood is almost always simplified by taking the natural logarithm; continuous likelihood values are typically small and differentiation of the likelihood function (with the purpose of a maximum likelihood search) is often hard. Log-likelihood values can be interpreted equally when the expression is used for model comparison and maximum likelihood search as the natural logarithm is a monotonically increasing function.

In the following we explain the calculation of the log-likelihood $\mathscr{L}(\mathscr{D}\,|\mathscr{M}) = \log \text{P}(\mathscr{D}\,|\mathscr{M})$ for a PNM ($\mathscr{M} = \text{PNM}$) for a dataset $\mathscr{D}$ formed by n independent data realizations $\mathscr{D}_k, k \in \{1,\ldots,n\}$, of the N-dimensional random vector $\mathbf{X}$, with $\mathscr{D}_k = \{d_1^k \ldots d_N^k\}$ and d_i^k the k-th realization of variable $X_i \in \mathbf{X}$. We have

$$\begin{aligned}\log \text{P}(\mathscr{D}\,|\text{PNM}) &= \log \text{P}(\mathscr{D}_1,\ldots,\mathscr{D}_n|\text{PNM}) = \log \prod_{k=1}^{n} \text{P}(\mathscr{D}_k|\text{PNM}) \\ &= \sum_{k=1}^{n} \log \text{P}(\mathscr{D}_k|\text{PNM}) = \sum_{k=1}^{n} \log \text{P}_{\text{PNM}}(\mathscr{D}_k)\end{aligned} \tag{14}$$

with P_{PNM} the probability density function as modelled by the corresponding PNM with a Gaussian multivariate probability. In this work we consider two types of PNMs, precision and Bayesian PNMs, deduced from PNs and BNs graphs, respectively. In the case of a Gaussian PNM given by a PN we get:

$$\begin{aligned}\mathscr{L}_{\mathrm{PN}}(\mathscr{D}\ |\mathrm{PNM}_{\mathrm{PN}}) &= \sum_{k=1}^{n} \log \mathrm{P}(\mathscr{D}_k|\mathrm{PNM}_{\mathrm{PN}})\\ &= \sum_{k=1}^{n} \log\{(2\pi)^{-N/2} \det(\boldsymbol{\Theta})^{1/2}\\ &\quad \times \exp\{-1/2 \sum_{i}^{N} \theta_{ii}(d_i^k)^2 - \sum_{i<j} \theta_{ij} d_i^k d_j^k\}\}.\end{aligned} \tag{15}$$

Entries in the sum are evaluations of the multivariate normal density function and executed with the R-package MVTNORM[45].

In the case of a Gaussian PNM given by a BN, from Eq. (5), we have

$$\begin{aligned}\mathscr{L}_{\mathrm{BN}}(\mathscr{D}\ |\mathrm{PNM}_{\mathrm{BN}}) &= \sum_{k=1}^{n} \log \mathrm{P}(\mathscr{D}_k|\mathrm{PNM}_{\mathrm{BN}})\\ &= \sum_{k=1}^{n} \log \prod_{i=1}^{N} \mathrm{P}_i(X_i = d_i^k|\Pi_{X_i} = d_{\Pi_{X_i}}^k)\\ &= \sum_{k=1}^{n} \sum_{i=1}^{N} \log \mathrm{P}_i(X_i = d_i^k|\Pi_{X_i} = d_{\Pi_{X_i}}^k),\end{aligned} \tag{16}$$

where $d_{\Pi_{X_i}}^k$ is a subset of $\mathscr{D}_k$ containing the k-th data realization of the parentset Π_{X_i} of X_i. From Eq. (6) we know that the conditional univariate densities in the sum in Eq. (16) are univariate normal and we execute them with the basic R-package STATS.

Data availability

The in silico network and the simulated gene expression dataset analysed during the current study are available through the DREAM5 paper[2] in Supplementary data 1. The microarray dataset 'E_coli_v4_Build_6' and the TF-gene interaction network dataset 'network_tf_gene' are available in the Many Microbe Microarray databases[1], http://m3d.mssm.edu/norm/E_coli_v4_Build_6.tar.gz and in the Regulon Data Base[33], http://regulondb.ccg.unam.mx/menu/download/datasets/files/network_tf_gene.txt, respectively.

Received: 15 February 2022; Accepted: 6 October 2022
Published online: 04 November 2022

References

1. Faith, J. J. *et al.* Many Microbe Microarrays Database: Uniformly normalized Affymetrix compendia with structured experimental metadata. *Nucleic Acids Res.* **36**, D866–D870. https://doi.org/10.1093/nar/gkm815 (2008).
2. The DREAM5 Consortium *et al.* Wisdom of crowds for robust gene network inference. *Nat. Methods.* **9**, 796–804. https://doi.org/10.1038/nmeth.2016 (2012).
3. McCall, M. N. Estimation of gene regulatory networks. *Postdoc J. J. Postdoctoral Res. Postdoctoral Affairs* **1**, 60–69 (2013).
4. Delgado, F. M. & Gómez-Vela, F. Computational methods for Gene Regulatory Networks reconstruction and analysis: A review. *Artif. Intell. Med.* **95**, 133–145. https://doi.org/10.1016/j.artmed.2018.10.006 (2019).
5. de Jong, H. Modeling and simulation of genetic regulatory systems: A literature review. *J. Comput. Biol.* **9**, 67–103. https://doi.org/10.1089/10665270252833208 (2002).
6. Chen, T., He, H. L. & Church, G. M. Modeling gene expression with differential equations. in *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 29–40 (1999).
7. Zhang, B. & Horvath, S. A general framework for weighted gene co-expression network analysis. *Stat. Appl. Genet. Mol. Biol.* https://doi.org/10.2202/1544-6115.1128 (2005).
8. Friedman, N., Linial, M., Nachman, I. & Pe'er, D. Using Bayesian networks to analyze expression data. in *Proceedings of the fourth annual international conference on Computational molecular biology*, RECOMB '00, 127–135. https://doi.org/10.1145/332306.332355 (Association for Computing Machinery, 2000).
9. Xing, L. *et al.* An improved Bayesian network method for reconstructing gene regulatory network based on candidate auto selection. *BMC Genom.* **18**, 844. https://doi.org/10.1186/s12864-017-4228-y (2017).
10. Hartemink, A. J., Gifford, D. K., Jaakkola, T. S. & Young, R. A. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. in *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 422–433 (2001).
11. Pe'er, D., Regev, A., Elidan, G. & Friedman, N. Inferring subnetworks from perturbed expression profiles. *Bioinformatics* **17**, S215–S224. https://doi.org/10.1093/bioinformatics/17.suppl_1.S215 (2001).
12. Zou, M. & Conzen, S. D. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics* **21**, 71–79. https://doi.org/10.1093/bioinformatics/bth463 (2005).
13. Werhli, A. V., Grzegorczyk, M. & Husmeier, D. Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and bayesian networks. *Bioinformatics* **22**, 2523–2531. https://doi.org/10.1093/bioinformatics/btl391 (2006).
14. Dobra, A. *et al.* Sparse graphical models for exploring gene expression data. *J. Multivariate Anal.* **90**, 196–212. https://doi.org/10.1016/j.jmva.2004.02.009 (2004).
15. Friedman, J., Hastie, T. & Tibshirani, R. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441. https://doi.org/10.1093/biostatistics/kxm045 (2008).
16. Li, Y. & Jackson, S. A. Gene network reconstruction by integration of prior biological knowledge. *G3 Genes Genomes Genet.* **5**, 1075–1079. https://doi.org/10.1534/g3.115.018127 (2015).

17. Zhao, H. & Duan, Z.-H. Cancer genetic network inference using Gaussian graphical models. *Bioinform. Biol. Insights* **13**, 117793221983940. https://doi.org/10.1177/1177932219839402 (2019).
18. Mukherjee, S. & Hill, S. M. Network clustering: Probing biological heterogeneity by sparse graphical models. *Bioinformatics*. **27**, 994–1000. https://doi.org/10.1093/bioinformatics/btr070 (2011)
19. Epskamp, S. & Fried, E. I. A tutorial on regularized partial correlation networks. *Psychol. Methods*. **23**, 617–634. https://doi.org/10.1037/met0000167 (2018).
20. Perederiy, V. Bankruptcy Prediction Revisited: Non-Traditional Ratios and Lasso Selection. SSRN Scholarly Paper ID 1518084, Social Science Research Network, Rochester, NY (2009). https://doi.org/10.2139/ssrn.1518084.
21. Chan-Lau, J. A. Lasso Regressions and Forecasting Models in Applied Stress Testing (2017). Library Catalog. www.imf.org.
22. Zerenner, T., Friederichs, P., Lehnertz, K. & Hense, A. A Gaussian graphical model approach to climate networks. *Chaos Interdiscip. J. Nonlinear Sci.* **24**, 023103. https://doi.org/10.1063/1.4870402 (2014).
23. He, B. & Tan, K. Understanding transcriptional regulatory networks using computational models. *Curr. Opinion Genet. Development* **37**, 101–108. https://doi.org/10.1016/j.gde.2016.02.002 (2016).
24. Åkesson, J., Lubovac-Pilav, Z., Magnusson, R. & Gustafsson, M. ComHub: Community predictions of hubs in gene regulatory networks. *BMC Bioinform.* **22**, 58. https://doi.org/10.1186/s12859-021-03987-y (2021).
25. Barabási, A.-L. & Albert, R. Emergence of scaling in random networks. *Science*. **286**, 509–512. https://doi.org/10.1126/science.286.5439.509 (1999).
26. Liu, Q. & Ihler, A. Learning scale free networks by reweighted l1 regularization. In Gordon, G., Dunson, D. & Dudík, M. (eds.) *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15 of *Proceedings of Machine Learning Research*, 40–48 (PMLR, Fort Lauderdale, FL, USA, 2011).
27. McGillivray, A., Khalili, A. & Stephens, D. A. Estimating Sparse Networks with Hubs. arXiv:1904.09394 [math, stat] (2020).
28. Graafland, C. E., Gutiérrez, J. M., López, J. M., Pazó, D. & Rodríguez, M. A. The probabilistic backbone of data-driven complex networks: an example in climate. *Sci. Rep*. **10**, 11484. https://doi.org/10.1038/s41598-020-67970-y (2020).
29. Scutari, M., Graafland, C. E. & Gutiérrez, J. M. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *Int. J. Approximate Reason.* **115**, 235–253. https://doi.org/10.1016/j.ijar.2019.10.003 (2019).
30. Yu, D., Lim, J., Wang, X., Liang, F. & Xiao, G. Enhanced construction of gene regulatory networks using hub gene information. *BMC Bioinform.* **18**, 186. https://doi.org/10.1186/s12859-017-1576-1 (2017).
31. Koller, D. & Friedman, N. *Probabilistic Graphical Models: Principles and Techniques—Adaptive Computation and Machine Learning* (The MIT Press, 2009).
32. Marbach, D., Schaffter, T., Mattiussi, C. & Floreano, D. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *J. Comput. Biol.* **16**, 229–239. https://doi.org/10.1089/cmb.2008.09TT (2009).
33. Santos-Zavaleta, A. *et al.* RegulonDB v 10.5: Tackling challenges to unify classic and high throughput knowledge of gene regulation in E. coli K-12. *Nucleic Acids Res.* **47**, 212–220. https://doi.org/10.1093/nar/gky1077 (2019).
34. Peng, J., Wang, P., Zhou, N. & Zhu, J. Partial correlation estimation by joint sparse regression models. *J. Am. Stat. Assoc.* **104**, 735–746. https://doi.org/10.1198/jasa.2009.0126 (2009).
35. Shachter, R. D. & Kenley, C. R. Gaussian influence diagrams. *Manag. Sci.* **35**, 527–550 (1989).
36. Castillo, E., Gutiérrez, J. M. & Hadi, A. S. *Expert Systems and Probabilistic Network Models* (Springer Publishing Company, Incorporated, 1997).
37. Scutari, M. Learning Bayesian Networks with the bnlearn R Package. *J. Stat. Softw.* **35**, 1–22. https://doi.org/10.18637/jss.v035.i03 (2010).
38. Tan, K. M. *et al.* Learning graphical models with hubs. *J. Mach. Learn. Res. JMLR* **15**, 3297–3331 (2014).
39. Verma, T. & Pearl, J. Equivalence and synthesis of causal models. in *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, UAI '90, 255–270 (Elsevier Science Inc., 1991).
40. Spirtes, P., Glymour, C. & Scheines, R. *Causation, Prediction, and Search, Lecture Notes in Statistics* (Springer-Verlag, 1993).
41. Russell, S. J. & Norvig, P. *Artificial Intelligence: A Modern Approach* (Prentice Hall, 1995).
42. Gámez, J. A., Mateo, J. L. & Puerta, J. M. Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining Knowl. Discov.* **22**, 106–148. https://doi.org/10.1007/s10618-010-0178-6 (2011).
43. Aragam, B. & Zhou, Q. Concave penalized estimation of sparse Gaussian Bayesian networks. *J. Mach. Learn. Res.* **16**, 56 (2015).
44. Aragam, B., Gu, J. & Zhou, Q. Learning large-scale Bayesian networks with the sparsebn Package. *J. Stat. Softw*. **91**, 1–38. https://doi.org/10.18637/jss.v091.i11 (2019).
45. Genz, A. & Bretz, F. *Computation of Multivariate Normal and t Probabilities, Lecture Notes in Statistics* (Springer-Verlag, 2009).

Acknowledgements
C.E.G would like to acknowledge the support of the funding from the Spanish Agencia Estatal de Investigación through the Unidad de Excelencia María de Maeztu with reference MDM-2017-0765.

Author contributions
C.E.G. and J.M.G. conceived and designed the study. C.E.G. wrote the main manuscript text. J.M.G. revised the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to C.E.G.