Chapter 2

Statistical models of cellular networks

In this chapter I describe statistical models to visualize the correlations structure of genes. The methods can be distinguished by how deeply they purge influences of other genes from the observed correlations (section 2.1). The most prominent models are Bayesian networks (section 2.2). To learn them from data I discuss score based approaches in section 2.3. Section 2.4 reviews benchmarking of models and section 2.5 shows how my own approaches developed in the following chapters relate to recent developments in literature.

2.1 Conditional independence models

Let a set V of p network components be given. In probabilistic models we treat each component $v \in V$ as a random variable X_v and the set of all components in the model as a random vector $\mathbf{X} = (X_1, \ldots, X_p)$. The dataset M consists of N measurements, that is, realizations $\mathbf{x}^1, \ldots, \mathbf{x}^N$ of the random vector $\mathbf{X}$. We think of it as a $p \times N$ matrix with genes corresponding to rows and measurements to columns.

Network components are identified with nodes in a graph. The goal will be to find an edge set $\mathcal{E}$ representing the dependency structure of the network components. We will call the graph $T = (V, \mathcal{E})$ the topology of the cellular network. Depending on the model, T can be directed or undirected, cyclic or acyclic. In the important special case, where T is a directed acyclic graph (DAG), we call it D. The biological meaning of a "network component" depends on what kind of data we analyze. Most of the time it will be microarray data and the network is a transcriptional gene regulatory network. So, we will mostly speak of network components as genes. But the same methods can also be applied to protein data, even though only few examples can be found in literature [148, 68, 114].

2.1.1 Coexpression networks

Biological processes result from concerted action of interacting molecules. On this general observation builds a simple idea, which underlies the first approaches to cluster expression profiles [37, 126] and is still widely used in functional genomics. It is called the *guilt-by-association heuristic*: if two genes show similar expression profiles, they are supposed to follow the same regulatory regime. To put it more pointedly: coexpression hints at coregulation. Coexpression networks are constructed by computing a similarity score for each pair of genes. If similarity is above a certain threshold, the gene pair gets connected in the graph, if not, it remains unconnected. Wolfe *et al.* [147] argue that networks of coexpressed genes provide a widely applicable framework for assigning gene function. They show that coexpression agrees well with functional similarity as it is encoded in the Gene Ontology [5].

Building coexpression networks The first critical point in building a coexpression network is how to formalize the notion of similarity of expression profiles. Several measures have been proposed. The most simple similarity measure is correlation. In a Gaussian model, zero correlation corresponds to statistical independence. Correlation networks are easy to interpret and can be accurately estimated even if $p \gg N$, that is, the number of genes is much larger than the number of samples. Stuart *et al.* [133] build a graph from coexpression across multiple organisms (humans, flies, worms and yeast). They find many coexpression relationships to be conserved over evolution. This implies a selective advantage and thus functional relationship between these gene-pairs. Bickel [10] generalizes correlation networks to time series data by introducing a time-lag for correlation.

Correlation is a linear measure of independence, non-linear dependencies between genes are not necessarily found. This problem can be avoided using networks built from pair-wise mutual information [18]. Another flexible similarity measure are kernel-functions [116], which are extensively used in wide parts of Machine Learning. Yamanishi *et al.* [148] use kernel functions for supervised network reconstruction. They show that the kernel formalism gives a unified framework for integrating different types of data including expression profiles and protein-interaction graphs. Then, they tune kernel parameters in known parts of a protein-interaction graph and use them to infer unknown parts. Kato *et al.* [68] weight the different data sources according to noise and information content when combining them in the kernel.

When comparing different types of tissues, e.g., healthy cells versus tumor cells, it may be interesting to find genes highly correlated under one condition, but losing this correlation under the second condition. Kostka and Spang [70] call this behaviour *differential coexpression* and interpret it as gain or loss of a regulatory mechanism. They introduce a correlation-based method to identify sets of differentially coexpressed genes.

The second critical point is how to assess significance of results. Many pairs of genes will show similar behaviour in expression profiles by chance even though they are

not biologically related. A practical, though time-consuming strategy consists in permuting the data matrix and comparing the network obtained on real data with the distribution of similarity scores achieved in the permutations. Bickel [10] uses permutations to estimate the false discovery rate of spurious connections. In the supervised setting of Yamanishi *et al.* [148] cross-validation can be applied to choose optimal parameters.

Problems of coexpression based approaches Fig. 2.1 shows several reasons, why three genes X, Y and Z can be found to be coexpressed. We cannot distinguish direct from indirect dependencies by just looking at similar expression patterns. High similarity of expression tells us little about the underlying biological mechanisms.

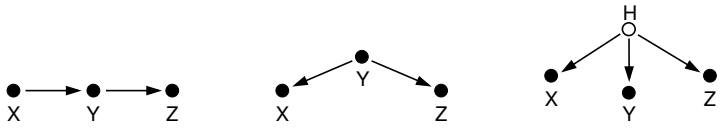

Figure 2.1: *Three reasons, why X, Y, and Z are coexpressed. They could be regulated in a cascade (left), or one regulates both others (middle), or there is a common "hidden" regulator (right), which is not part of the model.*

There are two possible solutions. Functional genomics has a long tradition of perturbing the natural state of a cell to infer gene-function from the observed effects. Interventions allow to decide between the three models in Fig. 2.1, because each one results in different predictions of effects, which can be compared to those obtained in experiments. Statisticians devised a different cure. Statistical methods search for correlations which cannot be explained by other variables. The theoretical background is the notion of *conditional independence*. Statistical methods filter out correlations, which can be attributed to other genes.

Conditional independence Conditional independence is defined as follows: Let X, Y, Z be random variables with joint distribution P. We say that X *is conditionally independent of* Y *given* Z (and write $X \perp Y \mid Z$) if and only if

$$P(X = x, Y = y \mid Z = z) \;=\; P(X = x \mid Z = z) \cdot P(Y = y \mid Z = z) \tag{2.1}$$

This is the same as saying

$$P(X = x \mid Y = y, Z = z) \;=\; P(X = x \mid Z = z)$$

and is a direct generalization of the independence condition for X and Y, namely,

$$P(X = x, Y = y) \;=\; P(X = x) \cdot P(Y = y).$$

The same definitions hold if conditioning is not on a single variable Z but on a set of variables $\mathbf{Z}$. For an interpretation, we can think of random variables as abstract pieces of knowledge obtained from, say, reading books [72]. Then $X \perp Y \mid Z$ means:

"Knowing Z, reading Y is irrelevant for reading X"; or in other words: "If I already know Z, then Y offers me no new information to understand X." Variable Z can explain the correlation between X and Y.

The statistical models we discuss in the following all build on conditional independence. To decide on an edge between X and Y in the graph, they ask questions of the form "Is X independent of Y given $\mathbf{Z}$?", but differ with respect to what $\mathbf{Z}$ stands for: either all other variables except for X and Y, or single third variables, or any subset of all the other variables. Coexpression networks can be seen as the special case $\mathbf{Z} = \emptyset$, which encodes marginal dependencies.

2.1.2 Full conditional models

Full conditional models ask: "Can the correlation observed between two genes be explained by *all other genes* in the model?" Nodes i and j are connected by an edge if and only if

$$X_i \not\perp X_j \mid \mathbf{X}_{\text{rest}}. \tag{2.2}$$

where "rest" denotes the set of all variables in V without i and j. Full conditional models become especially simple in a Gaussian setting. Assume that $\mathbf{X} \sim \mathrm{N}(\mu, \Sigma)$, where Σ is invertible. Let $K = \Sigma^{-1}$ be the *concentration matrix* of the distribution (also called the *precision matrix*). The value $-k_{ij}/\sqrt{k_{ii}k_{jj}}$ is called the *partial correlation coefficient* between genes i and j [72]. Then, it holds for $i, j \in V$ with $i \neq j$ that

$$X_i \perp X_j \mid \mathbf{X}_{\text{rest}} \quad \Leftrightarrow \quad k_{ij} = 0. \tag{2.3}$$

This relation is used to define Gaussian graphical models (GGMs) [72, 35]. A GGM is an undirected graph on vertex set V. To each vertex $i \in V$ corresponds a random variable $X_i \in \mathbf{X}$. The edge set of a GGM is defined by vanishing partial correlations. Vertices i and j are adjacent if and only if $k_{ij} \neq 0$. An example is shown in Fig. 2.2.

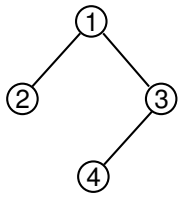

Figure 2.2: *Example of a full conditional model. Missing edges between nodes indicate independencies of the form $X_i \perp X_j \mid \mathbf{X}_{rest}$. We can read from the graph that $X_1 \perp X_4 \mid \{X_2, X_3\}$ and $X_2 \perp X_3 \mid \{X_1, X_4\}$ and $X_2 \perp X_4 \mid \{X_1, X_3\}$.*

The estimation of a GGM from data is a three-step process. First estimate the covariance matrix Σ, *e.g.*, by the sample covariance matrix $\hat{\Sigma} = \frac{1}{N-1}(M - \bar{M})(M - \bar{M})^T$, where $\bar{M}$ denotes the sample mean. Then, invert $\hat{\Sigma}$ to obtain an estimate $\hat{K}$ of the precision matrix K. Finally, employ statistical tests [72, 124, 33, 32] to decide, which entries in $\hat{K}$ are significantly different from zero.

Comparison to correlation networks Correlation graphs visualize the structure encoded in the correlation matrix Σ, which tells us about the similarity of expression

profiles. In GGMs, we model via the precision matrix $K = \Sigma^{-1}$, which tells us, how much correlation remains after we corrected for the influence of all other genes. GGMs not only filter out high correlations, which can be attributed to other genes, but may also draw attention to genes which are only very weakly correlated with a gene of interest, but highly related in terms of partial correlations in the context of the other neighboring genes in the GGM. These genes can be overlooked in correlation networks [30, 84].

GGMs have another clear advantage over correlation networks. Directly or indirectly, almost all genes will be correlated. Thus, the *correlation coefficient* is a weak criterion for dependence, but zero correlation is a strong indicator for independence. On the other hand, *partial correlation coefficients* usually vanish. They provide a strong measure of dependence and, correspondingly, only a weak criterion of independence [115].

Problems of GGMs Full conditional relationships can only be accurately estimated if the number of samples N is relatively large compared to the number of variables p. If the number of genes to be analyzed exceeds the number of distinct expression measurements (that is, if $p \gg N$), the correlation matrix of expression profiles between genes does not have full rank and cannot be inverted [115]. The $p \gg N$-situation is true for almost all genomic applications of graphical models. There are basically two ways out: either improve the estimators of partial correlations or resort to a simpler model. The basic idea in all of these approaches is that biological data are high-dimensional but *sparse*, in the sense that only a small number of genes will regulate one specific gene of interest. We end this section with examples of improved estimators and describe more strongly regularized models in the following section.

Several papers suggest ways to estimate GGMs in a $p \gg N$-situation. Kishino and Waddell [69] propose gene selection by setting very low partial correlation coefficents to zero. As they state, the estimate still remains unstable. Schäfer and Strimmer [115] improve all three steps of GGM construction. First they sample with replacement from the dataset to obtain many bootstrap [36] samples. Then, they estimate Σ by the mean covariance matrix achieved over all bootstrap replicates. Instead of the usual matrix inverse, they use the Moore-Penrose pseudoinverse, which is based on a singular value decomposition of $\hat{\Sigma}$ and can be applied also to singular matrices. Finally, they use false discovery rate multiple testing for the selection of edges to be included in the GGM.

2.1.3 First order conditional independence

First order conditional independence models ask: "Can the correlation between two genes be explained by a single third gene?" In contrast to GGMs, first order conditional independence models condition not on the whole rest, but only on single third

genes. Draw an edge between vertices i and j ($i \neq j$) if and only if the correlation coefficient $\rho_{ij} \neq 0$ and no third variable can explain the correlation:

$$X_i \not\perp X_j \mid X_k \quad \text{for all } k \in V \setminus \{i, j\}, \tag{2.4}$$

This general idea can be implemented in different ways: Basso *et al.* [7] build a model based on conditional mutual information. The resulting method is called ARACNe and was successfully applied to expression profiles of human B cells. In a Gaussian setting, first order conditional independence models were proposed by several authors [144, 145, 79, 27]. Testing for first order conditional independence involves only triples of genes at a time. Thus, the problem for GGMs in high dimensions no longer exists. Wille and Bühlmann [144] prove: if the full conditional independence graph (the GGM) contains no cycles, then the first order conditional independence graph coincides with the full conditional independence graph. Wille *et al.* [145] use sparse Gaussian graphical modelling to identify modules of closely related genes and candidate genes for cross-talk between pathways in the Isoprenoid gene network in *Arabidopsis thaliana*.

2.2 Bayesian networks

In the last sections we have seen methods to build graphs from

$$\begin{aligned} \text{marginal dependencies} \quad & X_i \not\perp X_j, \\ \text{full conditional dependencies} \quad & X_i \not\perp X_j \mid \mathbf{X}_{\text{rest}}, \\ \text{first order dependencies} \quad & X_i \not\perp X_j \mid X_k \quad \text{for all } k \in \text{rest}. \end{aligned}$$

The logcial next step is to ask for independencies *of all orders*. In the resulting graph, two vertices i and j are connected if *no subset* of the other variables can explain the correlation, that is, if

$$X_i \not\perp X_j \mid \mathbf{X}_S \quad \text{for all } S \subseteq V \setminus \{i, j\}. \tag{2.5}$$

This includes testing marginal, first order and full conditional independencies. Thus, the number of edges will be less compared to the models in the previous sections. The graph encoding independence statements of the form (2.5) for all pairs of nodes is still undirected. It can be shown that knowing independences of all orders gives a more advanced picture of correlation structure. The collection of independence statements already implies directions of some of the edges in the graph [96, 97, 127]. The resulting directed probabilistic model is called a *Bayesian network*.

Definition A (static) Bayesian network is a graphical representation of the dependency structure between the components of a random vector $\mathbf{X}$. The individual random variables are associated with the vertices of a directed acyclic graph (DAG) D, which describes the dependency structure. Each node is descibed by a local probability distribution (LPD) and the joint distribution $p(\mathbf{x})$ over all nodes factors as

$$p(\mathbf{x}) = \prod_{v \in V} p(x_v \mid \mathbf{x}_{pa(v)}, \theta_v), \tag{2.6}$$

where θ_v denotes the parametrization of the local distribution. The DAG structure implies an ordering of the variables. The parents of each node are those varibles that render it independent of all other predecessors. The factorization of the joint distribution in Eq. 2.6 is the key property of Bayesian networks. It allows to segment the set of variables into families, which can be treated individually. This basic definition of Bayesian networks poses a number of further questions, which will be answered in the following:

1. How do the local probability distributions $p(x_v \mid \mathbf{x}_{pa(v)}, \theta_v)$ look like?
2. How is conditional independence defined for DAGs?
3. How can we learn a Bayesian network structure from data?
4. Are there natural limits to structure learning?

Local probability distributions (LPDs) Bayesian network models differ with respect to assumptions on the local probability distributions $p(x_v|\mathbf{x}_{pa(v)}, \theta_v)$ attached to each node $v \in V$. Basically, there are two types of parametric LPDs used in practice: multinomial distributions for discrete nodes and Gaussian distributions (normal distributions) for continuous nodes. The general model in statistics is a mixture of a discrete and a continuous part. Additionally, there are approaches to use non-parametric regression models linking parents to children. In the following, we will shortly introduce each of these models.

- *Discrete LPDs.* A discrete node v with discrete parents $pa(v)$ follows a multinomial distribution:
$$X_v \mid \mathbf{x}_{pa(v)}, \theta_v \ \sim \ \text{Multin}(1, \theta_{v|\mathbf{x}_{pa(v)}}) \tag{2.7}$$
It is parametrized by a set of probability vectors $\theta_v = \{\theta_{v|\mathbf{x}_{pa(v)}}\}$, one for each configuration $\mathbf{x}_{pa(v)}$ of parents of v.
- *Gaussian LPDs.* A continuous node v with continuous parents $pa(v)$ follows a normal distribution:
$$X_v \mid \mathbf{x}_{pa(v)}, \theta_v \ \sim \ \text{N}(\mu_v, \sigma_v^2), \tag{2.8}$$
where the mean $\mu_v = \beta_v^{(0)} + \sum_{i \in pa(v)} \beta_v^{(i)} x_i$ is a linear combination of parent states. The normal distribution is parametrized by a vector $\theta_v = (\beta_v, \sigma_v^2)$ containing regression coefficients $\beta_v = (\beta_v^{(i)})_{i \in pa(v)}$ for each parent node and a variance for X_v.
- *Conditional Gaussian (CG) networks.* CG networks are a combination of discrete and Gaussian networks. Continuous nodes follow a Gaussian distribution and are allowed discrete and continuous parents, while discrete nodes follow a multinomial distribution and are restricted to discrete parents. Thus, the network can be divided into a completely discrete part and a mixed part containing discrete and continuous nodes. CG networks constitute the general class of graphical models studied in statistics [72].
- *Regression trees.* Segal *et al.* [119, 120] use regression trees as LPDs. These capture the local structure in the data [42, 21], whereas the DAG describes the global structure. Each regression tree is a rooted binary tree with parents in the DAG as

internal nodes. Each leaf node of the tree is associated to a univariate Gaussian distribution.

- *Non-parametric regression.* Instead of the parametric approaches discussed so far, the relationship between parents and children in the DAG can also be modeled by non-parametric regression models [64, 65, 66, 134]. The result is a non-linear continuous model. This is an advantage over multinomial or Gaussian Bayesian networks, which are either discrete or linear.
- *Boolean logic LPDs.* Bulashevska and Eils [16] constrain LPDs to noisy logic functions like OR, AND for activatory parent-child relations or NOR, NAND for inhibitory. This has the advantage of simplifying and regularizing the model, while at the same time making it easier to interpret.
- *Kinetic modeling.* Nachman *et al.* [89] use non-linear Michaelis-Mentens dynamics to model how the transcription rate of a gene depends on its regulators. This approach combines Bayesian networks with a biochemically realistic quantitative model of gene regulation.

Conditional independence in directed graphs In Fig. 2.2 we saw how to read off independence statements from a full conditional independence graph. How does this work in the case of Bayesian networks? The answer is given by the definition of *d-separation* [97] ("d" for directed). A path q in a DAG D is said to be d-separated (or blocked) by a set of nodes $\mathbf{S}$ if and only if at least one of the following two conditions holds:

1. q contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that the middle node m is in $\mathbf{S}$, or
2. q contains an inverted fork (or collider) $i \to m \leftarrow j$ such that the middle node m is **not** in $\mathbf{S}$ and such that no descendent of m is in $\mathbf{S}$.

If all paths between i and j are blocked by $\mathbf{S}$ then (and only then) holds $X_i \perp X_j \mid X_{\mathbf{S}}$. The three archetypical situations can be seen in Fig. 2.3. The definition of d-separation, also called the *Global Markov condition*, allows to read statements of statistical indepence off the DAG structure.

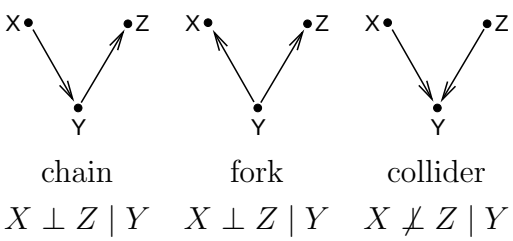

Figure 2.3: *The three archetypical situations in the definition of d-separation. In the chain and the fork, conditioning on the middle node makes the others independent. In a collider, X and Z are marginally independent, but get dependent once Y is known.*

Markov equivalence Many Bayesian networks may represent the same statements of conditional independence. They are statistically undistinguishable and we call them *Markov equivalent*. All equivalent networks share the same underlying undirect graph (called the *skeleton*) but may differ in the direction of edges, which are not

part of a *v-structure*, that is, a child with unmarried parents (same as a collider in Fig. 2.3). This was shown by Verma and Pearl [139]. It poses a theoretical limit on structure learning from data: even with infinitely many samples, we cannot resolve the structures in an equivalence class.

Acyclicity in a cyclic world Bayesian networks allow the highest resolution of correlation structure. Still, they suffer from a severe shortcoming: they are acyclic. With cycles, we cannot decompose the joint distribution as in Eq. 2.6. Biological networks are all known to contain feedback loops and cycles [4]. Modeling the cell cycle with an acyclic model [44] may not be the best idea. Fortunately, the cycle problem can be solved by assuming that the system evolves over time. This is shown in Fig. 2.4. We no longer model a static random vector $\mathbf{X}$ but a time series $\mathbf{X}[1], \ldots, \mathbf{X}[T]$ of observing $\mathbf{X}$ at T timepoints. If we assume that X_v at time $t+1$ can only have parents at time t, then cycles "unroll" and the resulting model is again acyclic and tractable: it is called a *Dynamic Bayesian network* (DBN) [45, 87]. DBNs found many applications in computational biology [154, 9, 157]. They are often combined with hidden variables [101], which can also capture non-transcriptional effects [8, 104, 105, 89, 93].

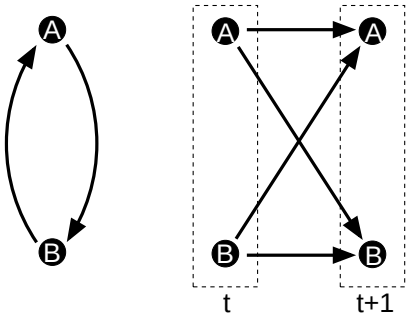

Figure 2.4: *The cycle unrolls into an acyclic graph over different time slices.*

2.3 Score based structure learning

In correlation networks, GGMs and sparse GGMs we use statistical tests for each gene pair to decide whether the data support an edge or not. The number of tests to be done in these models is limited, even though it can be big in the case of sparse GGMs. For Bayesian networks we would have to test independence of a gene pair for every subset of the other genes. This is called *constraint-based* learning of Bayesian networks. The examples discussed in [97, 127] involve only a handful of variables. For bigger problems testing gets infeasible very quickly. In applications in computational biology the network structure is thus mostly estimated by score based techniques.

2.3.1 Maximum likelihood scores

Maximum likelihood A straight-forward idea for model selection is to choose the DAG D, which allows the best fit to data M. This means maximizing the likelihood $p(M|D,\theta)$ as a function of θ. A score for DAG D is then given by

$$\text{score}_{ML}(D) \;=\; \max_{\theta}\; p(M|D,\theta) \tag{2.9}$$

Unfortunately, the likelihood is not an appropriate score to decide between models since it tends to overfitting. Richer models with more edges will provably have better likelihood than simpler ones. A standard solution to this problem is to penalize the maximum likelihood score according to model complexity. An often used example of this general strategy is scoring with the Bayesian information criterion.

Bayesian information criterion (BIC) Contrary to what the name suggests, the BIC score [117] is not a Bayesian score. It is a regularized maximum likelihood estimate, which penalizes the maximal likelihood of the model with respect to the number of model parameters to control overfitting. It is defined as

$$\text{score}_{BIC}(D) \;=\; \max_{\theta}\; p(M|D,\theta) - \frac{d}{2}\log N, \tag{2.10}$$

where d is the number of parameters. The BIC score can also be used to learn Bayesian networks with missing values or hidden variables. The likelihood has then to be maximized via the Expectation-Maximization (EM) algorithm. In such a scenario, the BIC score was used by Nachman *et al.* [89] to learn kinetic models of transcription factors and their targets. They treated protein activities and kinetic constants as hidden variables. In cases, where the likelihood is accessible to conjugate analysis, a full Bayesian approach is preferred over ML or BIC.

2.3.2 Bayesian scores

In Bayesian structure learning we evaluate the posterior probability of model topology D given data M:

$$\text{score}_{Bayes} \;=\; p(D|M) \;=\; \frac{p(M|D)\cdot p(D)}{p(M)} \tag{2.11}$$

The term $p(M)$ is an average of data likelihoods over all possible models. We do not need to compute it for relative model scoring. The term $p(D)$ is a prior over model structures. The main term is the marginal likelihood $p(M|D)$, which equals the full model likelihood averaged over parameters of local probability distributions, that is,

$$p(M|D) = \int_{\Theta} p(M|D,\theta)p(\theta|D)\;\mathrm{d}\Theta. \tag{2.12}$$

This is the reason, why the LPD parameters θ do not enter Eq. 2.11. They are treated as *nuisance parameters* and have been integrated out. It is important to note that the LPD parameters were not maximized as would be done in a maximum likelihood estimate or in a BIC score. Averaging instead of maximizing prevents the Bayesian score from overfitting.

Marginal likelihood of network structure The marginal likelihood $p(M|D)$ is the key component of Bayesian scoring metrics. Its computation depends on the choice of local probability distributions and local priors in the Bayesian network model. To

solve integral (2.12) analytically, the prior $p(\theta|D)$ must fit to the likelihood $p(M|D,\theta)$. Statistically, this fit is called "conjugacy". A prior distribution is called *conjugate* to a likelihood, if the posterior is of the same distributional form as the prior [49]. If no conjugate prior is available, the marginal likelihood has to be approximated. We shortly discuss the LPDs introduced in section 2.2.

- *Discrete LPDs.* The marginal likelihood for discrete Bayesian networks was first computed by Cooper and Herskovits [23]. It is further discussed by Heckerman *et al.* [58]. The conjugate prior for the multinomial distribution is the Dirichlet prior [49]. Assuming independence of the prior for each node and each parent configuration, the score decomposes into independent contributions for each family of nodes.
- *Gaussian LPDs.* Corresponding results exist for Gaussion networks using a Normal-Wishart prior [48]. The marginal likelihood again decomposes into node-wise contributions.
- *CG networks.* Conditional Gaussian networks are a mix of discrete and Gaussian nodes [11]. We discuss the computation of marginal likelihood in detail in section 3.4.2. Discrete and Gaussian marginal likelihoods are treated there as special cases.
- *Regression trees.* The marginal likelihood at each node of the DAG further splits into independent components for each leaf of the local regression tree. Conjugate analysis and analytic results are possible using normal-gamma priors for each leaf node [42, 21].
- *Non-parametric regression.* Conjugate analysis and analytic computation of the marginal likelihood are not possible. Imoto *et al.* [64] use a Laplace approximation to approach the true marginal likelihood.
- *Boolean logic LPDs.* Conjugate analysis and analytic computation of the marginal likelihood are not possible. Instead, Bulashevska and Eils [16] use Gibbs sampling to estimate the model posterior $p(D|M)$ and the parameter posterior $p(\theta|M)$.
- *Kinetic modeling.* Again, conjugate analysis is not possible. Nachman *et al.* [89] use the BIC score for model selection.

Likelihood equivalence It is sensible to postulate that DAGs in the same equivalence class get the same score. The score should not distinguish between undistinguishable models. This requirement limits the choice of permissible prior parameters when computing the marginal likelihood. We discuss here the discrete case of a multinomial node with a Dirichlet prior [58]. The Dirichlet parameters are a set $\{\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}}\}$, each element corresponding to a discrete node δ in state i_δ with discrete parent configuration $\mathbf{i}_{pa(\delta)}$. Likelihood equivalence constrains the Dirichlet parameters to the form

$$\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}} = \alpha \cdot P(I_\delta = i_\delta, \mathbf{I}_{pa(\delta)} = \mathbf{i}_{pa(\delta)}), \tag{2.13}$$

where P is a prior distribution over the joint states of node δ and its parents [58]. The scale parameter α of the Dirichlet prior—often interpreted as "equivalent sample size" or "prior strength"—is positive and independent of δ. It plays an important

role for regularization of network structure (see section 2.3.3). Two *ad hoc* choices are: for all $i_\delta \in \mathcal{I}_\delta$ and $\mathbf{i}_{pa(\delta)} \in \mathcal{I}_{pa(\delta)}$ set

$$\alpha_{i_\delta|\mathbf{i}_{pa(\delta)}} = \begin{cases} 1 & [23], \\ \alpha/|\mathcal{I}_\delta||\mathcal{I}_{pa(\delta)}| & [17]. \end{cases}$$

Both choices result in different scoring metrics. Heckerman *et al.* [58] call the first score the *K2 metric* after the K2 algorithm introduced in [23]. It is not likelihood equivalent. Heckerman calls the second score a *BDeu metric*. The name is an acronym for a *B*ayesian score using a *D*irichlet prior, which is likelihood *e*quivalent and *u*niform. It corresponds to the choice of a uniform prior in Eq. 2.13. How can likelihood equivalence be guaranteed generally? Heckerman *et al.* [58] and Geiger and Heckerman [48] introduce methods to deduce the parameter priors for all possible networks from one joint prior distribution in the discrete and continuous case, respectively. Bøttcher [11] generalizes the results to CG networks.

Structure prior Structure priors $p(D)$ help to focus inference on reasonable models by including biological prior knowledge or integrating different data sources. In some applications the task is not to learn a structure from scratch but to refine a prior network built from biological prior knowledge. The first idea is to restrict the search space to a—conveniently defined—vicinity $\mathcal{V}(\mathcal{P})$ of the prior network $\mathcal{P}$. All the DAGs in the restricted search space are considered equally likely. This can be interpreted as a rigid structure prior of the form

$$p(D) = \begin{cases} 1/|\mathcal{V}(\mathcal{P})| & \text{if } D \in \mathcal{V}(\mathcal{P}) \\ 0 & \text{else} \end{cases} \tag{2.14}$$

A smoother way to guarantee that DAGs similar to the prior network $\mathcal{P}$ get higher prior probability is the following. We measure the confidence of edge (v, w) by a value $0 < \kappa_{vw} \leq 1$. A structure prior can then be defined proportional to a product of weights κ_{vw} over all edges (v, w):

$$p(D) \propto \prod_{v,w \in V} \kappa_{vw}. \tag{2.15}$$

The normalization constant, which would be necessary to make the right-hand side a density, can be ignored when computing relative posterior probabilities. What are smart choices of κ_{vw}? There are several approaches suggested in literature, which are shortly described here.

1. Heckerman *et al.* [58] assume constant penalty $\kappa_{vw} \equiv \kappa$ for all edges, in which D and $\mathcal{P}$ differ. Thus, $p(D) \propto \kappa^\epsilon$ where ϵ is the number of edges in which D differs from the prior DAG $\mathcal{P}$.
2. Another approach [65, 134] uses a network prior in an iterative scheme. They construct a Bayesian network from microarray data, propose putative transcription factors from the network structure, and search for common motifs in the DNA

sequences of children and grand-children of transcription factors. Then, they re-learn the network by penalizing edges without motif evidence harder than edges with motif evidence.

3. Bernard *et al.* [9] define weights from p-values of binding location data. They assume that p-values follow an exponential distribution if the edge is present and a uniform distribution if it is not. By Bayes' rule they derive probabilities for an edge to be present given the p-values from the location data. The free parameter of the exponential distribution is then integrated out. The final probabilities $\mathcal{P}_{vw}$ are used as weights in a structure prior.

Fig. 2.5 shows a comparison of these three prior definitions. They can be organized by the weights κ_{vw} they give for the presence or absence of an edge given prior information in.

Prior $\mathcal{P}$

	D	
[58]	1	0
1	1	κ
0	κ	1

	D	
[65]	1	0
1	$e^{-\xi_1}$	1
0	$e^{-\xi_2}$	1

	D	
[9]	1	0
p-value	$\mathcal{P}_{vw}$	$1-\mathcal{P}_{vw}$

Figure 2.5: *Comparison of edge weights suggested by Heckerman* et al. *[58], Imoto* et al. *[65] and Bernard* et al. *[9]. Rows correspond to prior information. In the left two examples the prior can be described binary, on the right it is expressed as a p-value derived from a second data set. In the middle table holds $\xi_1 < \xi_2$, i.e. edges with motif evidence contribute more than edges without.*

Discretization Most often used in applications is the Bayesian score for discrete data. When learning gene regulatory networks from microarray data, we first need to preprocess the continuous gene expression values and discretize them. In general, discretization may be carried out for computational efficiency, or because background knowledge suggests that the underlying variables are indeed discrete. Discretizing continuous variables results in a loss of information. At the same time, this can be a loss of noise. Discretized data can be more stable with respect to random variations of the mRNA measurements. Several methods to discretize microarray data were proposed in literature:

1. Friedman *et al.* [44] discretize expression values into three categories, depending on whether the expression rate is significantly lower than, similar to, or greater than control, respectively.
2. Pe'er *et al.* [99] introduce an adaptive discretization procedure. They model the expression level of a gene in different experiments as samples from a mixture of normal distributions, where each normal component corresponds to a specific state. Then they use standard k-means clustering to estimate such a mixture.
3. Hartemink *et al.* [56] use a discretization coalescence method, which incrementally reduces the number of discretization levels for each gene while preserving as much total mutual information between genes as possible.

4. In the previous three approaches, expression levels were discretized before and independently of structure learning. Suboptimal discretization policies will lead to degraded network structure. To avoid this, Steck and Jaakkola [129] derive a scoring function to efficiently *jointly* optimize the discretization policy and the structure of the graphical model.

This section provides us with all the methodology we need to decide between candidate regulatory structures by Bayesian scoring. Once we have decided on a discretization policy and on the value of Dirichlet parameters, we need to compute the marginal likelihood of the data for every candidate structure. Biological prior knowledge can be incorporated via a structure prior to bias our choice towards reasonable models. Chapter 3 will give a detailed account of how to compute the marginal likelihood for discrete and Gaussian networks on observational and interventional data.

2.3.3 Regularization

Regularization is a technique used in Machine Learning to ensure uniqueness of solution and to fight overfitting by constraining admissible models [116, 83]. Regularization is always needed in $p \gg N$- situations. We already saw examples of regularization in section 2.1, when Gaussian graphical models were adapted to the $p \gg N$-situation [115, 144]. Different methods were proposed for Bayesian networks.

1. Steck and Jaakkola [128] show that a small scale parameter α in Eq. 2.13 leads to a strong regularization of the model structure and a sparse graph given a sufficiently large data set. In particular, the empty graph is obtained in the limit of a vanishing scale parameter. This is diametrically opposite to what one may expect in this limit, namely the complete graph from an unregularized maximum likelihood estimate.
2. Another way to regularize Bayesian networks is to constrain the forms, the local probability distributions can take. Bulashevska and Eils [16] suggest learning noisy logic gates for parent-child relationships. The drawback is that Bayesian conjugate analysis, which leads to the analytic solution of the marginal likelihood, is no longer possible and Gibbs sampling has to be applied.
3. Module networks [119, 120] constrain the number of parameters in the model by assuming that groups of genes (so called *modules*) share the same dependence on regulators. Learning module networks involves an iteration of assigning genes to modules and searching for dependencies between modules.

2.3.4 Model selection and assessment

Exhaustive search To search for the DAG with highest score is mathematically trivial: compute the score for every possible DAG and choose the one that achieves

the highest value. What makes exhaustive search computationally infeasible is the huge number of DAGs. The number of DAGs on n edges is

$$a_n \;=\; \sum_{k=1}^{n} (-1)^{k-1} \binom{n}{k} 2^{k(n-k)} \; a_{n-k} \tag{2.16}$$

with $a_0 = 1$ [108]. The number of DAGs increases explosively, as the first few steps in the recursion show: 1, 1, 3, 25, 543, 29 281, 3 781 503, 1 138 779 265. That means, we have to think of some heuristic strategy to find high-scoring Bayesian networks without enumerating all possible ones.

Defining search space First we need to decide how to describe models of interest. This defines the model space, in which we search for models describing the data well. To apply search heuristics we have to equip search space with a neighborhood relation, that is, operators to move from one point of the search space to the next one.

1. The most simple search space results from defining a neighborhood relation on DAGs. Two DAGs are neighbors if they differ by one edge, which is either missing in one of them or directed the other way round.
2. Madigan *et al.* [78] and Chickering [20] restrict the search space to Markov equivalence classes of DAGs which uniquely describe a joint distribution. Thus, no time is lost in evaluating DAG models which are equivalent anyway.
3. Friedman and Koller [43] search over orders of nodes rather than over network structures. They argue that the space of orders is smaller and more regular than the space of structures, and has a much smoother posterior landscape.

Search heuristics Most of the following search algorithms can be applied to all search spaces, even though they are usually applied to DAGs. They return a single best network.

1. A simple and fast but still powerful method is *hillclimbing* by greedy search. First, choose a point in search space to start from, e.g. a random graph or the empty graph. Compute the posterior probability for all graphs in the neighborhood of the current graph. Select the graph with highest score. Iterate until no graph in the neighborhood has a larger score than the current graph. This procedure gets you to local maxima of the Bayesian scoring metric. The K2-algorithm [23] is a variant of greedy search, which assumes that the order of nodes is known.
2. The *sparse candidate algorithm* [46] restricts the number of possible parents for each node by searching for pairs of nodes which are highly dependent.
3. The *ideal parent algorithm* [90, 89] constructs a parent profile perfectly explaining the child behaviour and uses it to guide parent selection and to restrict the search space.
4. Peña *et al.* [100] grow Bayesian networks starting from a target gene of interest. They iteratively add to the Bayesian network parents and children of all the genes

already included in it. The algorithm stops after a predefined number of steps and thus, intuitively, highlights the surrounding area of the seed gene without having to compute the complete Bayesian network over all genes.

5. Friedman [39, 40] introduces the *structural EM algorithm* to learn Bayesian networks in the presence of missing values or hidden variables. It is an extension of the Expectation-Maximization (EM) algorithm that performs structure search *inside* the EM procedure.

Assessing uncertainty The problem with optimal models is, as Edwards [35] puts it: "Any method (or statistician) that takes a complex multivariate dataset and, from it, claims to identify one true model, is both naive and misleading". The emphasis is on "one true model". Better than choosing a single best model is to explore the whole posterior distribution. Direct sampling from the posterior is impossible due to the intractability of the denominator in Eq. 2.11, but there are other methods available.

1. The most we know about the data distribution is the empirical distribution of observations in the dataset. A classical approach to assess variability in the data is bootstrapping [36]. The strategy is to sample with replacement from the observations in the data set to get a number of bootstrap datasets, and then learn a network on every bootstrap dataset. The relative frequency of network features in the resulting network structures can be used as a measure of reliability [44, 99].
2. Bootstrap samples can contain multiple copies of identical data points. This implies strong statistical dependencies between variables when given a small dataset. As a consequence, the resulting network structure can be considerably biased towards denser graphs. Steck and Jaakkola [131] propose a correction for this bias.
3. As a simple way to avoid the bootstrap-bias Steck and Jaakkola [129] use the *leave-k-out* method. Instead of resampling with replacement, k cases are left out of the dataset when estimating a model. Repeating this many times also gives an estimate of model variability.
4. Markov Chain Monte Carlo (MCMC) is a simulation technique, which can be used to sample from the posterior $p(D|M)$. Given a network structure, a new neighboring structure is proposed. This new structure is accepted with the Metropolis Hastings acceptance criterion [57]. The iteration of this procedure produces a Markov chain that under fairly general conditions converges in distribution to the true posterior. MCMC is used by Husmeier [62] to learn dynamic Bayesian networks. Madigan *et al.* [78] use MCMC over Markov equivalence classes and Friedman and Koller [43] over orders of nodes.

2.4 Benchmarking

Graphical models visualize a multivariate dependency structure. They can only answer biological questions if they succeed in reliably and accurately reconstructing bi-

ologically relevant features of cellular networks. Unfortunately, rigorous assessment and benchmarking of methods are still rare.

- One of the first evaluation studies is by Smith *et al.* [125]. They sample data from a songbird's brain model and report excellent recovery success when learning a Bayesian network from it.
- Zak *et al.* [155] develop a realistic 10 gene network, where the biological processes at the different levels of transcription, translation and post-translational modifications were modeled with systems of differential equations. They show that linear and log-linear methods fail to recover the network structure.
- Husmeier [62] uses the same simulation network [155] to specify sensitivity and specificity of dynamic Bayesian networks. He demonstrates how the network inference performance varies with the training set size, the degree of inadequacy of prior assumptions, and the experimental sampling strategy. By analyzing ROC curves Husmeier can show fair performance of DBNs.
- Wimberly *et al.* [146] test 10 algorithms, including Boolean and Bayesian networks, on a simulation [14] of the genetic network of the sea urchin embryo [25]. They report that reconstruction is unreliable with all methods and that the performance of the better algorithms quickly degrades as simulations become more realistic.
- Basso *et al.* [7] show that their own method, ARACNe, compares favorably against static Bayesian networks on a simulated network with 19 nodes [154]—but only if the dataset includes several hundreds of observations. On the other hand, Hartemink [55] finds *dynamic* Bayesian networks to be even more accurate than ARACNe on the same dataset.

All in all the results are not promising. Graphical models from microarray data need a big sample size and capture only parts of biologically relevant networks. One reason for this shortcoming is that the models we discussed so far all use purely observational data, where the cellular network was not perturbed experimentally. In simulations [156, 82] and on real data [114] it was found that data from perturbation experiments greatly improve performance in network reconstruction. Thus, the following section 3 will introduce methodology for learning from effects of interventions in a probabilistic framework suitable to capture the noise inherent in biological experiments. This helps to improve the accuracy of network reconstruction.

2.5 A roadmap to network reconstruction

Fig. 2.6 organizes network reconstruction methods with respect to basic questions: Does the data include gene knockout or knockdown experiments? If not, we call it *purely observational data*; if yes, we call it *interventional data*. Is the model probabilistic or deterministic? Does the model allow for changes over time? If yes, we call it *dynamic*, else *static*. Does the model describe transcriptional regulatory networks? And if yes: are additional non-transcriptional effects taken into account?

In the leaf nodes of the decision tree methods fall together that are methodologically similar. Some branches in the tree are missing. Mostly, the reason is not that it would be impossible to follow them, but simply that we found no approach doing it.

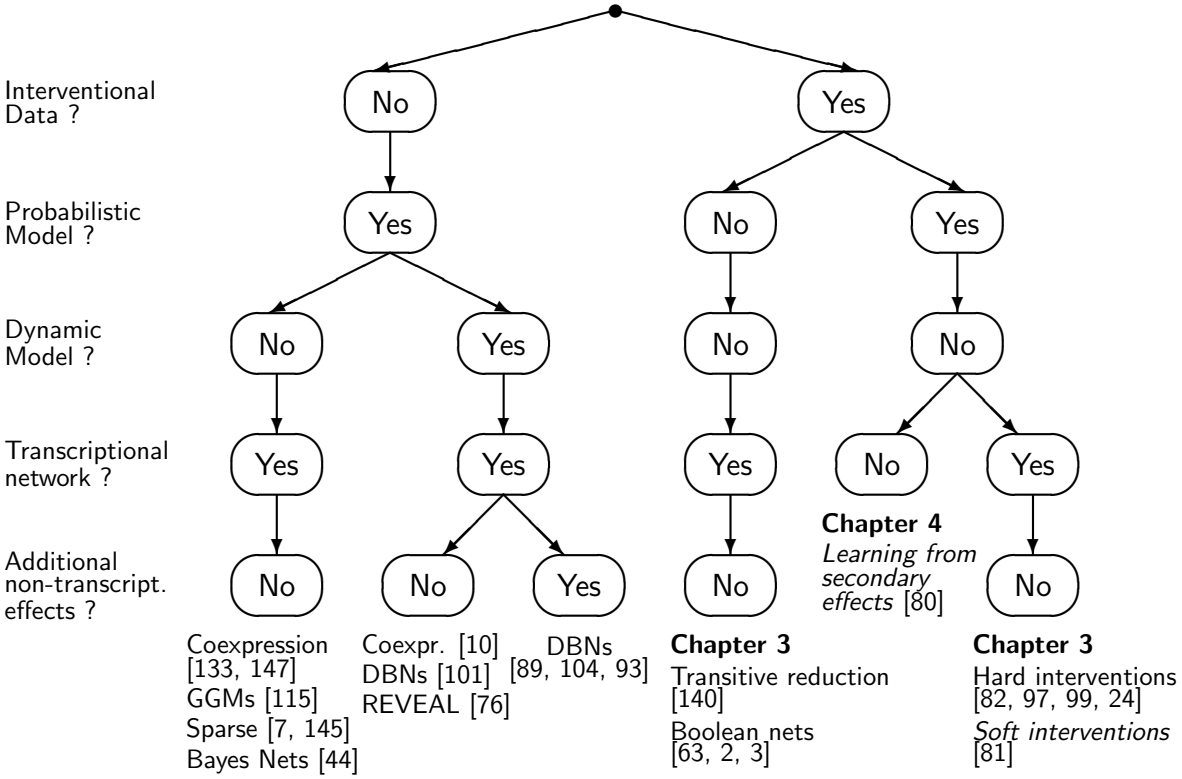

Figure 2.6: *A guide to the literature on network reconstruction. The methods discussed in this section all fall into the left branch of the tree. The next two sections will deal with learning transcriptional regulatory networks and non-transcriptional pathways from interventions. The main contributions of this dissertation are* soft interventions *and* learning from secondary effects.

Fig. 2.6 shows representative examples and relates our own methods to other approaches. The main contributions of this dissertation are soft interventions and learning from secondary effects. They can be found in the right-most branch of the tree. Both are static probabilistic models for interventional data. Soft interventions are used for gene regulation networks, in which effects of interventions can be observed at the other genes in the model. Learning from secondary effects infers non-transcriptional pathway features from expression data. This model expands the mRNA centered view of graphical models to non-transcriptional parts of signaling pathways.