CS-E5885 Modeling biological networks
Bayesian networks as models of biological networks

Harri Lähdesmäki

Department of Computer Science
Aalto University

February 8, 2022

Outline

- Bayes nets
- Joint probability factorization
- Inference using Bayes nets
- Dynamic Bayes nets
- Parameter inference
- Network structure inference
- Greedy hill-climbing and bootstrap
- Incorporating prior biological knowledge

- We will follow (loosely) Chapters 10 and 26 from (Murphy, 2012)

Motivation

- Sometimes we are not able to model the dynamics of a biological system because
  - The underlying biology/model is too complex
  - We do not have sufficient amount/quality time-series data
- In such cases we may be interested in modelling correlations, dependencies and independencies between molecular species

Motivation

- Sometimes we are not able to model the dynamics of a biological system because
  - The underlying biology/model is too complex
  - We do not have sufficient amount/quality time-series data
- In such cases we may be interested in modelling correlations, dependencies and independencies between molecular species
- In general, we would like to model the joint distribution of molecules involved in our system, i.e. $\mathbf{x} = (x_1, \ldots, x_V)$

$$p(\mathbf{x}|\theta) = P(x_1, \ldots, x_V|\theta) = P(x_{1:V}|\theta),$$

  where indexing $1 : V$ denotes $\{1, \ldots, V\}$
- Lets assume that each variable can take K different values, then we will need $K^V - 1$ parameters to represent this joint distribution

Motivation (2)

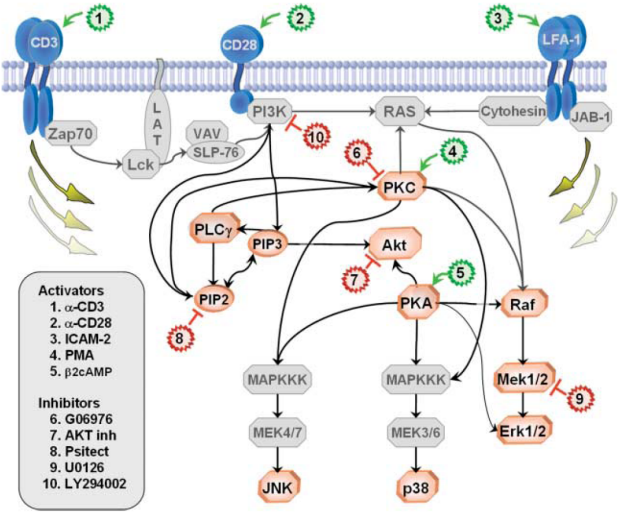

Figure: Figure from (Sachs et al., 2005)

Chain rule

- By the chain rule, we get

$$p(x_{1:V}|\theta) = p(x_1|x_{2:V}, \theta)p(x_2|x_{3:V}, \theta) \dots p(x_{V-1}|x_V, \theta)p(x_V|\theta)$$

- The number of parameters required to represent all conditional probability tables remains the same

Conditional independence

- Conditional independence is important for efficient representation of joint distributions
- Recall that two random variables X and Y are conditionally independent given Z iff

$$p(X, Y|Z) = p(X|Z)p(Y|Z)$$

and is denoted as

$$X \perp Y|Z$$

Conditional independence

- Conditional independence is important for efficient representation of joint distributions
- Recall that two random variables X and Y are conditionally independent given Z iff

$$p(X, Y|Z) = p(X|Z)p(Y|Z)$$

and is denoted as

$$X \perp Y|Z$$

- For example, if we assume $x_{t+1} \perp x_{t-1:1}|x_t$ we get the first-order Markov assumption
- The joint distribution for the first-order Markov is then

$$p(x_{1:V}) = p(x_1)\prod_{i=1}^{V} p(x_i|x_{i-1})$$

Graphical models

- Graphical model is a way to represent joint distributions by making conditional independency assumptions
  - Nodes of the graph represent random variables
  - Lack of edges encode conditional independencies
- Bayesian network is a directed graphical model whose graph is directed acyclic graph
  - Despite the name, there is nothing inherently Bayesian in Bayes nets
- Bayes nets are also sometimes called as causal nets
  - Bayes nets are not a causal model although they can be given a causal interpretation

Bayesian networks

- Given a set of random variables $\mathbf{x} = (x_1, \ldots, x_V)$, a Bayesian network is defined as a pair (G, θ), where
  - G is a directed acyclic graph (DAG), which is a graphical representation of the conditional independencies between variables in $\mathbf{x}$
  - θ is the set of parameters for the conditional probability distributions of these variables

Bayesian networks

- Given a set of random variables $\mathbf{x} = (x_1, \ldots, x_V)$, a Bayesian network is defined as a pair (G, θ), where
  - G is a directed acyclic graph (DAG), which is a graphical representation of the conditional independencies between variables in $\mathbf{x}$
  - θ is the set of parameters for the conditional probability distributions of these variables
- The conditional probabilities encode our knowledge about the state of X_i given all possible states of its parents
- For discrete-valued BNs we can represent the conditional probabilities as conditional probability tables (CPT)
  - Multinomial distributions
- For continuous-valued BNs, we use distributions

Topological ordering

- A key property of Bayesian networks is that nodes can be ordered such that parents of a node x_i, $\mathrm{pa}(x_i)$, come before the child x_i itself
- This is called a topological ordering
- For example:

$$p(x_{1:5}) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2, x_3)p(x_5|x_3)$$

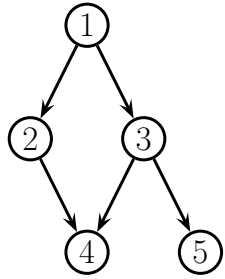

Figure: Figure from (Murphy, 2012)

Probability factorization

- In a Bayesian network, the joint probability factorizes as

$$p(\mathbf{x}) = \prod_{i=1}^{V} p(x_i | \mathrm{pa}(x_i))$$

  where $\mathrm{pa}(x_i)$ denotes the parents of node x_i in the graph G
- This probability factorization represents the conditional (in)dependencies of the variables
- Bayesian networks can represent any distribution over $\mathbf{x}$
- An example with discrete/binary-valued variables and CPTs/multinomial distributions

Probability factorization: example

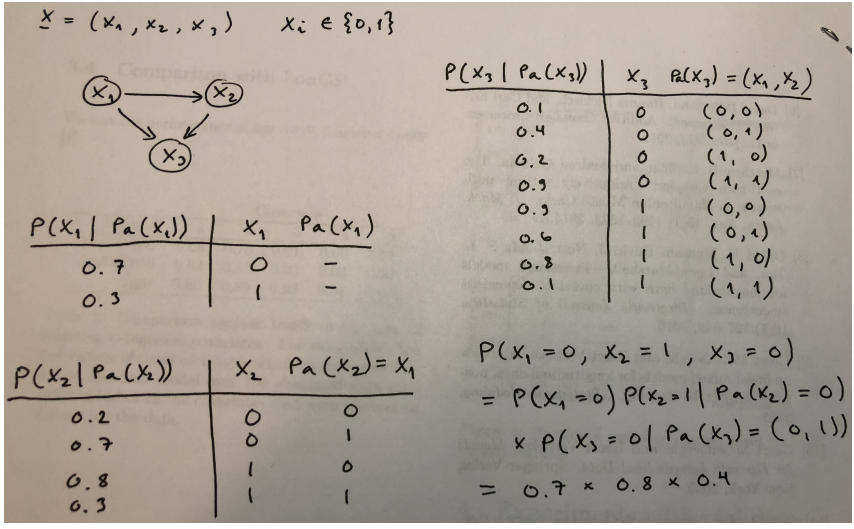

Figure: A three-node binary-valued Bayes net and probability factorization.

Inference using Bayes nets

- A key motivation for compact representation of joint distributions is to perform probabilistic inference
- Let us assume we have visible variables $\mathbf{x}_v$ and hidden variables $\mathbf{x}_h$ such that $\mathbf{x} = \mathbf{x}_v \cup \mathbf{x}_h$ (and $\mathbf{x}_v \cap \mathbf{x}_h = \emptyset$)

Inference using Bayes nets

- A key motivation for compact representation of joint distributions is to perform probabilistic inference
- Let us assume we have visible variables $\mathbf{x}_v$ and hidden variables $\mathbf{x}_h$ such that $\mathbf{x} = \mathbf{x}_v \cup \mathbf{x}_h$ (and $\mathbf{x}_v \cap \mathbf{x}_h = \emptyset$)
- Inference refers to computing the posterior probability of $\mathbf{x}_h$ given the value of $\mathbf{x}_v$

$$p(\mathbf{x}_h|\mathbf{x}_v, \theta) = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\theta)}{p(\mathbf{x}_v|\theta)} = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\theta)}{\sum_{h'} p(\mathbf{x}_{h'}, \mathbf{x}_v|\theta)}$$

- Example of the above equation on the next slide

Inference using Bayes nets

- A key motivation for compact representation of joint distributions is to perform probabilistic inference
- Let us assume we have visible variables $\mathbf{x}_v$ and hidden variables $\mathbf{x}_h$ such that $\mathbf{x} = \mathbf{x}_v \cup \mathbf{x}_h$ (and $\mathbf{x}_v \cap \mathbf{x}_h = \emptyset$)
- Inference refers to computing the posterior probability of $\mathbf{x}_h$ given the value of $\mathbf{x}_v$

$$p(\mathbf{x}_h|\mathbf{x}_v, \theta) = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\theta)}{p(\mathbf{x}_v|\theta)} = \frac{p(\mathbf{x}_h, \mathbf{x}_v|\theta)}{\sum_{h'} p(\mathbf{x}_{h'}, \mathbf{x}_v|\theta)}$$

- Example of the above equation on the next slide
- A straightforward summation will not be sufficiently efficient for large systems
  - Efficient algorithms exist

Inference using Bayes nets: example

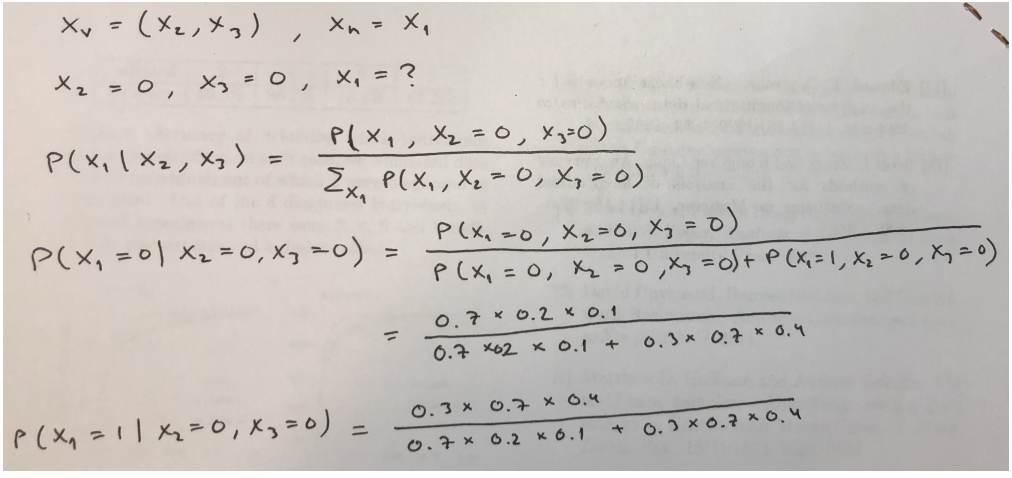

Figure: An illustration of Bayes net inference for a three-node, binary-valued, partly-observed Bayes net.

Dynamic Bayesian networks

- Note that nowhere in the previous formulation was there any mention of time t
  - Bayesian networks, by default, are static — they do not consider time or causality but only conditional (in)dependencies
  - Static networks, including Bayesian networks, are directed acyclic graphs (DAGs), which can be too restrictive models for biological networks
- Dynamic Bayesian Networks (DBNs) are temporal extensions of BNs
  - The probability factorization is performed for a discrete-time stochastic process $\mathbf{x}(t) = (x_1(t), \ldots, x_V(t))^T$, $t = 0, 1, 2, \ldots$

Dynamic Bayesian networks (2)

- In the simplest case, we assume the process can be modeled as an unrolled version of a standard static Bayesian network
  - Parents of each node $x_i(t)$, $\mathrm{pa}(x_i(t))$, are among the nodes at the previous time slice $\mathbf{x}(t-1)$
  - Process becomes a first order process
  - For discrete-valued networks, this corresponds to a discrete-time, discrete-state Markov chain
- Note that the underlying directed graph remain acyclic
- Regardless of the restriction of static Bayes nets, both static and dynamic networks can be considered for e.g. gene regulation

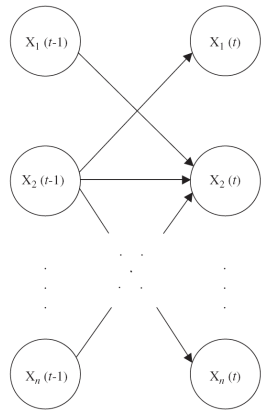

Figure: An illustration of the DBN model structure

Dynamic Bayesian networks (3)

- In a first-order DBN, the probability factorization for a time series of length T can be written as

$$p(\mathbf{x}(1), \ldots, \mathbf{x}(T)) = p(\mathbf{x}(1)) \prod_{t=2}^{T} p(\mathbf{x}(t)|\mathbf{x}(t-1))$$

Dynamic Bayesian networks (3)

- In a first-order DBN, the probability factorization for a time series of length T can be written as

$$\begin{aligned} p(\mathbf{x}(1), \ldots, \mathbf{x}(T)) &= p(\mathbf{x}(1)) \prod_{t=2}^{T} p(\mathbf{x}(t)|\mathbf{x}(t-1)) \\ &= p(\mathbf{x}(1)) \prod_{t=2}^{T} \prod_{i=1}^{V} p(x_i(t)|\mathrm{pa}(x_i(t))), \end{aligned}$$

where the parents of $x_i(t)$ represent the conditional (in)dependencies between the consecutive time steps

Bayes net modeling challenges

- After observing a dataset, denoted by D, we may want to learn a graphical model
  - A problem commonly encountered in biology
  - Estimate parameters θ for conditional dependencies interest, given a network structure G (easier)
  - Estimate the structure of the network, G (more difficult)
  - Estimate both structure and parameters

Learning from complete data

- Dataset is said to be complete if
  - There are no hidden variables
  - All the variables are fully observed

Learning from complete data

- Dataset is said to be complete if
  - There are no hidden variables
  - All the variables are fully observed
- Given a complete dataset $D = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ the likelihood of (static) BN is

$$p(D|\theta) = \prod_{k=1}^{N} p(\mathbf{x}_k|\theta)$$

Learning from complete data

- Dataset is said to be complete if
  - There are no hidden variables
  - All the variables are fully observed
- Given a complete dataset $D = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ the likelihood of (static) BN is

$$p(D|\theta) = \prod_{k=1}^{N} p(\mathbf{x}_k|\theta) = \prod_{k=1}^{N} \prod_{i=1}^{V} p(x_{ki}|\mathbf{pa}(x_{ki}|\theta_i))$$

Learning from complete data

- Dataset is said to be complete if
  - There are no hidden variables
  - All the variables are fully observed
- Given a complete dataset $D = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ the likelihood of (static) BN is

$$\begin{aligned} p(D|\theta) &= \prod_{k=1}^{N} p(\mathbf{x}_k|\theta) = \prod_{k=1}^{N}\prod_{i=1}^{V} p(x_{ki}|\mathbf{pa}(x_{ki}|\theta_i)) \\ &= \prod_{i=1}^{V}\prod_{k=1}^{N} p(x_{ki}|\mathbf{pa}(x_{ki}|\theta_i)) \end{aligned}$$

Learning from complete data

- Dataset is said to be complete if
  - There are no hidden variables
  - All the variables are fully observed
- Given a complete dataset $D = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ the likelihood of (static) BN is

$$
\begin{aligned}
p(D|\theta) &= \prod_{k=1}^{N} p(\mathbf{x}_k|\theta) = \prod_{k=1}^{N} \prod_{i=1}^{V} p(x_{ki}|\mathbf{pa}(x_{ki}|\theta_i)) \\
&= \prod_{i=1}^{V} \prod_{k=1}^{N} p(x_{ki}|\mathbf{pa}(x_{ki}|\theta_i)) \\
&= \prod_{i=1}^{V} p(D_i|\theta_i),
\end{aligned}
$$

where $D_i = x_{1:N,i}$ denotes the data associated to the ith variable and its parents

$\Rightarrow$ Optimization of the parameters can be done separately for each node

A discrete-valued Bayesian network model

- Even though the amount of mRNA or protein levels, for example, can vary in a scale that is most conveniently modeled as continuous (or countably infinite), we can still model the system by assuming that it operates with functionally discrete states
  - "activated"/"not activated" (2 states)
  - "under expressed"/"normal"/"over expressed" (3 states)

A discrete-valued Bayesian network model

- Even though the amount of mRNA or protein levels, for example, can vary in a scale that is most conveniently modeled as continuous (or countably infinite), we can still model the system by assuming that it operates with functionally discrete states
  - "activated"/"not activated" (2 states)
  - "under expressed"/"normal"/"over expressed" (3 states)
- Qualitative models can sometimes be learned even when the quality of the data is not sufficient for more accurate model classes
- As will be seen, with the discrete-valued observations the Bayesian network learning is relatively simple (in principle)
  - For now we assume here that the structure of the model is known

Summarizing the data

- Recall that likelihood factorizes over nodes $p(D|\theta) = \prod_{i=1}^{V} p(D_i|\theta_i)$
- Recall that for discrete-valued Bayes nets, local conditional probability distributions correspond to multinomial distributions
- Let N_{ijk} be the number of times we observe variable/node i in state k given parent node configuration j in data set D, i.e., $(x_i = k, \mathbf{pa}(x_i) = j)$
- For example, for a binary-valued network and a node x_i with 2 parents: $k \in \{0, 1\}$ and $j \in \{00, 01, 10, 11\}$

Summarizing the data

- Recall that likelihood factorizes over nodes $p(D|\theta) = \prod_{i=1}^{V} p(D_i|\theta_i)$
- Recall that for discrete-valued Bayes nets, local conditional probability distributions correspond to multinomial distributions
- Let N_{ijk} be the number of times we observe variable/node i in state k given parent node configuration j in data set D, i.e., $(x_i = k, \mathbf{pa}(x_i) = j)$
- For example, for a binary-valued network and a node x_i with 2 parents: $k \in \{0, 1\}$ and $j \in \{00, 01, 10, 11\}$
- Summarize the total number of observations for variable i with parent node configuration j,

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk},$$

where r_i denotes the number of possible values for variable i

Summarizing the data

- Recall that likelihood factorizes over nodes $p(D|\theta) = \prod_{i=1}^{V} p(D_i|\theta_i)$
- Recall that for discrete-valued Bayes nets, local conditional probability distributions correspond to multinomial distributions
- Let N_{ijk} be the number of times we observe variable/node i in state k given parent node configuration j in data set D, i.e., $(x_i = k, \mathbf{pa}(x_i) = j)$
- For example, for a binary-valued network and a node x_i with 2 parents: $k \in \{0, 1\}$ and $j \in \{00, 01, 10, 11\}$
- Summarize the total number of observations for variable i with parent node configuration j,

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk},$$

  where r_i denotes the number of possible values for variable i
- In frequentist setting, the well known ML estimate of multinomial probabilities is obtained by the normalized counts

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$$

*Bayesian parameter estimation

- For the Bayesian estimation, we need a parameter prior $p(\theta)$
  - A commonly used prior is the Dirichlet distribution
- Suppose the prior factorizes

$$p(\theta) = \prod_{i=1}^{V} p(\theta_i)$$

then

$$\begin{aligned} p(\theta|D) \propto p(D|\theta)p(\theta) &= \left(\prod_{i=1}^{V} p(D_i|\theta_i)\right)\left(\prod_{i=1}^{V} p(\theta_i)\right) \\ &= \prod_{i=1}^{V} (p(D_i|\theta_i)p(\theta_i)) \end{aligned}$$

- It can be shown that e.g. the posterior mean estimates are

$$\overline{\theta}_{ijk} = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}}$$

Bayes scoring of network strructures

- In Bayesian context, a natural score for a network structure G is the posterior probability given the observed data D:

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)},$$

  where
  - $P(D|G)$ is the marginal likelihood of the data
  - $P(G)$ encodes our priori knowledge of the network structure G
- Note that the "score" has exactly the same form as for other model classes previously

Bayes scoring of network strructures

- In Bayesian context, a natural score for a network structure G is the posterior probability given the observed data D:

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)},$$

  where
  - $P(D|G)$ is the marginal likelihood of the data
  - $P(G)$ encodes our priori knowledge of the network structure G
- Note that the "score" has exactly the same form as for other model classes previously
- Since probability $P(D)$ is not dependent on the structure, it is not needed to compare the scores of different networks
- What remains is thus

$$P(G|D) \propto P(D|G)P(G),$$

Learning the network structure

- We can obtain an analytically tractable form of the marginal likelihood for discrete-valued BNs (assuming Dirichlet parameter prior that factorizes over nodes and parent configurations):

$$
\begin{aligned}
P(D|G) &= \int_\theta p(D|\theta, G) p(\theta|G)\, d\theta \\
&= \dots \\
&= \prod_{i=1}^{V} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})} \\
&= \prod_{i=1}^{V} \text{score}_i
\end{aligned}
$$

Problems in practice

- The number of different static or dynamic Bayes network structures is super-exponential
  - The number of networks increases as $2^{(n^2)}$ for dynamic Bayesian networks, where n is the number of variables
- Finding the structure with maximal Bayes score is an NP hard problem even if we set a bound $k > 1$ for the maximum number of parents
- Thus, an exhaustive search and scoring approach for the different models will not work in practice (assuming more than a handful of variables)
  - Heuristics can be used to e.g. add parents to a node one at a time as long as the Bayesian score increases
  - More sophisticated Markov chain Monte Carlo methods

Problems in practice

- The number of different static or dynamic Bayes network structures is super-exponential
  - The number of networks increases as $2^{(n^2)}$ for dynamic Bayesian networks, where n is the number of variables
- Finding the structure with maximal Bayes score is an NP hard problem even if we set a bound $k > 1$ for the maximum number of parents
- Thus, an exhaustive search and scoring approach for the different models will not work in practice (assuming more than a handful of variables)
  - Heuristics can be used to e.g. add parents to a node one at a time as long as the Bayesian score increases
  - More sophisticated Markov chain Monte Carlo methods
- In small sample settings the a posteriori distribution may be rather flat
  - Looking for a single optimal model is not a good idea — we should consider the entire distribution, or in practice, several models with a good fit

Greedy hill-climbing learning

- An ad hoc method for computationally efficient learning for large networks
- Define the neighborhood of a network structure G, $\mathrm{nbd}(G)$, as the set of networks structures that can be reached from G by
  - Adding
  - Deleting
  - Reversing

  a single edge such that the acyclicity constraint is not violated
- Define a score of a network structure G, $\mathrm{score}(G)$, e.g. as
  - Posterior probability $P(G|D)$
  - Cross-validated predictive performance
  - ...

Greedy hill-climbing learning

- An ad hoc method for computationally efficient learning for large networks
- Define the neighborhood of a network structure G, $\mathrm{nbd}(G)$, as the set of networks structures that can be reached from G by
  - Adding
  - Deleting
  - Reversing

  a single edge such that the acyclicity constraint is not violated
- Define a score of a network structure G, $\mathrm{score}(G)$, e.g. as
  - Posterior probability $P(G|D)$
  - Cross-validated predictive performance
  - ...

1. Set $i = 0$ and choose an initial network structure $G^{(i)}$
2. For each $G' \in \mathrm{nbd}(G^{(i)})$ compute $\mathrm{score}(G')$
3. $G^{(i+1)} = \arg\max_{G' \in \mathrm{nbd}(G^{(i)})} \mathrm{score}(G')$
   - If $\mathrm{score}(G^{(i+1)}) > \mathrm{score}(G^{(i)})$, set $i := i + 1$ and go back to 2
   - Else $G_{\mathrm{opt}} = G^{(i)}$

Bootstrapping for network structure learning

- An (ad hoc) method for identifying a set of networks that have a high score (i.e. fit the data well)
- Given a data set $D = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, define a bootstrap sample of D as a set of n data points chosen from D with replacement, $D^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*)$

Bootstrapping for network structure learning

- An (ad hoc) method for identifying a set of networks that have a high score (i.e. fit the data well)
- Given a data set $D = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, define a bootstrap sample of D as a set of n data points chosen from D with replacement, $D^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_n^*)$

1. Generate b bootstrap samples, $D_1^*, \ldots, D_b^*$
2. For each of the bootstrap samples find the optimal network structure G_{opt}^* using e.g. the hill-climbing algorithm
3. This results in b high scoring network structures $G_{\text{opt},1}^*, \ldots, G_{\text{opt},b}^*$

Bayesian approach to structural properties

- Motivated by the fact the posterior distribution over all network structures tend to be flat (at least for small sample sizes), we can focus features that can be inferred reliably
- For example, we can define an indicator function f with value 1 if and only if the structure of the model contains an edge/path between nodes x_i and x_j

Bayesian approach to structural properties

- Motivated by the fact the posterior distribution over all network structures tend to be flat (at least for small sample sizes), we can focus features that can be inferred reliably
- For example, we can define an indicator function f with value 1 if and only if the structure of the model contains an edge/path between nodes x_i and x_j
- The posterior mean (or probability) of function f can be computed as

$$P(f|D) = \sum_{G \in \mathcal{G}} f(G)P(G|D)$$

where $\mathcal{G}$ contains all valid Bayes net structures

Bayesian approach to structural properties

- Motivated by the fact the posterior distribution over all network structures tend to be flat (at least for small sample sizes), we can focus features that can be inferred reliably
- For example, we can define an indicator function f with value 1 if and only if the structure of the model contains an edge/path between nodes x_i and x_j
- The posterior mean (or probability) of function f can be computed as

$$P(f|D) = \sum_{G \in \mathcal{G}} f(G) P(G|D)$$

where $\mathcal{G}$ contains all valid Bayes net structures

- This probability can be approximated e.g. by counting the number of times our chosen feature is present among the b bootstrapped optimal networks $G^*_{\text{opt},i}$, $i = 1, \ldots, b$

$$P(f|D) \approx \frac{1}{b} \sum_{i=1}^{b} f(G^*_{\text{opt},i})$$

Bayesian networks for signaling pathways

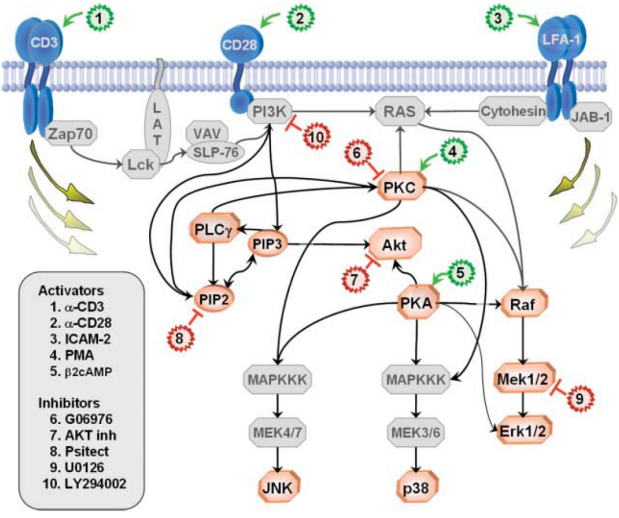

Figure: Figure from (Sachs et al., 2005)

Bayesian networks for signaling pathways (2)

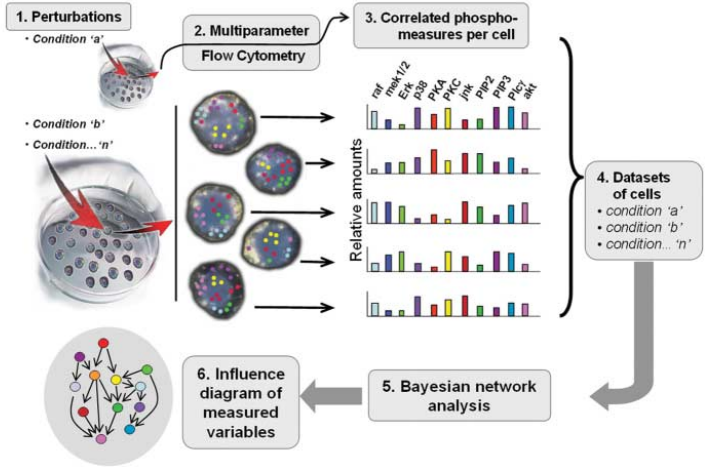

Figure: Figure from (Sachs et al., 2005)

Bayesian networks for signaling pathways (3)

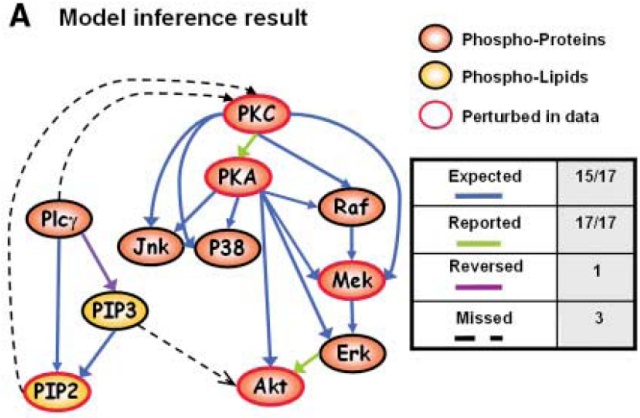

Figure: Figure from (Sachs et al., 2005)

Markov chain Monte Carlo

- Previously we have discussed about MCMC for parameter learning
- It turns out we can use the Metropolis-Hastings algorithm also to sample network structures
- The idea behind MCMC: generate a Markov chain over $\mathcal{G}$ such that its stationary distribution is the posterior $P(G|D)$

Markov chain Monte Carlo

- Previously we have discussed about MCMC for parameter learning
- It turns out we can use the Metropolis-Hastings algorithm also to sample network structures
- The idea behind MCMC: generate a Markov chain over $\mathcal{G}$ such that its stationary distribution is the posterior $P(G|D)$
- Need to define a proposal distribution $Q(G|G')$:
  - E.g. uniformly sample a network G' from the local neighborhood of G
- Accept new structure G' with

$$\alpha = \min\left\{1, \frac{P(G'|D)}{P(G|D)} \times \frac{Q(G|G')}{Q(G'|G)}\right\}$$

- Collect a (dependent) sample $\{G_{L+1}, G_{L+2}, \ldots, G_{L+S}\}$ where the first L samples are discarded as burn-in samples

Markov chain Monte Carlo

- Previously we have discussed about MCMC for parameter learning
- It turns out we can use the Metropolis-Hastings algorithm also to sample network structures
- The idea behind MCMC: generate a Markov chain over $\mathcal{G}$ such that its stationary distribution is the posterior $P(G|D)$
- Need to define a proposal distribution $Q(G|G')$:
  - E.g. uniformly sample a network G' from the local neighborhood of G
- Accept new structure G' with

$$\alpha = \min\left\{1, \frac{P(G'|D)}{P(G|D)} \times \frac{Q(G|G')}{Q(G'|G)}\right\}$$

- Collect a (dependent) sample $\{G_{L+1}, G_{L+2}, \ldots, G_{L+S}\}$ where the first L samples are discarded as burn-in samples
- Posterior probability of an edge can then be approximated with

$$P(f|D) \approx \frac{1}{S}\sum_{i=1}^{S} f(G_{L+i})$$

Markov chain Monte Carlo (2)

- MAP network structure for the signaling pathway example

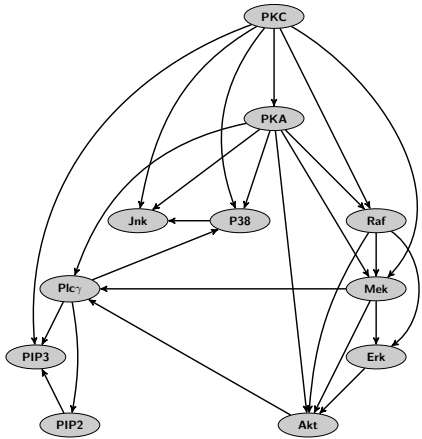

Figure: From (Larjo and Lähdesmäki, 2015)

Combining data with biological knowledge

- So far our network structure selection has relied on data/marginal likelihood (or other score) only
- Typically lots of biological knowledge exists that can be incorporated into the analysis via the (subjective) prior distribution $P(G)$

Combining data with biological knowledge

- So far our network structure selection has relied on data/marginal likelihood (or other score) only
- Typically lots of biological knowledge exists that can be incorporated into the analysis via the (subjective) prior distribution $P(G)$
- Exact form of prior information depends on a particular applications
- The following general formulation works in many applications assuming prior information about a directed edge between nodes i and j, E_{ij}, can be represented probabilistically as $P(E_{ij}|\mathcal{K}) = \beta_{ij}$

$$\begin{aligned} \log P(G) &\propto \sum_{E_{ij} \in G} \log P(E_{ij} \in G|\mathcal{K}) + \sum_{E_{ij} \notin G} \log P(E_{ij} \notin G|\mathcal{K}) \\ &= \sum_{E_{ij} \in G} \log \beta_{ij} + \sum_{E_{ij} \notin G} \log(1 - \beta_{ij}) \end{aligned}$$

where $\mathcal{K}$ denote prior knowledge and $P(E_{ij} \notin G|\mathcal{K}) = 1 - P(E_{ij} \in G|\mathcal{K}) = 1 - \beta_{ij}$

Combining data with biological knowledge

- Some examples of prior information
  - Signaling and protein-protein interaction networks: if interaction between a pair of proteins i and j has been reported previously, set $\beta_{ij} \approx 1$
  - Transcriptional networks: if a protein i has been observed to bind promoter of gene j, set $\beta_{ij} \approx 1$
  - Transcriptional networks: if promoter of gene j contains a DNA sequence motif that protein i recognizes and binds, set $\beta_{ij} \approx 1$
  - ...
- In the absence of prior knowledge, set β_{ij} to a value less than $1/2$ to prefer sparse networks

Causality via perturbations

- So-called perturbations can be used to complement the network structure learning
  - For transcriptional networks, knock-out genes related to areas of network which are most uncertain
  - In standard static BNs edge directionality is not definite: knock-out genes to test the causal direction
    - Knocking out TF should have effect on regulated genes
    - Knocking out regulated gene should not have effect on TF
  - Iterative approach, costly and time-consuming (requires wet-lab work between model building)
  - Sometimes other genes help replace the function of knock-out gene (robustness)

References

- Murphy K (2012) Machine learning: a probabilistic perspective, MIT Press.
- Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., & Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, Vol. 308, No. 5721, pp. 523-529.
- Larjo A and Lähdesmäki H, Using multi-step proposal distribution for improved MCMC convergence in Bayesian network structure learning, *EURASIP Journal on Bioinformatics and Systems Biology*, Vol. 6, 2015.