Inference in Bayesian Networks

Pablo M. Olmos, olmos@tsc.uc3m.es

Course on Bayesian Networks, November 2016

Index

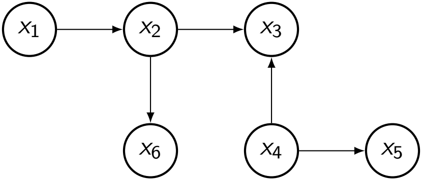
Section 1

Discrete Inference using Belief Propagation

Discrete Inference

- Let $X_j \in \mathcal{X}$, $j = 1, \ldots, 5$, be discrete R.V., where $K \doteq |\mathcal{X}|$
- Consider the following BN:

$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)p(x_4)p(x_3|x_2, x_4)p(x_5|x_4)p(x_6|x_2)$$



- **Inference**: evaluate the probability distribution over some set of variables, given the values of another set of variables.

Brute-force Inference

$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)p(x_4)p(x_3|x_2, x_4)p(x_5|x_4)p(x_6|x_2)$$

- Assume each variable is binary, i.e., $\mathcal{X} = \{0, 1\}$.
- For example, how can we compute $p(x_2|x_1 = 0)$?

Naive approach:

$$p(x_1 = 0, x_2) = \sum_{x_3, x_4, x_5, x_6} p(x_1 = 0, x_2, x_3, x_4, x_5, x_6) \qquad \text{[32 terms]}$$

$$p(x_1 = 0) = \sum_{x_2} p(x_1 = 0, x_2) \qquad \text{[2 terms]}$$

$$p(x_2|x_1 = 0) = \frac{p(x_1 = 0, x_2)}{p(x_1 = 0)} \qquad \text{[2 terms]}$$

The naive approach for discrete inference for n R.V. each taking K possible values has complexity $\mathcal{O}(K^n)$.

A more efficient approach

$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)p(x_4)p(x_3|x_2,x_4)p(x_5|x_4)p(x_6|x_2)$$

The Variable Elimination method exploits the factorization of $p(\boldsymbol{x})$ and distributive law to efficiently compute marginals.

$$p(x_1=0,x_2) = \sum_{x_3,x_4,x_5,x_6} p(x_1=0,x_2,x_3,x_4,x_5,x_6)$$

$$= p(x_1=0)p(x_2|x_1=0)\left(\sum_{x_6} p(x_6|x_2)\right)\left(\sum_{x_3,x_4} p(x_4)p(x_3|x_2,x_4)\sum_{x_5} p(x_5|x_4)\right)$$

$$= p(x_1=0)p(x_2|x_1=0)\underbrace{\sum_{x_3,x_4} p(x_4)p(x_3|x_2,x_4)}_{f(x_2)} \qquad \text{[8 terms]}$$

uc3m

Re-using computations

$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)p(x_4)p(x_3|x_2,x_4)p(x_5|x_4)p(x_6|x_2)$$

- Imagine we are also interested in computing $p(x_1 = 0, x_6)$.

$$p(x_1 = 0, x_6) = \sum_{x_2,x_3,x_4,x_5} p(x_1 = 0, x_2, x_3, x_4, x_5, x_6)$$

$$= p(x_1 = 0) \sum_{x_2} p(x_6|x_2)p(x_2|x_1 = 0) \underbrace{\left(\sum_{x_3,x_4} p(x_4)p(x_3|x_2,x_4) \right)}_{f(x_2)}$$

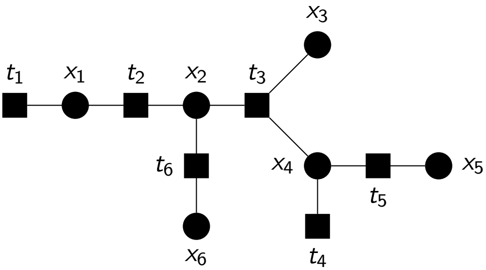
- Storing Intermediate computations (such as $f(x_2)$) makes infernce very efficient if multiple queries are made over $p(\boldsymbol{x})$.
- Belief Propagation!

uc3m

Factor graphs

- In inference, it's often easier to convert directed and undirected graphs into factor graphs.

$$\begin{aligned} p(\boldsymbol{x}) &= p(x_1)p(x_2|x_1)p(x_4)p(x_3|x_2,x_4)p(x_5|x_4)p(x_6|x_2) \\ &= t_1(x_1)t_2(x_1,x_2)t_3(x_2,x_3,x_4)t_4(x_4)t_5(x_4,x_5)t_6(x_2,x_6) \end{aligned}$$



- Variables nodes: we draw circles for each variable X_i in the distribution.
- Factor nodes: we draw filled dots for each factor t_j in the distribution.

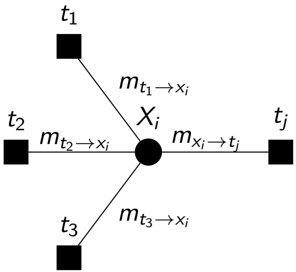
Belief Propagation (I)

- Local computations are regarded as *messages* between the nodes in the factor graph.
- Iterative message-passing algorithm.
- $m_{x_i \to t_j}(x_i)$ denotes the message sent from node X_i to factor node t_j.
- $m_{t_j \to x_i}(x_i)$ denotes the message sent from factor node t_j to variable node X_i.
- Both messages are indeed functions of X_i, namely each message is a stored table of $|\mathcal{X}|$ values.
- At each iteration, messages are updated following simple **update rules** according to a valid **schedule**.

Update rules (I)

Variable-to-factor message

$$m_{x_i \to t_j}(x_i) = \prod_{\substack{k \in \mathtt{Ne}(X_i) \\ k \neq j}} m_{t_k \to x_i}(x_i)$$

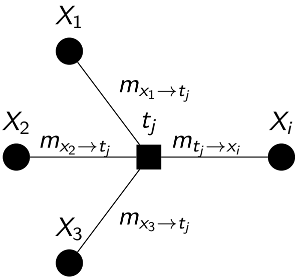


At variable nodes we simply multiply incomming messages.

Update rules (II)

Factor-to-variable message

$$m_{t_j \to x_i}(x_i) = \sum_{\boldsymbol{x}_j \sim x_i} t_j(\boldsymbol{x}_j) \prod_{\substack{m \in \mathtt{Ne}(t_j) \\ m \neq i}} m_{x_m \to t_j}(x_m)$$



At factor nodes we perform local marginalization.

Belief Propagation (II)

If factor graph is **cycle-free**:

- **Convergence guaranteed** after a few iterations.
- Upon convergence, variable marginals can be computed by multiplying incoming messages and normalize:

$$p_{x_i}(x_i) = \frac{1}{Z_i} \prod_{k \in \mathtt{Ne}(X_i)} m_{t_k \to x_i}(x_i),$$

  where the normalization constant is trivially obtained using the fact that the marginal must sum up to 1.

Belief Propagation (III)

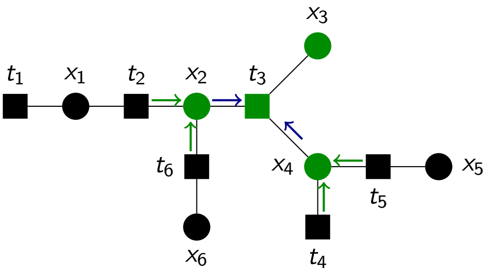
If factor graph is **contains cycles**:

- Loopy Belief Propagation: run propagation as if graph is simply connected.
- **Convergence not guaranteed** in general.
- Approximate results! Often works well in practice, unless the graph is very dense.
- Exact Inference is possible by transforming the graph into a cycle-free graph (E.g. by clustering variable nodes). **Junction Tree algorithm!**
- JTE complexity is in general very high unless the graph structure is very regular (e.g. computer vision applications).

Belief Propagation (IV)

Computing/estimating joint marginals $p(x_i, \boldsymbol{x}_{\mathrm{pa}_i})$ from BP messages:

- Multiply the factor $p(x_i|\boldsymbol{x}_{\mathrm{pa}_i})$ by all those messages coming to the cluster $(x_i, \boldsymbol{x}_{\mathrm{pa}_i})$ from the **rest of the factor graph**



$$
\begin{aligned}
p(x_2, x_3, x_4) &\propto t_3(x_2, x_3, x_4) m_{t_2 \to x_2}(x_2) m_{t_6 \to x_2}(x_2) m_{t_4 \to x_4}(x_4) m_{t_5 \to x_4}(x_4) \\
&= t_3(x_2, x_3, x_4) m_{x_2 \to t_3}(x_3) m_{x_4 \to t_3}(x_4)
\end{aligned}
$$

Section 2

Inference in Gaussian Linear BNs

Gaussian Linear BNs

- Let X_i, $i = 1, \ldots, n$, be real-valued R.Vs. such that

$$X_i = \sum_{k \in \mathrm{pa}_i} b_{ki}(X_k - \mu_k) + c_i,$$

where $c_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, $i = 1, \ldots, n$, are independent R.V.

- $\boldsymbol{X} = \boldsymbol{B}(\boldsymbol{X} - \boldsymbol{\mu}) + \boldsymbol{c}$, where $\boldsymbol{B}_{n \times n}$ can always be re-arranged to be upper-triangular (otherwise graph is not DAG).
- In other words,

$$X_i | \boldsymbol{x}_{\mathrm{pa}_i} \sim \mathcal{N}(m_i, \sigma_i^2), \qquad m_i = \mu_i + \sum_{k \in \mathrm{pa}_i} b_{ki}(x_k - \mu_k)$$

Exact Inference in Linear BNs

- $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where

$$\boldsymbol{\Sigma} = \mathbb{E}[(\boldsymbol{X} - \boldsymbol{\mu})^T(\boldsymbol{X} - \boldsymbol{\mu})] = (\boldsymbol{I} - \boldsymbol{B}^T)^{-1}\boldsymbol{D}(\boldsymbol{I} - \boldsymbol{B})^{-1}$$

  and $\boldsymbol{D} = \text{diag}(\sigma_i^2)$.

- **Naive approach**: $(\boldsymbol{I} - \boldsymbol{B}^T)^{-1} \rightarrow \mathcal{O}(n^3)$ complexity.

- **Gaussian Belief Propagation**: $\rightarrow \mathcal{O}(n_{\text{max}}^3)$ complexity, where n_{max} is the maximum number of variables connected to a factor node in the factor graph representation of $p(\boldsymbol{x})$.

Gaussian Linear BNs

- Sometimes a slightly different model is used. For $i = 1, \ldots, n$ we have

$$X_i = \sum_{k \in \mathrm{pa}_i} b_{ki} X_k + c_i$$

- $\boldsymbol{X} = \boldsymbol{B}\boldsymbol{X} + \boldsymbol{c}$
- $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with the same covariance matrix:

$$\boldsymbol{\Sigma} = \mathbb{E}[(\boldsymbol{X} - \boldsymbol{\mu})^T (\boldsymbol{X} - \boldsymbol{\mu})] = (\boldsymbol{I} - \boldsymbol{B}^T)^{-1} \boldsymbol{D} (\boldsymbol{I} - \boldsymbol{B})^{-1}$$

- To compute the mean elements a_i, $i = 1, \ldots, n$, we need to trasverse the graph in topological order (from parents to child nodes). Assuming a_k for $k \in \mathrm{pa}_i$ have been computed, then

$$a_i = \mu_i + \sum_{k \in \mathrm{pa}_i} b_{ki} a_k$$

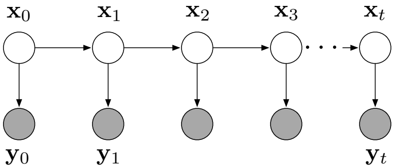
Hybrid discrete & Gaussian Linear BNs

- Certain class of probabilistic models containing both continuous and discrete R.V. where exact inference is analytically tractable.
- Assume we have n real-valued nodes X_i such that

$$X_i = \sum_{k \in \text{pa}_i} b_{ki}(Z_i) X_k + c_i,$$

  where both $b_{ki}(Z_i)$ and $c_i \sim \mathcal{N}\left(\mu_i(Z_i), \sigma_i^2(Z_i)\right)$ depend on Z_i **a discrete R.V.**.
- Inference is exact, we essentially average over a mixture of Gaussian terms, each given by a certain configuration of the vector $\boldsymbol{z}$. **Number of terms in the mixture grows exponentially fast with n in the general case.**
- **Good reference:** *Inference and Learning in Hybrid Bayessian Networks*, Kevin P. Murphy, Report, 1998 (public online).

Inference in Hidden markov models and Linear Gaussian state-space models



- Time-Series (speech processing, tracking, ...)
- In HMMs, the states X_t are discrete.
- In linear Gaussian SSMs, the states are real Gaussian vectors.
- Both HMMs and SSMs can be represented as singly connected DAGs.
- The forward–backward algorithm in hidden Markov models (HMMs), and the Kalman smoothing algorithm in SSMs are both instances of belief propagation.

Section 3

Approximate Inference

Approximate Inference

- Few representative cases where exact inference is possible. Yet it can be very slow.
  - Discrete inference is always possible (just performing sums), but the Junction Tree Algorithm can be cumbersome.
- In many other scenarios, exact inference is not even possible. We must therefore resort to **approximation techniques**.
  1. Variational methods & Mean Field approximations.
  2. Loopy Belief Propagation, Approximate Message Passing, Expectation Propagation.
  3. Sampling (Monte Carlo) methods. Importance Sampling.
  4. Monte Carlo Markov Chain (MCMC), and includes as special cases Gibbs sampling and the Metropolis-Hasting algorithm.
- Approximate Inference is **a huge topic**: see the references for more details.

Section 4

References

References

Books

- Christopher M. Bishop, Pattern Recognition and Machine Learning, Ed. Springer. Chapters 8-11.
- David Barber, Bayesian Reasoning and Machine Learning. Chapters 1-7.
- Kevin P. Murphy, Machine Learning: A Probabilistic Perspective. Chapters 10, 11, 18-24.

Articles:

- Kevin P. Murphy, A Brief Introduction to Graphical Models and Bayesian Networks, 1998 (online).
- Sam Roweis & Zoubin Ghahramani, 1999. A Unifying Review of Linear Gaussian Models, Neural Computation 11(2) (1999) pp.305-345
- C. Huang and A. Darwiche, 1996. İnference in Belief Networks: A procedural guide", Intl. J. Approximate Reasoning, 15(3):225-263.
- M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, 1997. .An introduction to variational methods for graphical models."
- D. Heckerman, 1996. .A tutorial on learning with Bayesian networks", Microsoft Research tech. report, MSR-TR-95-06.

uc3m

Section 5

Software

Software for Graphical Models

- BUGS and WinBUGS: inference via Gibbs sampling, not very scalable
- HUGIN: widely used, commercial, focus on exact inference
- **Kevin Murphy's Bayes Net Toolbox**: Matlab, widely used
- Microsoft's Infer.NET: advanced scalable libraries implementing factor graph propagation, EP, and variational message passing.
- Jeff Bilmes' GMTK: very good at HMMs and related time series models

A personal library

- My personal website `https://github.com/olmosUC3M/Inference-and-Learning-in-discrete-Bayesian-Networks.git`
- Not really scalable. The focus is on teaching through practice over small example using Python's notebooks.
- Easy to interpret and extend.
- Open Source.