Computational Complexity of Bayesian Networks

Johan Kwisthout and Cassio P. de Campos

Radboud University Nijmegen / Queen's University Belfast

UAI, 2015

Complexity theory

- Many computations on Bayesian networks are NP-hard
- Meaning (no more, no less) that we cannot hope for poly time algorithms that solve *all* instances
- A better understanding of complexity allows us to
  - Get insight in what makes particular instances hard
  - Understand why and when computations can be tractable
  - Use this knowledge in practical applications
- Why go beyond NP-hardness to find exact complexity classes etc.?
  - For exactly the reasons above!
- See lecture notes for detailed background at `www.socsci.ru.nl/johank/uai2015`

Today's menu

- We assume you know *something* about complexity theory
  - Turing Machines
  - Classes P, NP; NP-hardness
  - polynomial-time reductions
- We will build on that by adding the following concepts
  - Probabilistic Turing Machines
  - Oracle Machines
  - Complexity class PP and PP with oracles
  - Fixed-parameter tractability
- We will demonstrate complexity results of
  - Inference problem (compute $\Pr(\mathbf{H} = \mathbf{h} \mid \mathbf{E} = \mathbf{e})$)
  - MAP problem (compute $\arg\max_{\mathbf{h}} \Pr(\mathbf{H} = \mathbf{h} \mid \mathbf{E} = \mathbf{e})$)
- We will show what makes hard problems easy

Notation

- We use the following notational conventions
  - Network: $\mathcal{B} = (\mathbf{G}_{\mathcal{B}}, \Pr)$
  - Variable: X, Sets of variables: $\mathbf{X}$
  - Value assignment: x, Joint value assignment: $\mathbf{x}$
  - Evidence (observations): $\mathbf{E} = \mathbf{e}$
- Our canonical problems are SAT variants
  - Boolean formula ϕ with variables $X_1, \ldots, X_n$, possibly partitioned into subsets
  - In this context: quantifiers $\exists$ and MAJ
  - Simplest version: given ϕ, does there *exists* ($\exists$) a truth assignment to the variables that satisfies ϕ?
  - Other example: given ϕ, does the *majority* (MAJ) of truth assignments to the variables satisfy ϕ?

Hard and Complete

- A problem Π is *hard* for a complexity class C if every problem in C can be reduced to Π
- Reductions are polynomial-time *many-one* reductions
- Π is polynomial-time many-one reducible to Π′ if there exists a polynomial-time computable function f such that $x \in \Pi \Leftrightarrow f(x) \in \Pi'$
- A problem Π is *complete* for a class C if it is both in C and hard for C.
- Such a problem may be regarded as being ‘at least as hard’ as any other problem in C: since we can reduce any problem in C to Π in polynomial time, a polynomial time algorithm for Π would imply a polynomial time algorithm for *every* problem in C

P, NP, #P

- The complexity class P (short for *polynomial time*) is the class of all languages that are decidable on a deterministic TM in a time which is polynomial in the length of the input string x
- The class NP (*non-deterministic polynomial time*) is the class of all languages that are decidable on a *non*-deterministic TM in a time which is polynomial in the length of the input string x
- The class #P is a function class; a function f is in #P if $f(x)$ computes the number of accepting paths for a particular non-deterministic TM when given x as input; thus #P is defined as the class of counting problems which have a decision variant in NP

Probabilistic Turing Machine

- A *Probabilistic* TM (PTM) is similar to a non-deterministic TM, but the transitions are *probabilistic* rather than simply non-deterministic
- For each transition, the next state is determined stochastically according to some probability distribution
- Without loss of generality we assume that a PTM has two possible next states q_1 and q_2 at each transition, and that the next state will be q_1 with some probability p and q_2 with probability $1 - p$
- A PTM accepts a language L if the probability of ending in an accepting state, when presented an input x on its tape, is strictly larger than $1/2$ if and only if $x \in L$. If the transition probabilities are uniformly distributed, the machine accepts if the *majority* of its computation paths accepts

In BPP or in PP, that's the question

- PP and BPP are classes of decision problems that are decidable by a probabilistic Turing machine in polynomial time with a particular (two-sided) probability of error
- The difference between these two classes is in the probability $1/2 + \epsilon$ that a *Yes*-instance is accepted
  - *Yes*-instances for problems in PP are accepted with probability $1/2 + 1/c^n$ (for a constant $c > 1$)
  - *Yes*-instances for problems in BPP are accepted with a probability $1/2 + 1/n^c$
- PP-complete problems, such as the problem of determining whether the *majority* of truth assignments to a Boolean formula ϕ satisfies ϕ, are considered to be intractable; indeed, it can be shown that NP $\subseteq$ PP.
- The canonical PP-complete problem is MAJSAT: given a formula ϕ, does the majority of truth assignments satisfy it?

Summon the oracle!

- An Oracle Machine is a Turing Machine which is enhanced with an oracle tape, two designated oracle states q_{O_Y} and q_{O_N}, and an oracle for deciding membership queries for a particular language L_O
- Apart from its usual operations, the TM can write a string x on the oracle tape and query the oracle
- The oracle then decides whether $x \in L_O$ in a single state transition and puts the TM in state q_{O_Y} or q_{O_N}, depending on the 'yes'/'no' outcome of the decision
- We can regard the oracle as a 'black box' that can answer membership queries in one step.
- We will write $\mathcal{M}^{\mathsf{C}}$ to denote an Oracle Machine with access to an oracle that decides languages in C
- E.g., the class of problems decidable by a nondeterministic TM with access to an oracle for problems in PP is $\mathsf{NP}^{\mathsf{PP}}$

Fixed Parameter Tractability

- Sometimes problems are intractable (i.e., NP-hard) in general, but become tractable if some *parameters* of the problem can be assumed to be small.
- A problem Π is called fixed-parameter tractable for a parameter κ if it can be solved in time $\mathcal{O}(f(\kappa) \cdot |x|^c)$ for a constant $c > 1$ and an arbitrary computable function f.
- In practice, this means that problem instances can be solved efficiently, even when the problem is NP-hard in general, if κ is known to be small.
- The parameterized complexity class FPT consists of all fixed parameter tractable problems $\kappa-\Pi$.

INFERENCE

Have a look at these two problems:

EXACT INFERENCE
Instance: A Bayesian network $\mathcal{B} = (\mathbf{G}_{\mathcal{B}}, \Pr)$, where **V** is partitioned into a set of evidence nodes **E** with a joint value assignment **e**, a set of intermediate nodes **I**, and an explanation set **H** with a joint value assignment **h**.
Output: The probability $\Pr(\mathbf{H} = \mathbf{h} \mid \mathbf{E} = \mathbf{e})$.

THRESHOLD INFERENCE
Instance: A Bayesian network $\mathcal{B} = (\mathbf{G}_{\mathcal{B}}, \Pr)$, where **V** is partitioned into a set of evidence nodes **E** with a joint value assignment **e**, a set of intermediate nodes **I**, and an explanation set **H** with a joint value assignment **h**. Let $0 \leq q < 1$.
Question: Is the probability $\Pr(\mathbf{H} = \mathbf{h} \mid \mathbf{E} = \mathbf{e}) > q$?

What is the relation between both problems?

THRESHOLD INFERENCE is PP-complete

- Computational complexity theory typically deals with *decision problems*
- If we can solve THRESHOLD INFERENCE in poly time, we can also solve EXACT INFERENCE in poly time (why?)
- In this lecture we will show that THRESHOLD INFERENCE is PP-complete, meaning
  - THRESHOLD INFERENCE is in PP, and
  - THRESHOLD INFERENCE is PP-hard
- In the Lecture Notes we show that EXACT INFERENCE is #P-hard and in #P modulo a simple normalization
- #P is a counting class, outputting the number of accepting paths on a Probabilistic Turing Machine

THRESHOLD INFERENCE is in PP

- To show that THRESHOLD INFERENCE is in PP, we argue that THRESHOLD INFERENCE can be decided in polynomial time by a Probabilistic Turing Machine
- For brevity we will assume no evidence, i.e., the question we answer is: Given a network $\mathcal{B}$ with designated sets **H** and **H**, and $0 \leq q < 1$, is the probability $\Pr(\mathbf{H} = \mathbf{h}) > q$?
- We construct a PTM $\mathcal{M}$ such that, on such an input, it arrives in an accepting state with probability strictly larger than $1/2$ if and only if $\Pr(\mathbf{h}) > q$.
- $\mathcal{M}$ computes a joint probability $\Pr(y_1, \ldots, y_n)$ by iterating over i using a topological sort of the graph, and choosing a value for each variable Y_i conform the probability distribution in its CPT given the values that are already assigned to the parents of Y_i.

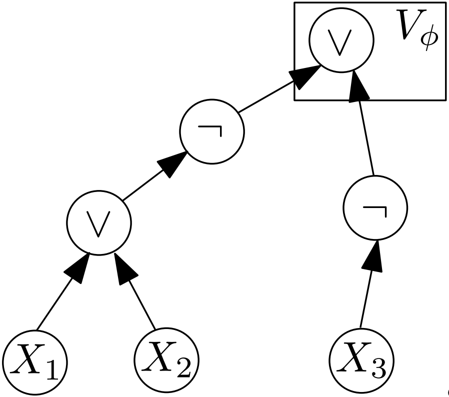
THRESHOLD INFERENCE is in PP

- Each computation path then corresponds to a specific joint value assignment to the variables in the network, and the probability of arriving in a particular state corresponds with the probability of that assignment.
- After iteration, we accept with probability $1/2 + (1 - q) \cdot \epsilon$, if the joint value assignment to $Y_1, \ldots, Y_n$ is consistent with **h**, and we accept with probability $1/2 - q \cdot \epsilon$ if the joint value assignment is *not* consistent with **h**.
- The probability of entering an accepting state is hence $\Pr(\mathbf{h}) \cdot (1/2 + (1 - q)\epsilon) + (1 - \Pr(\mathbf{h})) \cdot (1/2 - q \cdot \epsilon) = 1/2 + \Pr(\mathbf{h}) \cdot \epsilon - q \cdot \epsilon$.
- Indeed the probability of arriving in an accepting state is strictly larger than $1/2$ if and only if $\Pr(\mathbf{h}) > q$.

THRESHOLD INFERENCE is PP-hard

- We now show that THRESHOLD INFERENCE is PP-hard. We do so by reducing MAJSAT, which is known to be PP-complete, to THRESHOLD INFERENCE
- We construct a Bayesian network $\mathcal{B}_\phi$ from a given Boolean formula ϕ with n variables as follows:
  - For each propositional variable x_i in ϕ, a binary stochastic variable X_i is added to $\mathcal{B}_\phi$, with possible values TRUE and FALSE and a uniform probability distribution.
  - For each logical operator in ϕ, an additional binary variable in $\mathcal{B}_\phi$ is introduced, whose parents are the variables that correspond to the input of the operator, and whose CPT is equal to the truth table of that operator
  - The top-level operator in ϕ is denoted as V_ϕ.
- On the next slide, the network $\mathcal{B}_\phi$ is shown for the formula $\neg(x_1 \vee x_2) \vee \neg x_3$.

THRESHOLD INFERENCE is PP-hard



$\phi = \neg(x_1 \vee x_2) \vee \neg x_3$

THRESHOLD INFERENCE is PP-hard

- Now, for an arbitrary truth assignment **x** to the set of all propositional variables **X** in the formula ϕ we have that $\Pr(V_\phi = \text{TRUE} \mid \mathbf{X} = \mathbf{x})$ equals 1 if **x** satisfies ϕ, and 0 if **x** does not satisfy ϕ.
- Without any given joint value assignment, the prior probability $\Pr(V_\phi = \text{TRUE})$ is $\frac{\#_\phi}{2^n}$, where $\#_\phi$ is the number of satisfying truth assignments of the set of propositional variables **X**.
- Note that the above network $\mathcal{B}_\phi$ can be constructed from ϕ in polynomial time.
- We reduce MAJSAT to THRESHOLD INFERENCE. Let ϕ be a MAJSAT-instance and let $\mathcal{B}_\phi$ be the network as constructed above. Now, $\Pr(V_\phi = \text{TRUE}) > 1/2$ if and only if the majority of truth assignments satisfy ϕ.

THRESHOLD INFERENCE is PP-complete

- Given that THRESHOLD INFERENCE is PP-hard and in PP, it is PP-complete
- It is easy to show that NP $\subseteq$ PP and that THRESHOLD INFERENCE is NP-hard
- Why the additional work to prove exact complexity class?
  - PP is a class of a different nature than NP. This has effect on approximation strategies, fixed parameter tractability, etc.
  - Proving completeness for 'higher' complexity classes will typically also give intractability results for constrained problems – Cassio will talk about that

Approximation of MAP

- What does it mean for an algorithm to *approximate* MAP?
- Merriam-Webster dictionary: *approximate*: ‘to be very similar to but not exactly like (something)’
- In CS, this similarity is typically defined in terms of *value*:
  - ‘approximate solution A has a value that is close to the value of the optimal solution’
- However, other notions of approximation can be relevant
  - ‘approximate solution A' closely resembles the optimal solution’
  - ‘approximate solution A'' ranks within the top-m solutions’
  - ‘approximate solution A''' is quite likely to be the optimal solution’
- Note that these notions can refer to completely different solutions

Some formal notation

- For an arbitrary MAP instance $\{\mathcal{B}, \mathbf{H}, \mathbf{E}, \mathbf{I}, \mathbf{e}\}$, let $\mathit{cansol}_{\mathcal{B}}$ refer to the set of candidate solutions to $\{\mathcal{B}, \mathbf{H}, \mathbf{E}, \mathbf{I}, \mathbf{e}\}$, with $\mathit{optsol}_{\mathcal{B}} \in \mathit{cansol}_{\mathcal{B}}$ denoting the *optimal* solution (or, in case of a draw, one of the optimal solutions) to the MAP instance
- When $\mathit{cansol}_{\mathcal{B}}$ is ordered according to the probability of the candidate solutions (breaking ties between candidate solutions with the same probability arbitrarily), then $\mathit{optsol}_{\mathcal{B}}^{1\ldots m}$ refers to the set of the first m elements in $\mathit{cansol}_{\mathcal{B}}$, viz. the m most probable solutions to the MAP instance
- For a particular notion of approximation, we refer to an (unspecified) approximate solution as $\mathit{approxsol}_{\mathcal{B}} \in \mathit{cansol}_{\mathcal{B}}$

Approximation results

Definition (additive value-approximation of MAP)

Let $optsol_{\mathcal{B}}$ be the optimal solution to a MAP problem. An explanation $approxsol_{\mathcal{B}} \in cansol_{\mathcal{B}}$ is defined to ρ-additive value-approximate $optsol_{\mathcal{B}}$ if $\Pr(optsol_{\mathcal{B}}, \mathbf{e}) - \Pr(approxsol_{\mathcal{B}}, \mathbf{e}) \leq \rho$.

Result (Kwisthout, 2011)

It is NP*-hard to* ρ*-additive value-approximate MAP for* $\rho > \Pr(optsol_{\mathcal{B}}, \mathbf{e}) - \epsilon$ *for any constant* $\epsilon > 0$.

Approximation results

Definition (relative value-approximation of MAP)

Let $\mathit{optsol}_{\mathcal{B}}$ be the optimal solution to a MAP problem. An explanation $\mathit{approxsol}_{\mathcal{B}} \in \mathit{cansol}_{\mathcal{B}}$ is defined to ρ-relative value-approximate $\mathit{optsol}_{\mathcal{B}}$ if $\frac{\Pr(\mathit{optsol}_{\mathcal{B}} \mid \mathbf{e})}{\Pr(\mathit{approxsol}_{\mathcal{B}} \mid \mathbf{e})} \leq \rho$.

Result (Abdelbar & Hedetniemi, 1998)

It is NP*-hard to* ρ*-relative value-approximate MAP for* $\frac{\Pr(\mathit{optsol}_{\mathcal{B}} \mid \mathbf{e})}{\Pr(\mathit{approxsol}_{\mathcal{B}} \mid \mathbf{e})} \leq \rho$ *for any* $\rho > 1$*.*

Approximation results

Definition (structure-approximation of MAP)

Let $optsol_{\mathcal{B}}$ be the optimal solution to a MAP problem and let d_H be the Hamming distance. An explanation $approxsol_{\mathcal{B}} \in cansol_{\mathcal{B}}$ is defined to d-structure-approximate $optsol_{\mathcal{B}}$ if $d_H(approxsol_{\mathcal{B}}, optsol_{\mathcal{B}}) \leq d$.

Result (Kwisthout, 2013)

It is NP*-hard to* d*-structure-approximate MAP for any* $d \leq |optsol_{\mathcal{B}}| - 1$.

Approximation results

Definition (rank-approximation of MAP)

Let $optsol_{\mathcal{B}}^{1\ldots m} \subseteq cansol_{\mathcal{B}}$ be the set of the m most probable solutions to a MAP problem and let $optsol_{\mathcal{B}}$ be the optimal solution. An explanation $approxsol_{\mathcal{B}} \in cansol_{\mathcal{B}}$ is defined to m-rank-approximate $optsol_{\mathcal{B}}$ if $approxsol_{\mathcal{B}} \in optsol_{\mathcal{B}}^{1\ldots m}$.

Result (Kwisthout, 2015)

It is NP*-hard to m-rank-approximate MAP for any constant m.*

Approximation results

Definition (expectation-approximation of MAP)

Let *optsol*$_{\mathcal{B}}$ be the optimal solution to a MAP problem and let $\mathbb{E}$ be the the expectation function. An explanation *approxsol*$_{\mathcal{B}} \in$ *cansol*$_{\mathcal{B}}$ is defined to ϵ-expectation-approximate *optsol*$_{\mathcal{B}}$ if $\mathbb{E}(\Pr(\mathit{optsol}_{\mathcal{B}}) \neq \Pr(\mathit{approxsol}_{\mathcal{B}})) < \epsilon$.

Result (Folklore)

There cannot exist a randomized algorithm that ϵ-expectation-approximates MAP in polynomial time for $\epsilon < 1/2 - 1/n^c$ for a constant c unless NP $\subseteq$ BPP.

Summary

Approximation	constraints	assumption
value, additive	$c = 2, d = 2, \|\mathbf{E}\| = 1, \mathbf{I} = \varnothing$	P $\neq$ NP
value, ratio	$c = 2, d = 3, \mathbf{E} = \varnothing$	P $\neq$ NP
structure	$c = 3, d = 3, \mathbf{I} = \varnothing$	P $\neq$ NP
rank	$c = 2, d = 2, \|\mathbf{E}\| = 1, \mathbf{I} = \varnothing$	P $\neq$ NP
expectation	$c = 2, d = 2, \|\mathbf{E}\| = 1, \mathbf{I} = \varnothing$	NP $\not\subseteq$ BPP

Table: Summary of intractability results for MAP approximations