Introduction to Graphical Models

lecture 3 - Elimination Algorithm & Belief Propagation

Marc Toussaint

TU Berlin

– factor graphs

– Elilmination Algorithm

– Belief Propagation & special cases

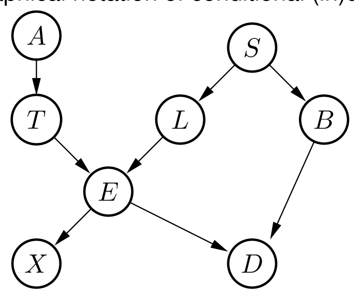
last time's summary

- today:
  – what is a Bayes Net
  – what is inference good for
  – usage of inference software

- next time:
  – if you had to program such an inference software
  – algorithms for inference
  – factor graphs, elimination, Belief Propagation

recap: Bayesian Networks

- Bayesian Network: graphical notation of conditional (in)dependencies



$\iff P(D, X, E, B, L, T, S, A) =$
$P(D|E, B)\ P(X|E)\ P(E|T, L)\ P(B|S)\ P(L|S)\ P(T|A)\ P(S)\ P(A)$

- A Bayesian network is a DAG that defines for each node X_i what the parents $\pi(i)$ such that

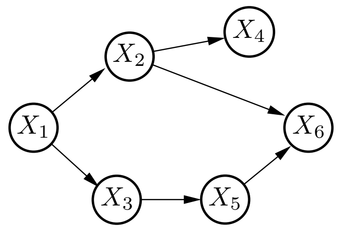
$$P(X_{1:n}) = \prod_{i=1}^{n} P(X_i \,|\, X_{\pi(i)})$$

(notation: $X_{\pi(i)} = (X_a, .., X_b)$ if $\pi(i) = (a, .., b)$)

Bayes Net $\rightarrow$ facor graph

- for the computations, what matters are the *factors* the joint is build of:

Example:



$\iff P(x_{1:6}) =$
$P(x_1)\ P(x_2|x_1)\ P(x_3|x_1)\ P(x_4|x_2)\ P(x_5|x_3)\ P(x_6|x_2, x_5)$

problem: compute $P(x_1, x_6)$

Bayes Net → facor graph

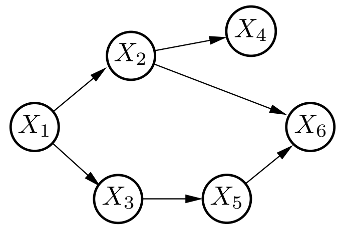
... continued...

$$
\begin{aligned}
&P(x_1, x_6) \\
&= \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} P(x_1)\ P(x_2|x_1)\ P(x_3|x_1)\ P(x_4|x_2)\ P(x_5|x_3)\ P(x_6|x_2, x_5) \\
&= P(x_1) \sum_{x_2} P(x_2|x_1) \sum_{x_3} P(x_3|x_1) \sum_{x_4} P(x_4|x_2) \sum_{x_5} P(x_5|x_3)\ P(x_6|x_2, x_5) \\
&= P(x_1) \sum_{x_2} P(x_2|x_1) \sum_{x_3} P(x_3|x_1) \sum_{x_4} P(x_4|x_2)\ t_1(x_2, x_3, x_6) \\
&= P(x_1) \sum_{x_2} P(x_2|x_1) \sum_{x_3} P(x_3|x_1)\ t_1(x_2, x_3, x_6) \sum_{x_4} P(x_4|x_2) \\
&= P(x_1) \sum_{x_2} P(x_2|x_1)\ t_2(x_2) \sum_{x_3} P(x_3|x_1)\ t_1(x_2, x_3, x_6) \\
&= P(x_1) \sum_{x_2} P(x_2|x_1)\ t_2(x_2)\ t_3(x_1, x_2, x_6) \\
&= P(x_1)\ t_4(x_1, x_6)
\end{aligned}
$$

→ what matters is: on which variables depends each term

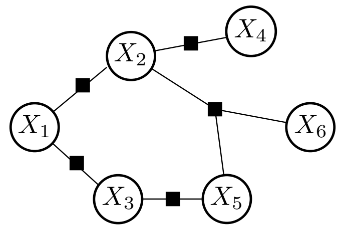
Bayes Net → facor graph

- Bayesian Network:



$\iff$ $P(x_{1:6}) = P(x_1)\ P(x_2|x_1)\ P(x_3|x_1)\ P(x_4|x_2)\ P(x_5|x_3)\ P(x_6|x_2, x_5)$

- Factor Graph:



$\iff$ $P(x_{1:6}) = \psi_1(x_1, x_2)\ \psi_2(x_3, x_1)\ \psi_3(x_2, x_4)\ \psi_4(x_3, x_5)\ \psi_5 x_2, x_5, x_6)$

factor graphs

- mathematically: a factor graph is given by a
  – a set of random variables variables $X_1, .., X_n$
  – a set of cliques $C_1, .., C_k$ (which are tuples of variables)
  – for each clique a factor $\psi_i(X_{C_i})$ s.t.:

$$P(X_1, .., X_n) = \prod_{i=1}^{k} \psi_i(X_{C_i})$$

(notation: $X_C = (X_a, .., X_b)$ if $C = (a, .., b)$)

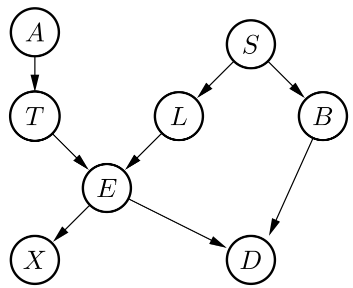
- graphically: a factor graph is a bi-partite graph with
  – factors (black boxes) connecting to
  – variables (circles)

- a factor graph is more general than a Bayes Net:
  – describes general couplings between variables in terms of common factors
  – not only contidional probabilities

- easy to represent in a computer

factor graphs

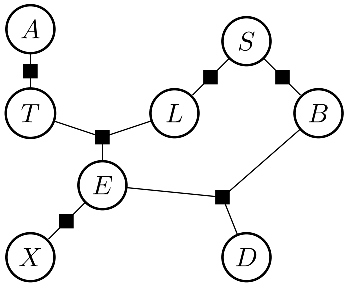
- asia example:
  Bayes Net:



$\iff$ $P(D, X, E, B, L, T, S, A) =$
$P(D|E, B)\ P(X|E)\ P(E|T, L)\ P(B|S)\ P(L|S)\ P(T|A)\ P(S)\ P(A)$

factor graphs

- asia example:
  factor graph:



$\iff P(D, X, E, B, L, T, S, A) =$
$\psi_1(D, E, B)\ \psi_2(X, E)\ \psi_3(E, T, L)\ \psi_4(B, S)\ \psi_5(L, S)\ \psi_6(T, A)$

- Bayes Net $\to$ factor graph   corresponds to *moralization*...

digression: additive decomposable functions

- graphical models describe how a joint factors
  – factorization corresponds to independence (by def)
  – factors correspond to "directly coupled/interacting" variables

- take the neg-log of the joint:

$$E(X_{1:n}) := -\log P(X_{1:n}) = \sum_{i=1}^{k} \phi_i(X_{C_i}) \quad \text{with} \quad \phi_i = -\log \psi_i$$

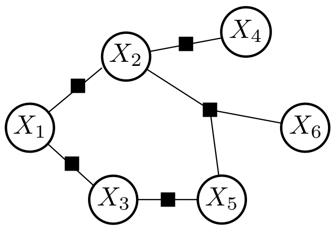
  assigns an *error* or *energy* to every possible configuration $x_{1:n}$
  [Physics: at temperature T an ensemble of particles is distributed as $P(X_{1:n}) \propto \exp(-E(X_{1:n})/T)$]

- $E(X_{1:n})$ is an additive decomposable function!
  – optimization: find $\mathrm{argmin}_{X_{1:n}} E(X_{1:n})$
  – additive decomposition makes optimization easier
  – expresses independence in the sense of optimization
  – optimization of E closely related to inference in P

Elimination Algorithm

... same example as above – in terms of a factor graph

- Factor Graph:



$\iff P(x_{1:6}) = \psi(x_1, x_2)\ \psi(x_3, x_1)\ \psi(x_2, x_4)\ \psi(x_3, x_5)\ \psi(x_2, x_5, x_6)$

problem: compute $P(x_1, x_6)$

Elimination Algorithm

$$
\begin{aligned}
&P(x_1, x_6) \\
&= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} \psi(x_1, x_2)\ \psi(x_3, x_1)\ \psi(x_2, x_4)\ \psi(x_3, x_5)\ \psi(x_2, x_5, x_6) \\
&= \sum_{x_2}\sum_{x_3}\sum_{x_4} \psi(x_1, x_2)\ \psi(x_3, x_1)\ \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5)\ \psi(x_2, x_5, x_6) \\
&= \sum_{x_2}\sum_{x_3}\sum_{x_4} \psi(x_1, x_2)\ \psi(x_3, x_1)\ \psi(x_2, x_4)\ t_1(x_2, x_3, x_6) \\
&= \sum_{x_2}\sum_{x_3} \psi(x_1, x_2)\ \psi(x_3, x_1)\ t_1(x_2, x_3, x_6) \sum_{x_4} \psi(x_2, x_4) \\
&= \sum_{x_2}\sum_{x_3} \psi(x_1, x_2)\ \psi(x_3, x_1)\ t_1(x_2, x_3, x_6)\ t_2(x_2) \\
&= \sum_{x_2} \psi(x_1, x_2)\ t_2(x_2) \sum_{x_3} \psi(x_3, x_1)\ t_1(x_2, x_3, x_6) \\
&= \sum_{x_2} \psi(x_1, x_2)\ t_2(x_2)\ t_3(x_1, x_2, x_6) \\
&= t_4(x_1, x_6)
\end{aligned}
$$

- we can automate this!

Elimination Algorithm

- `eliminate_single_variable`(F, i)

  1: **Input:** list F of factors, variable id i
  2: **Output:** list F of factors
  3: find relevant subset $f \subset F$ of factors over i: $f = \{C : i \in C\}$
  4: define remaining clique C_t = all variables in f except i $C_t = \text{vars}(f) \setminus \{i\}$
  5: compute temporary factor $t(X_{C_t}) = \sum_{X_i} \prod_{\psi \in f} \psi$
  6: remove old factors f and append new temporary factor t to F
  7: return F

- `elimination_algorithm`(m, F, C_o)

  1: **Input:** list F of factors, tuple C_o of output variables ids
  2: **Output:** single factor m over variables X_{C_o}
  3: define all variables present in F: $V = \text{vars}(F)$
  4: define variables to be eliminated: $E = V \setminus C_o$
  5: for all $i \in E$: `eliminate_single_variable`(F, i)
  6: for all remaining factors, compute the product $m = \prod_{\psi \in F} \psi$
  7: return m

Elimination Algorithm

- pros:
  – very simple, trivial to prove correct
  (does exactly what we'd do on paper)

- cons:
  – computes only one marginal $P(X_i)$
  – need to call it n-times to compute all marginals $P(X_1)$,.., $P(X_n)$

Belief propagation

- ... do somehow the same as elimination, but:
  – more locally
  – with other kinds of temporary factors, reusable for *all* marginals

- belief propagation:
  – compute *messages*

$$\mu_{C \to i}(X_i) = \sum_{X_C \setminus X_i} \psi_C(X_C) \prod_{j \in C, j \neq i} \mu_{j \to C}(X_j) \,,$$

$$\mu_{i \to C}(X_i) = \prod_{D \in \nu(i), D \neq C} \mu_{D \to i}(X_i)$$

  – from the messages, compute *beliefs*

$$b_C(X_C) := \psi_C(X_C) \prod_{i \in C} \mu_{i \to C}(X_i) \,, \quad b_i(X_i) := \prod_{C \in \nu(i)} \mu_{C \to i}(X_i)$$

  – factor-to-variable messages $\mu_{C \to i}$
  – variable-to-factor messages $\mu_{i \to C}$

understanding BP

1) when can we resolve the recursive equations?

2) compare to Elimination Algorithm on a tree

3) trees, independent sources of information, & Naive Bayes!

4) local consistency as fixed point of message updates

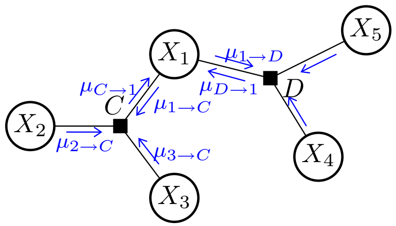
5) the problem with loops, Bethe approximation

BP – resolving the recursion

- BP equations:

$$\mu_{C\to i}(X_i) = \sum_{X_C \setminus X_i} \psi_C(X_C) \prod_{j\in C, j\neq i} \mu_{j\to C}(X_j)\ ,$$

$$\mu_{i\to C}(X_i) = \prod_{D\in\nu(i), D\neq C} \mu_{D\to i}(X_i)$$



- the recursive dependencies in the BP equations can be resolved *iff the graph is a tree!*

BP – relation to Elimination Algorithm

- consider the factor graph (tree!) $P_{x_{1:6}} = \psi(x_1,x_2,x_3)\ \psi(x_1,x_4,x_5)$
- Elimination Algorithm: (pick elimination order from leaves to root)

$$P(x_1) = \Big[\sum_{x_2,x_3} \psi(x_1,x_2,x_3)\Big]\ \Big[\sum_{x_4,x_5} \psi(x_1,x_4,x_5)\Big]$$

$$P(x_1) = t_1(x_1)\ t_2(x_1)$$

- Belief Propagation:

$$b(x_1) = \mu_{C\to 1}(x_1)\ \mu_{D\to 1}(x_1)$$

$$\mu_{C\to 1} = \sum_{x_2,x_3} \psi(x_1,x_2,x_3)$$

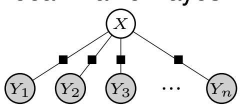
$$\mu_{D\to 1} = \sum_{x_4,x_5} \psi(x_1,x_4,x_5)$$

- messages correspond to temporal factors in Elimination Alg!
  $\Rightarrow$ the computed belief is equal to the marginal from the Elimination Algorithm
  $\Rightarrow$ when we compute *all* messages on a tree, we can return *all* marginals!
- *proof of correctness of BP on trees*

BP on trees & Naive Bayes

- what's so special about trees?
  – we can resolve recursive BP equations, and:
  – *the branches of each node in a tree contain independent information*

- recall Naive Bayes:



$\iff\ P(X, Y_{1:n}) = P(X)\ \prod_{i=1}^n P(Y_i|X)$

  – one hidden variable, many *conditionally independent* evidences
  – posterior: $P(x|y_{1:n}) \propto P(x)\ \prod_{i=1}^n \mu_i(x)$ with $\mu_i(x) := P(Y_i{=}y_i\,|\,x)$
  – multiplying distributions $\leftrightarrow$ fusing (independent!) information!

$\Rightarrow$ *every node in a tree is like Naive Bayes, with each branch contributing independent information!*
  – the (posterior) belief at a node is the product of all incoming messages! $\ (b_i(X_i) := \prod_{C\in\nu(i)} \mu_{C\to i}(X_i))$

BP update equations

- what if the model is not a tree? cannot resolve the recursions...!?

- use BP equations as update equations:
  – initialize all messages as one: $\mu_{C\to i} = 1,\ \mu_{i\to C} = 1$
  – update messages

$$\mu^{\text{new}}_{C\to i}(X_i) = \sum_{X_C\setminus X_i} \psi_C(X_C) \prod_{j\in C, j\neq i} \mu^{\text{old}}_{j\to C}(X_j)\ ,$$

$$\mu^{\text{new}}_{i\to C}(X_i) = \prod_{D\in\nu(i), D\neq C} \mu^{\text{old}}_{D\to i}(X_i)$$

  – compute current *beliefs*

$$b_C(X_C) := \psi_C(X_C) \prod_{i\in C} \mu_{i\to C}(X_i)\ ,\quad b_i(X_i) := \prod_{C\in\nu(i)} \mu_{C\to i}(X_i)$$

- alternative equations:

$$\mu^{\text{new}}_{C\to i}(X_i) = \frac{1}{\mu^{\text{old}}_{i\to C}(X_i)} \sum_{X_C\setminus X_i} b^{\text{old}}_C(X_C)$$

$$\mu^{\text{new}}_{i\to C}(X_i) = \frac{1}{\mu^{\text{old}}_{C\to i}(X_i)}\ b^{\text{old}}_i(X_i)$$

BP & marginal consistency as fixed point

- definition of marginal consistency:
  when two cliques C and D and share a variable X_i, then their marginal beliefs should coincide,

$$\sum_{X_C \setminus X_i} b(X_C) = \sum_{X_D \setminus X_i} b(X_D) = b(X_i) \tag{1}$$

  Note, consistency also implies

$$b(X_i) = \mu_{C \to i}(X_i)\ \mu_{i \to C}(X_i)$$

- *marginal consistency is a fixed point of the BP updates!*
  (if (1) holds, the BP update do not change the messages)
  – trivialy to see with the alternative update equations

$$\mu^{\text{new}}_{C \to i}(X_i) = \frac{1}{\mu^{\text{old}}_{i \to C}(X_i)} \sum_{X_C \setminus X_i} b^{\text{old}}_C(X_C)$$

$$\mu^{\text{new}}_{i \to C}(X_i) = \frac{1}{\mu^{\text{old}}_{C \to i}(X_i)}\ b^{\text{old}}_i(X_i)$$

BP – summary so far

- 1) BP (with recursive computation of messages) leads to exact inference on trees ($\leftrightarrow$ elimination algorithm)
- 2) marginal consistency is a fixed point of the update equations
  – this statement is true also non-trees! (loopy graphs)
  – on trees, the parallel update of messages will converge to the true messages
  – on non-trees, when it converges, then to a state of marginal consistency
- apply BP on loopy graphs?

BP & the problem with loops

[no fully rigorous treatement in this lecture]

- problem on an intuitive level:
  – loops $\Rightarrow$ this is no Naive Bayes anymore!
  – branches of a node to not represent independent information anymore!
  – BP is multiplying (=fusion) messages from dependent sources of information
  – echo effects

  $\Rightarrow$ can diverge
  $\Rightarrow$ typically converges, but to a perturbed results
  (e.g., positiv feedback $\rightarrow$ over confident posteriors)

BP & the problem with loops

- there exists a theory on what loopy BP converges to
  Bethe approximation, (Yedidia, Freeman, & Weiss, 2001)
- we shouldn't be overly disappointed:
  – if BP was exact on loopy graphs we could efficiently solve NP hard problems...
  – loopy BP is a *very* interesting approximation to solving an NP hard problem
  – is hence also applied in context of combinatorial optimization (e.g., SAT problems)
- ways to reduce (not fully resolve!) the problems with loops:
  – Generalized BP
  – loop corrections
  – ongoing research

BP – wrapup

- BP very powerful inference method
  – local computations, local integration of messages (Naive Bayes)
  – very concrete idea/model of information processing on networks
  – exact on trees

- different versions
  – recursive computation of exact messages (possible only on trees) → exact inference
  – initialize all messages as 1, then update them iteratively
  – parallel update (recompute factor-to-variable, then variable-to-factor messages)
  – sequential update (recompute messages in some order)

- further reading
  – lecture notes
  `http://user.cs.tu-berlin.de/~mtoussai/notes/index.html`
  – the references therein!

BP & important special cases

- general BP equations:

$$\mu_{C\to i}(X_i) = \sum_{X_C \setminus X_i} \psi_C(X_C) \prod_{j \in C, j \neq i} \mu_{j\to C}(X_j) \ ,$$
$$\mu_{i\to C}(X_i) = \prod_{D \in \nu(i), D \neq C} \mu_{D\to i}(X_i)$$

- *special case pair-wise factors*: each clique C is a pair $C = (X_i, X_j)$
  we can define *variable-to-variable* messages $\mu_{j\to i}(X_i) := \mu_{C\to i}(X_i)$
  where $C = (X_i, X_j)$ is unique

$$\mu_{j\to i}(X_i) = \sum_{X_j} \psi_C(X_i, X_j) \prod_{k: k \neq i} \mu_{k\to j}(X_j) \ ,$$

- is an important special case
  – Hidden Markov Model
  – Boltzmann machine (model of a neural network)

BP & important special cases

- general BP equations:

$$\mu_{C\to i}(X_i) = \sum_{X_C \setminus X_i} \psi_C(X_C) \prod_{j \in C, j \neq i} \mu_{j\to C}(X_j) \ ,$$
$$\mu_{i\to C}(X_i) = \prod_{D \in \nu(i), D \neq C} \mu_{D\to i}(X_i)$$

- *special case* each variable X_i is contained in only two cliques
  we can define *clique-to-clique* messages $\mu_{D\to C}(X_i) := \mu_{i\to C}(X_i)$
  where $i = C \cap D$ is unique

$$\mu_{D\to C}(X_i) = \sum_{X_D \setminus X_i} \psi_D(X_D) \prod_{E: E \neq C} \mu_{E\to D}(X_{E\cap D}) \ ,$$

- also relevant:
  – these are the inference equations on a "Junction Tree"
  (role of variables is replaced by separators)

Summary

- factor graphs:
  – simply represent the factors in the joint, and which variables they depend on

- elimination algorithm:
  – summing over a variable produces a new temporary factor
  – iteratively: summation, augment the list with the new factor, take the old factors out of the list

- Belief Propagation (aka Message Passing)
  – generic inference method
  – exact on trees (equivalent to elimination algorithm)
  – approximate on loops
  – special case for pair-wise coupling (e.g., HMMs, many more)