EE512A – Advanced Inference in Graphical Models

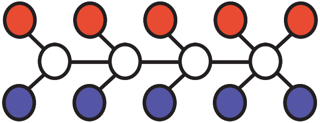
— Fall Quarter, Lecture 10 —

http://j.ee.washington.edu/~bilmes/classes/ee512a_fall_2014/

Prof. Jeff Bilmes

University of Washington, Seattle
Department of Electrical Engineering
http://melodi.ee.washington.edu/~bilmes

Nov 3rd, 2014



Announcements

- Wainwright and Jordan *Graphical Models, Exponential Families, and Variational Inference* `http://www.nowpublishers.com/product.aspx?product=MAL&doi=2200000001`
- Read chapters 1,2, and 3 in this book!

Class Road Map - EE512a

- L1 (9/29): Introduction, Families, Semantics
- L2 (10/1): MRFs, elimination, Inference on Trees
- L3 (10/6): Tree inference, message passing, more general queries, non-tree)
- L4 (10/8): Non-trees, perfect elimination, triangulated graphs
- L5 (10/13): triangulated graphs, k-trees, the triangulation process/heuristics
- L6 (10/15): multiple queries, decomposable models, junction trees
- L7 (10/20): junction trees, begin intersection graphs
- L8 (10/22): intersection graphs, inference on junction trees
- L9 (10/27): inference on junction trees, semirings,
- L10 (11/3): conditioning, hardness, LBP
- L11 (11/5): LBP, exponential models, mean params and polytopes
- L13 (11/10):
- L14 (11/12):
- L15 (11/17):
- L16 (11/19):
- L17 (11/24):
- L18 (11/26):
- L19 (12/1):
- L20 (12/3):
- Final Presentations: (12/10):

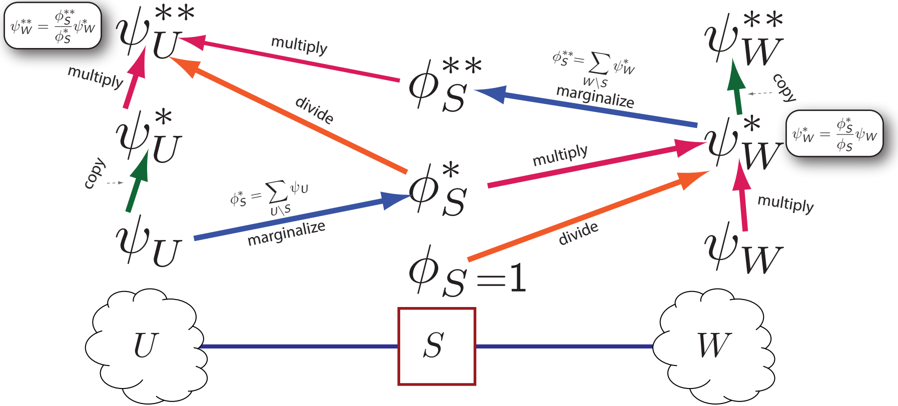
Finals Week: Dec 8th-12th, 2014.

Recap

- Message passing on junction tree nodes, definition of messages, divide out old, multiply in new.
- Messages in both directions.
- For general tree, we have MPP like in 1-tree case.
- Suff condition: locally consistent.
- Thm: MPP renders cliques locally consistent between pairs.
- In JT (r.i.p.) locally consistent ensures globally consistent.
- In JT (r.i.p.), running MPP gives marginals.
- Commutative semiring - other algebraic objects can be used.
- Time and memory complexity is $O(Nr^{\omega+1})$ where ω is the tree-width.

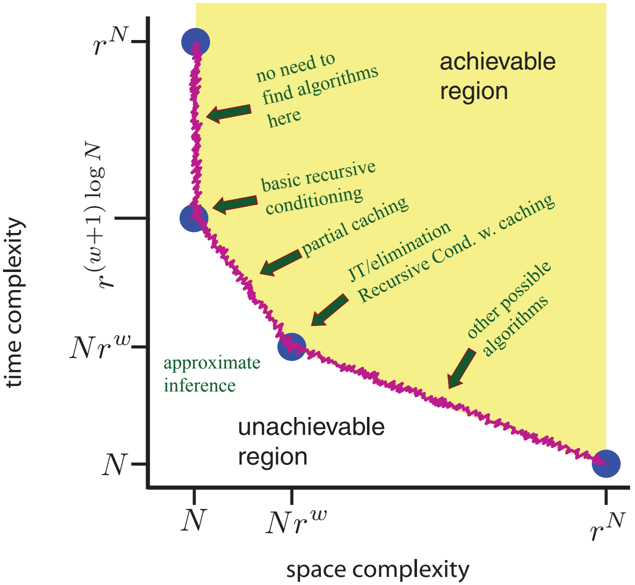
Forward/Backward Messages Along Cluster Tree Edge

Summarizing, forward and backwards messages proceed as follows:



Recall: $S = U \cap W$, and we initialize ψ_U and ψ_W with factors that are contained in U or W.

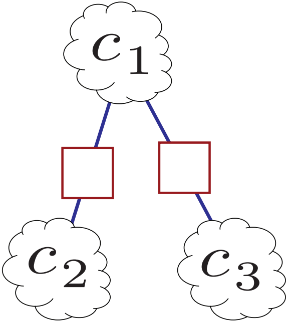
Time-Space Tradeoffs



Recursive Conditioning, three cluster version

Example: 3-cluster version

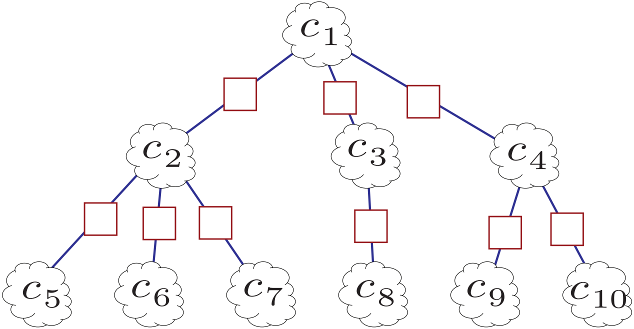
$\alpha_1 \leftarrow 0$;
for $x_{C_1} \in \mathsf{D}_{X_{C_1}}$ **do**
  $\alpha_{2|1} \leftarrow 0$; $\alpha_{3|1} \leftarrow 0$;
  for $x_{C_2 \setminus C_1} \in \mathsf{D}_{X_{C_2 \setminus C_1}}$ **do**
    $\alpha_{2|1} \mathrel{+}= p(x_{C_1 \cup C_2})$
  for $x_{C_3 \setminus C_1} \in \mathsf{D}_{X_{C_3 \setminus C_1}}$ **do**
    $\alpha_{3|1} \mathrel{+}= p(x_{C_1 \cup C_3})$
  $\alpha_1 \mathrel{+}= \alpha_{2|1}\alpha_{3|1}$



- Outer loop costs $O(|\mathsf{D}_{X_{C_1}}|)$. Inner loops each cost $O(|\mathsf{D}_{X_{C_2 \setminus C_1}}|)$ (assuming C_1 and C_2 are same size).
- Total cost is $O(|\mathsf{D}_{X_{C_1 \cup C_2}}|)$, better than $O(|\mathsf{D}_{X_{C_1 \cup C_2 \cup C_3}}|) = O(r^N)$
- Memory: still linear.

Recursive Conditioning with good order

- We can order the cliques in a different way though. Note that this is not necessarily a junction tree, although it might be. Rather, this is more akin to a decomposition trees we saw earlier in the course, but it is not that either. Instead, it is more of a "conditioning tree"
- Depth of tree is $d = O(\log N)$



Recursive Conditioning with good order

All α's initialized to 0 before 'for' loop where they are accumulated.

1 **for** $x_{C_1} \in \mathsf{D}_{X_{C_1}}$ **do**
2   **for** $x_{C_2 \setminus C_1} \in \mathsf{D}_{X_{C_2 \setminus C_1}}$ **do**
3   **for** $x_{C_5 \setminus C_{1,2}} \in \mathsf{D}_{X_{C_5 \setminus C_{1,2}}}$ **do**
4             $\alpha_{5|1,2} \mathrel{+}= p(x_{C_{1,2,5}})$
5   **for** $x_{C_6 \setminus C_{1,2}} \in \mathsf{D}_{X_{C_6 \setminus C_{1,2}}}$ **do**
6             $\alpha_{6|1,2} \mathrel{+}= p(x_{C_{1,2,6}})$
7   **for** $x_{C_7 \setminus C_{1,2}} \in \mathsf{D}_{X_{C_7 \setminus C_{1,2}}}$ **do**
8             $\alpha_{7|1,2} \mathrel{+}= p(x_{C_{1,2,7}})$
9         $\alpha_{2|1} \mathrel{+}= \alpha_{5|1,2}\alpha_{6|1,2}\alpha_{7|1,2}$
10   *Include lines 12-22 here*

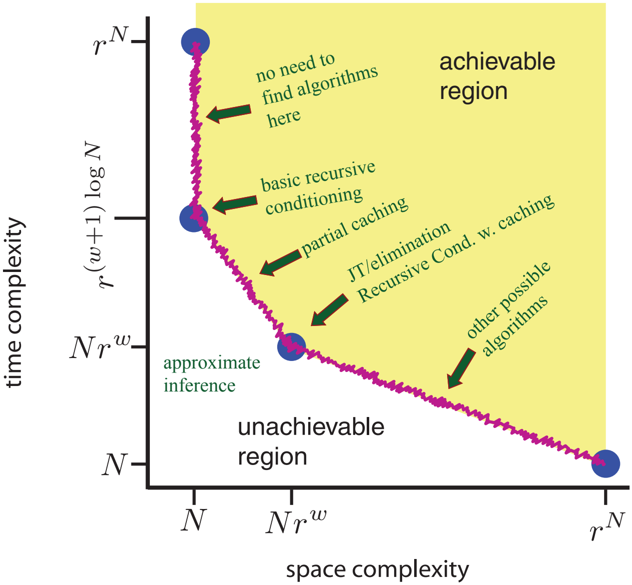
11 *Lines 12-22, include at line 10 above*
12 **for** $x_{C_3 \setminus C_1} \in \mathsf{D}_{X_{C_3 \setminus C_1}}$ **do**
13   **for** $x_{C_8 \setminus C_{1,3}} \in \mathsf{D}_{X_{C_8 \setminus C_{1,3}}}$ **do**
14         $\alpha_{8|1,3} \mathrel{+}= p(x_{C_{1,3,8}})$
15     $\alpha_{3|1} \mathrel{+}= \alpha_{8|1,3}$
16 **for** $x_{C_4 \setminus C_1} \in \mathsf{D}_{X_{C_4 \setminus C_1}}$ **do**
17   **for** $x_{C_9 \setminus C_{1,4}} \in \mathsf{D}_{X_{C_9 \setminus C_{1,4}}}$ **do**
18         $\alpha_{9|1,4} \mathrel{+}= p(x_{C_{1,4,9}})$
19   **for** $x_{C_{10} \setminus C_{1,4}} \in \mathsf{D}_{X_{C_{10} \setminus C_{1,4}}}$ **do**
20         $\alpha_{10|1,4} \mathrel{+}= p(x_{C_{1,4,10}})$
21     $\alpha_{4|1} \mathrel{+}= \alpha_{9|1,4}\alpha_{10|1,4}$
22 $\alpha_1 \mathrel{+}= \alpha_{2|1}\alpha_{3|1}\alpha_{4|1}$

Recursive Conditioning with good order

- When we're all done, $\alpha_1 = p(\bar{x}_E)$ (again, assuming evidence is treated as multiplies by $\delta(x, \bar{x})$).
- How much space is needed? $O(N)$ still since in worst case, depth of the tree is number of maxcliques (which is $O(N)$).
- How much time? Depends on number of α-accumulates, or number of leaf-nodes in the tree. Depth is $d = \log N$. Each clique gets run about r^{w+1} times, and runs the nodes below it about that many times.
- We get a time complexity of:

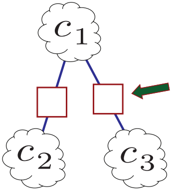
$$\underbrace{r^{w+1} r^{w+1} \dots r^{w+1}}_{d \text{ times}} = r^{(w+1)\log N} \tag{10.21}$$

Time-Space Tradeoffs



Recursive Conditioning with good order

- How to get other points on frontier?
- Note that in previous algorithm, for each set of variable values in intersection set (square boxes), we were solving the same sub-problem multiple times.
- We can cache the solutions for each value, at the cost of more memory. If everything is cached, space complexity will increase to $O(Nr^w)$ and time complexity will decrease to $O(Nr^w)$ (like the JT case).



we need not solve each entry in this intersection set multiple times. Instead, we can cache values. Total number of entrie is $O(Nr^W)$

Value-specific Caching

- Many algorithms use value specific caching. I.e., depending on the values of some variables currently conditioned on, we might actually get an entirely different set of maxcliques (or set of sets of maxcliques) below. Each should ideally be treated differently.
- We can construct and memoize the *dependency sets*, the set of variables and their values that induce particular sub-computations. Each sub-computation might be a computation of a sum, or it might even be a computation of zero (called a no-good, or a conflict). Each of these can be memoized and re-used whenever the dependency set becomes active again.
- the order of the cliques and the order of the variables in the cliques might dynamically change depending on previously instantiated values. We might not even use cliques at all, and do this at the granularity of variables and their values.

Value-Elimination

- This is the basis of the value elimination algorithm (Bacchus-2003), a general procedure for probabilistic inference. It gets much of its inspiration from the techniques used to produce fast SAT and constraint satisfaction problem (CSP) engines.
- This is especially useful if we have many zeros (sparsity) in the distribution and/or if there is much value specific independence.

Hardness

- Even with conditioning, search, etc. Complexity of exact inference is always exponential in at least the tree-width of any covering graph if we do it as we've been describing.

Hardness

- Even with conditioning, search, etc. Complexity of exact inference is always exponential in at least the tree-width of any covering graph if we do it as we've been describing.
- Unfortunately, finding the best exponent (i.e., finding the best covering triangulated graph (with minimal tree-width)) is, as we saw in earlier lectures, an NP-complete optimization problem.

Hardness

- Even with conditioning, search, etc. Complexity of exact inference is always exponential in at least the tree-width of any covering graph if we do it as we've been describing.
- Unfortunately, finding the best exponent (i.e., finding the best covering triangulated graph (with minimal tree-width)) is, as we saw in earlier lectures, an NP-complete optimization problem.
- Even worse, inference itself is NP-complete. There are some graphs that can't be solved in polynomial time unless P=NP (so it seems exponential cost is probably inevitable).

Hardness of Inference

- Consider the 3-SAT problem (which is a canonical NP-complete problem). Given list of N variables, and a collection of M clauses (constraints), where each clause is a disjunction ("or") of 3 literals (a variable or its negation). Clauses are organized in a conjunction ("and").

Hardness of Inference

- Consider the 3-SAT problem (which is a canonical NP-complete problem). Given list of N variables, and a collection of M clauses (constraints), where each clause is a disjunction ("or") of 3 literals (a variable or its negation). Clauses are organized in a conjunction ("and").
- *Question:* is there a satisfying truth assignment of the variables (assignment of variable values that makes the conjunction of disjunctions true).

Hardness of Inference

- Consider the 3-SAT problem (which is a canonical NP-complete problem). Given list of N variables, and a collection of M clauses (constraints), where each clause is a disjunction ("or") of 3 literals (a variable or its negation). Clauses are organized in a conjunction ("and").
- *Question:* is there a satisfying truth assignment of the variables (assignment of variable values that makes the conjunction of disjunctions true).
- Two examples:

$$(x_1 \vee x_4 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_4 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \tag{10.1}$$

and also

$$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_7 \vee x_8 \vee x_9) \wedge (\bar{x}_9 \vee x_{10} \vee x_{11}) \wedge (\bar{x}_{11} \vee \bar{x}_{12} \vee \bar{x}_3) \tag{10.2}$$

Hardness of Inference

- In the general case, we have N variables and M clauses, either of which might be very large. If we can solve this problem in polynomial time in N, then all NP-complete problems can be solved in polynomial time.

Hardness of Inference

- In the general case, we have N variables and M clauses, either of which might be very large. If we can solve this problem in polynomial time in N, then all NP-complete problems can be solved in polynomial time.
- To show that inference in Bayesian networks is NP-complete, all we need to do is find a BN or MRF that encodes this problem using the appropriate commutative semiring (which in our case, we'll take to be the max-product semiring).

Hardness of Inference

- In the general case, we have N variables and M clauses, either of which might be very large. If we can solve this problem in polynomial time in N, then all NP-complete problems can be solved in polynomial time.
- To show that inference in Bayesian networks is NP-complete, all we need to do is find a BN or MRF that encodes this problem using the appropriate commutative semiring (which in our case, we'll take to be the max-product semiring).
- Let $\{x_i\}_{i=1}^N$ be the set of variables, and let C_j be the index set of the variables for clause $0 \leq j \leq M$.

Hardness of Inference

- In the general case, we have N variables and M clauses, either of which might be very large. If we can solve this problem in polynomial time in N, then all NP-complete problems can be solved in polynomial time.
- To show that inference in Bayesian networks is NP-complete, all we need to do is find a BN or MRF that encodes this problem using the appropriate commutative semiring (which in our case, we'll take to be the max-product semiring).
- Let $\{x_i\}_{i=1}^N$ be the set of variables, and let C_j be the index set of the variables for clause $0 \leq j \leq M$.
- Define binary-valued functions $f_j(x_{C_j})$ such that $f_j = 1$ iff the clause is satisfied by the current values of the variables x_{C_j}, otherwise $f_j = 0$.

Hardness of Inference

- With this formulation, we get factorization as follows

$$\prod_j f_j(x_{C_j}) \tag{10.3}$$

which is possible to evaluate to unity iff the logic formula is satisfiable.

Hardness of Inference

- With this formulation, we get factorization as follows

$$\prod_j f_j(x_{C_j}) \tag{10.3}$$

which is possible to evaluate to unity iff the logic formula is satisfiable.
- If we take $p(x) \propto$ the above, we've got an MRF for SAT and we're done.

Hardness of Inference

- With this formulation, we get factorization as follows

$$\prod_j f_j(x_{C_j}) \tag{10.3}$$

  which is possible to evaluate to unity iff the logic formula is satisfiable.
- If we take $p(x) \propto$ the above, we've got an MRF for SAT and we're done.
- Next, consider BN with N binary variables $\{x_i\}_{i=1}^N$ and M additional variables $\{y_j\}_{j=1}^M$ with M CPTS of the form:

$$p(y_j = 1 | x_{C_j}) = \begin{cases} 1 & \text{if } f_j(x_{C_j}) = 1 \\ 0 & \text{else} \end{cases} \text{, and for } x_i \; p(x_i = 1) = 0.5 \tag{10.4}$$

Hardness of Inference

- With this formulation, we get factorization as follows

$$\prod_j f_j(x_{C_j}) \tag{10.3}$$

which is possible to evaluate to unity iff the logic formula is satisfiable.

- If we take $p(x) \propto$ the above, we've got an MRF for SAT and we're done.
- Next, consider BN with N binary variables $\{x_i\}_{i=1}^N$ and M additional variables $\{y_j\}_{j=1}^M$ with M CPTS of the form:

$$p(y_j = 1|x_{C_j}) = \begin{cases} 1 & \text{if } f_j(x_{C_j}) = 1 \\ 0 & \text{else} \end{cases}, \text{ and for } x_i \; p(x_i = 1) = 0.5 \tag{10.4}$$

- This gives joint distribution that factorizes

$$p(x_{1:N}, y_{1:M}) = \prod_i p(x_i) \prod_j p(y_j|x_{C_j})$$

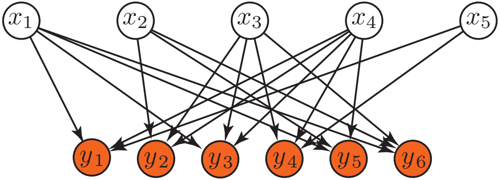
Hardness of Inference

- Create following BN, as evidence set use $y_j = 1$ for all $j \in 1 \dots M$
- Use max-sum semi-ring, so goal is to find the assignment to the x variables that maximize the joint probability.
- Resulting max evaluation is 1 iff original 3-SAT formula is satisfiable.

Hardness of Inference

- Example: $N = 5, M = 6$ in following 3-SAT formula and BN

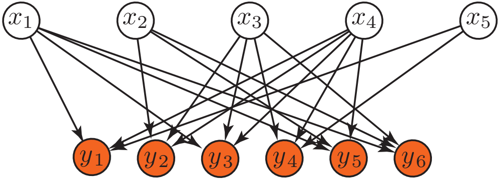
$(x_1 \vee x_4 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_4 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3$



Hardness of Inference

- Example: $N = 5, M = 6$ in following 3-SAT formula and BN

$$(x_1 \vee x_4 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_4 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x$$

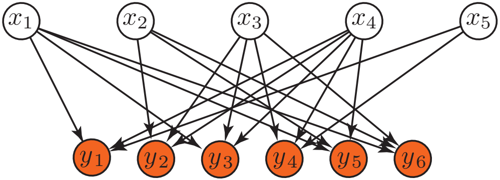


- MPE/Viterbi assignment to $x_{1:5}$ has non-zero probability iff original formula is SAT, BN inference (in general) NP-complete.

Hardness of Inference

- Example: $N = 5, M = 6$ in following 3-SAT formula and BN

$(x_1 \vee x_4 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_4 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3$

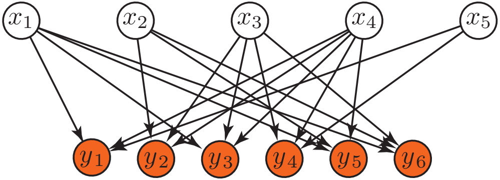


- MPE/Viterbi assignment to $x_{1:5}$ has non-zero probability iff original formula is SAT, BN inference (in general) NP-complete.
- **Doesn't** mean exact inference is always intractable, rather can't hope for a polynomial solution in all cases unless $P = NP$.

Hardness of Inference

- Example: $N = 5, M = 6$ in following 3-SAT formula and BN

$$(x_1 \vee x_4 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_4 \vee x_3) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3$$



- MPE/Viterbi assignment to $x_{1:5}$ has non-zero probability iff original formula is SAT, BN inference (in general) NP-complete.
- **Doesn't** mean exact inference is always intractable, rather can't hope for a polynomial solution in all cases unless $P = NP$.
- Moreover, even low tree-width graphs can be computationally challenging (i.e., large state space or random variable domain size).

Recap

- Time and memory complexity is $O(Nr^{\omega+1})$ where ω is the tree-width.

Recap

- Time and memory complexity is $O(Nr^{\omega+1})$ where ω is the tree-width.
- We can use conditioning (e.g., cutset conditioning) to get other points. E.g., condition on a set that renders the remainder of the set a tree. Same computation less memory.

Recap

- Time and memory complexity is $O(Nr^{\omega+1})$ where ω is the tree-width.
- We can use conditioning (e.g., cutset conditioning) to get other points. E.g., condition on a set that renders the remainder of the set a tree. Same computation less memory.
- Recursive conditioning (and similar such algorithms) allows is to get linear memory but a time complexity of $O(r^{(w+1)\log N})$.

Recap

- Time and memory complexity is $O(Nr^{\omega+1})$ where ω is the tree-width.
- We can use conditioning (e.g., cutset conditioning) to get other points. E.g., condition on a set that renders the remainder of the set a tree. Same computation less memory.
- Recursive conditioning (and similar such algorithms) allows is to get linear memory but a time complexity of $O(r^{(w+1)\log N})$.
- In general, many time-space tradeoffs for exact inference. Many algorithms along the achievable/unachievable frontier are SAT/CSP based, and use conditioning combined with various caching, and clause learning/deduction (e.g., nogood learning).

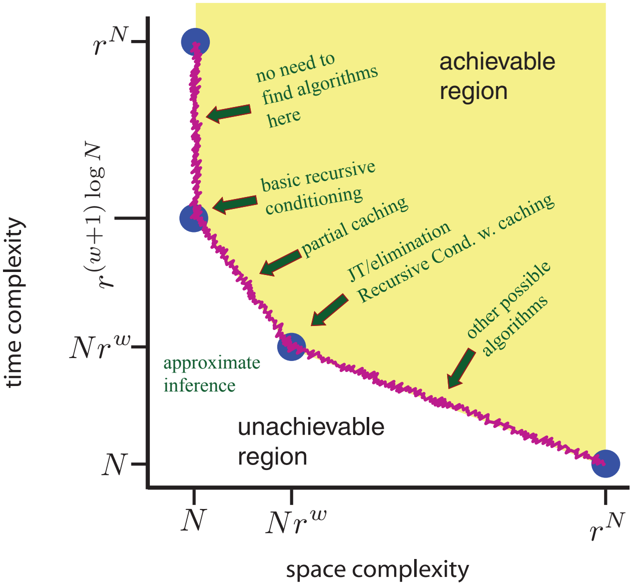
Recap

- Time and memory complexity is $O(Nr^{\omega+1})$ where ω is the tree-width.
- We can use conditioning (e.g., cutset conditioning) to get other points. E.g., condition on a set that renders the remainder of the set a tree. Same computation less memory.
- Recursive conditioning (and similar such algorithms) allows is to get linear memory but a time complexity of $O(r^{(w+1)\log N})$.
- In general, many time-space tradeoffs for exact inference. Many algorithms along the achievable/unachievable frontier are SAT/CSP based, and use conditioning combined with various caching, and clause learning/deduction (e.g., nogood learning).
- To get a better time/space profile, need to do approximation.

Recap

- Time and memory complexity is $O(Nr^{\omega+1})$ where ω is the tree-width.
- We can use conditioning (e.g., cutset conditioning) to get other points. E.g., condition on a set that renders the remainder of the set a tree. Same computation less memory.
- Recursive conditioning (and similar such algorithms) allows is to get linear memory but a time complexity of $O(r^{(w+1)\log N})$.
- In general, many time-space tradeoffs for exact inference. Many algorithms along the achievable/unachievable frontier are SAT/CSP based, and use conditioning combined with various caching, and clause learning/deduction (e.g., nogood learning).
- To get a better time/space profile, need to do approximation.
- For any given degree of distortion, there is a time/space tradeoff profile.

Time-Space Tradeoffs



Approximation: Two general approaches

- exact solution to approximate problem - approximate problem

Approximation: Two general approaches

- exact solution to approximate problem - approximate problem
  1. learning with or using a model with a structural restriction, structure learning, using a k-tree for a lower k than one knows is true. Make sure k is small enough so that exact inference can be performed, and make sure that, in that low tree-width model, one has best possible graph

Approximation: Two general approaches

- exact solution to approximate problem - approximate problem
  1. learning with or using a model with a structural restriction, structure learning, using a k-tree for a lower k than one knows is true. Make sure k is small enough so that exact inference can be performed, and make sure that, in that low tree-width model, one has best possible graph
  2. Functional restrictions to the model (i.e., use factors or potential functions that obey certain properties). Then certain fast algorithms (e.g., graph-cut) can be performed.

Approximation: Two general approaches

- exact solution to approximate problem - approximate problem
  1. learning with or using a model with a structural restriction, structure learning, using a k-tree for a lower k than one knows is true. Make sure k is small enough so that exact inference can be performed, and make sure that, in that low tree-width model, one has best possible graph
  2. Functional restrictions to the model (i.e., use factors or potential functions that obey certain properties). Then certain fast algorithms (e.g., graph-cut) can be performed.
- approximate solution to exact problem - approximate inference

Approximation: Two general approaches

- exact solution to approximate problem - approximate problem
  1. learning with or using a model with a structural restriction, structure learning, using a k-tree for a lower k than one knows is true. Make sure k is small enough so that exact inference can be performed, and make sure that, in that low tree-width model, one has best possible graph
  2. Functional restrictions to the model (i.e., use factors or potential functions that obey certain properties). Then certain fast algorithms (e.g., graph-cut) can be performed.
- approximate solution to exact problem - approximate inference
  1. Message or other form of propagation, variational approaches, LP relaxations, loopy belief propagation (LBP)

Approximation: Two general approaches

- exact solution to approximate problem - approximate problem
  1. learning with or using a model with a structural restriction, structure learning, using a k-tree for a lower k than one knows is true. Make sure k is small enough so that exact inference can be performed, and make sure that, in that low tree-width model, one has best possible graph
  2. Functional restrictions to the model (i.e., use factors or potential functions that obey certain properties). Then certain fast algorithms (e.g., graph-cut) can be performed.
- approximate solution to exact problem - approximate inference
  1. Message or other form of propagation, variational approaches, LP relaxations, loopy belief propagation (LBP)
  2. sampling (Monte Carlo, MCMC, importance sampling) and pruning (e.g., search based A*, score based, number of hypothesis based) procedures

Approximation: Two general approaches

- exact solution to approximate problem - approximate problem
  1. learning with or using a model with a structural restriction, structure learning, using a k-tree for a lower k than one knows is true. Make sure k is small enough so that exact inference can be performed, and make sure that, in that low tree-width model, one has best possible graph
  2. Functional restrictions to the model (i.e., use factors or potential functions that obey certain properties). Then certain fast algorithms (e.g., graph-cut) can be performed.
- approximate solution to exact problem - approximate inference
  1. Message or other form of propagation, variational approaches, LP relaxations, loopy belief propagation (LBP)
  2. sampling (Monte Carlo, MCMC, importance sampling) and pruning (e.g., search based A*, score based, number of hypothesis based) procedures
- Both methods only guaranteed approximate quality solutions.

Approximation: Two general approaches

- exact solution to approximate problem - approximate problem
  1. learning with or using a model with a structural restriction, structure learning, using a k-tree for a lower k than one knows is true. Make sure k is small enough so that exact inference can be performed, and make sure that, in that low tree-width model, one has best possible graph
  2. Functional restrictions to the model (i.e., use factors or potential functions that obey certain properties). Then certain fast algorithms (e.g., graph-cut) can be performed.
- approximate solution to exact problem - approximate inference
  1. Message or other form of propagation, variational approaches, LP relaxations, loopy belief propagation (LBP)
  2. sampling (Monte Carlo, MCMC, importance sampling) and pruning (e.g., search based A*, score based, number of hypothesis based) procedures
- Both methods only guaranteed approximate quality solutions.
- No longer in the achievable region in time-space tradoff graph, new set of time/space tradeoffs to achieve a particular accuracy.

Belief Propagation: message definition

Generic message definition

$$\mu_{i\to j}(x_j) = \sum_{x_i} \psi_{i,j}(x_i, x_j) \prod_{k\in\delta(i)\setminus\{j\}} \mu_{k\to i}(x_i) \tag{10.5}$$

- If graph is a tree, and if we obey MPP order, then we will reach a point where we've got marginals. I.e.,

$$p(x_i) \propto \prod_{j\in\delta(i)} \mu_{j\to i}(x_i) \tag{10.6}$$

and

$$p(x_i, x_j) \propto \psi_{i,j}(x_i, x_j) \prod_{k\in\delta(i)\setminus\{j\}} \mu_{k\to i}(x_i) \prod_{\ell\in\delta(j)\setminus\{i\}} \mu_{\ell\to j}(x_j) M \tag{10.7}$$

Belief Propagation: message definition w. unaries

Often, we see that nodes have potential functions as well. I.e., we have edge potentials $\psi_{i,j}(x_i, x_j)$ for $(i,j) \in E(G)$ and $\psi_i(x_i)$ for $i \in V(G)$. Also we might normalize each step (for numerical reasons). We get:

$$\mu_{i \to j}(x_j) \propto \sum_{x_i} \psi_{i,j}(x_i, x_j) \psi_i(x_i) \prod_{k \in \delta(i) \setminus \{j\}} \mu_{k \to i}(x_i) \tag{10.8}$$

such that $\mu_{i \to j}(x_j)$ sums to 1. If G is a tree, and we obey MPP, we get

$$p(x_i) \propto \psi_i(x_i) \prod_{j \in \delta(i)} \mu_{j \to i}(x_i) \tag{10.9}$$

and

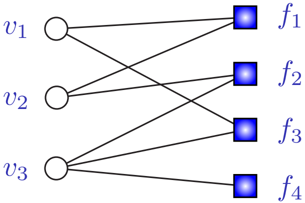
$$p(x_i, x_j) \propto \psi_{i,j}(x_i, x_j) \prod_{k \in \delta(i) \setminus \{j\}} \mu_{k \to i}(x_i) \prod_{\ell \in \delta(j) \setminus \{i\}} \mu_{\ell \to j}(x_j) \tag{10.10}$$

Belief Propagation: Generality

- So far, the "belief propagation" (BP) messages are done along edges, pairwise interaction, factors of the form $\psi_{ij}(x_i, x_j)$. What about higher order interaction $\psi_C(x_C)$ where $|C| > 2$?

Belief Propagation: Generality

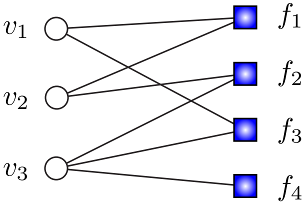
- So far, the "belief propagation" (BP) messages are done along edges, pairwise interaction, factors of the form $\psi_{ij}(x_i, x_j)$. What about higher order interaction $\psi_C(x_C)$ where $|C| > 2$?
- Recall a factor graph, where the factors themselves are represented on the right hand side of a bipartite graph.



$$p(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_1, x_3) f_4(x_3)$$

Belief Propagation: Generality

- So far, the "belief propagation" (BP) messages are done along edges, pairwise interaction, factors of the form $\psi_{ij}(x_i, x_j)$. What about higher order interaction $\psi_C(x_C)$ where $|C| > 2$?
- Recall a factor graph, where the factors themselves are represented on the right hand side of a bipartite graph.

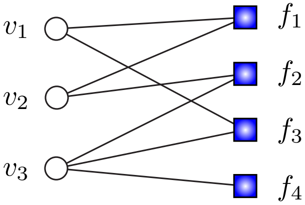


$$p(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_1, x_3) f_4(x_3)$$

- It is common to define a form of BP on a factor graph, going back and forth, between left and right nodes.

Belief Propagation: Generality

- So far, the "belief propagation" (BP) messages are done along edges, pairwise interaction, factors of the form $\psi_{ij}(x_i, x_j)$. What about higher order interaction $\psi_C(x_C)$ where $|C| > 2$?
- Recall a factor graph, where the factors themselves are represented on the right hand side of a bipartite graph.



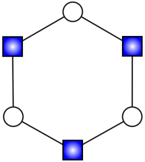
$$p(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_1, x_3) f_4(x_3)$$

- It is common to define a form of BP on a factor graph, going back and forth, between left and right nodes.
- Recall, an MRF doesn't distinguish between multiple pairwise interactions vs. one higher-order interaction.

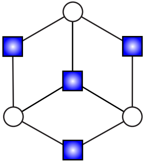
Generality and Specificity

- Consider the following three graphical models, the first two factor graphs and the third a MRF.

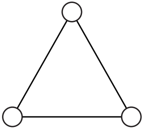
G_1



G_2

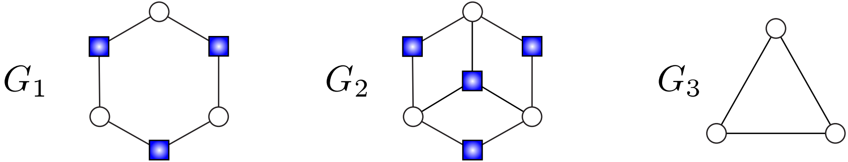


G_3



Generality and Specificity

- Consider the following three graphical models, the first two factor graphs and the third a MRF.

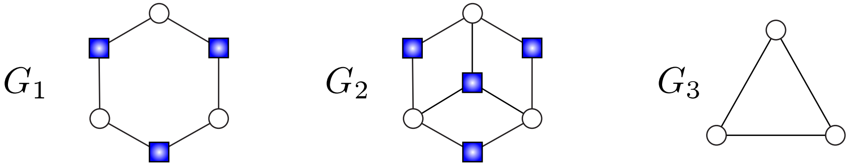


- Left: any distribution that can be written as

$$p_1(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_3, x_1) \tag{10.11}$$

Generality and Specificity

- Consider the following three graphical models, the first two factor graphs and the third a MRF.



- Left: any distribution that can be written as

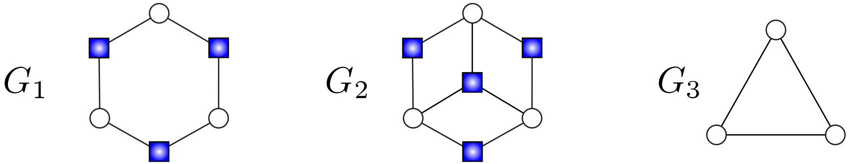
$$p_1(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_3, x_1) \tag{10.11}$$

- Center: any distribution that can be written as

$$p_2(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_3, x_1) f_4(x_1, x_2, x_3) \tag{10.12}$$

Generality and Specificity

- Consider the following three graphical models, the first two factor graphs and the third a MRF.



- Left: any distribution that can be written as

$$p_1(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_3, x_1) \tag{10.11}$$

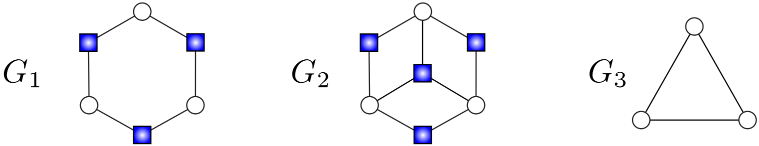
- Center: any distribution that can be written as

$$p_2(x_1, x_2, x_3) = f_1(x_1, x_2) f_2(x_2, x_3) f_3(x_3, x_1) f_4(x_1, x_2, x_3) \tag{10.12}$$

- example

$$\log p(x_1, x_2, x_3) = c + c_{12} x_1 x_2 + c_{23} x_2 x_3 + c_{13} x_1 x_3 + c_{123} x_1 x_2 x_3 \tag{10.13}$$

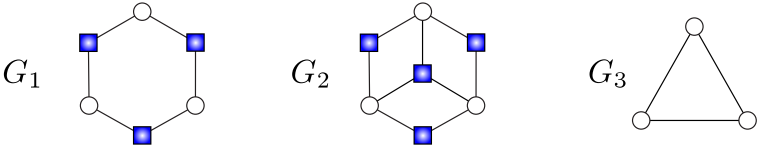
Generality and Specificity



- Right figure: all distributions that can be written:

$$p_3(x_1, x_2, x_3) = \psi(x_1, x_2, x_3) \tag{10.14}$$

Generality and Specificity

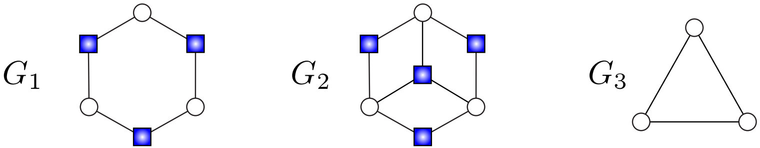


- Right figure: all distributions that can be written:

$$p_3(x_1, x_2, x_3) = \psi(x_1, x_2, x_3) \tag{10.14}$$

- We have $p_1, p_2, p_3 \in \mathcal{F}(G_2, \mathcal{M}^{(\text{fg})})$ and that $p_1 \in \mathcal{F}(G_1, \mathcal{M}^{(\text{fg})})$ but that $p_2, p_3 \notin \mathcal{F}(G_1, \mathcal{M}^{(\text{fg})})$. Moreover, it is clear that $p_1, p_2, p_3 \in \mathcal{F}(G_3, \mathcal{M}^{(\text{f})})$.

Generality and Specificity



- Right figure: all distributions that can be written:

$$p_3(x_1, x_2, x_3) = \psi(x_1, x_2, x_3) \tag{10.14}$$

- We have $p_1, p_2, p_3 \in \mathcal{F}(G_2, \mathcal{M}^{(\text{fg})})$ and that $p_1 \in \mathcal{F}(G_1, \mathcal{M}^{(\text{fg})})$ but that $p_2, p_3 \notin \mathcal{F}(G_1, \mathcal{M}^{(\text{fg})})$. Moreover, it is clear that $p_1, p_2, p_3 \in \mathcal{F}(G_3, \mathcal{M}^{(\text{f})})$.
- Can we stay with an MRF with this limitation (i.e., MRF's inability to discern order of interaction amongst variables in a clique)?

Pairwise MRF representing higher order MRF

- transform an MRF with higher order potentials to an MRF with only pairwise potentials, but with more variables.

Pairwise MRF representing higher order MRF

- transform an MRF with higher order potentials to an MRF with only pairwise potentials, but with more variables.
- Suppose we have $\psi_C(x_C)$ where $|C| > 2$. Define a new (single, scalar) variable z_C where $z_C \in \mathsf{D}_{Z_C}$ and where $|\mathsf{D}_{Z_C}| = |\mathsf{D}_{X_C}|$.

Pairwise MRF representing higher order MRF

- transform an MRF with higher order potentials to an MRF with only pairwise potentials, but with more variables.
- Suppose we have $\psi_C(x_C)$ where $|C| > 2$. Define a new (single, scalar) variable z_C where $z_C \in \mathsf{D}_{Z_C}$ and where $|\mathsf{D}_{Z_C}| = |\mathsf{D}_{X_C}|$.
- Each scalar value $z_C \in \mathsf{D}_{Z_C}$ represents a vector of values $x_C \in \mathsf{D}_{X_C}$, and let $x_i(z_C)$ represent the value of x_i associated with z_C, and let $z_C(x_C)$ represent the value of z_C corresponding to vector x_C.

Pairwise MRF representing higher order MRF

- transform an MRF with higher order potentials to an MRF with only pairwise potentials, but with more variables.
- Suppose we have $\psi_C(x_C)$ where $|C| > 2$. Define a new (single, scalar) variable z_C where $z_C \in \mathsf{D}_{Z_C}$ and where $|\mathsf{D}_{Z_C}| = |\mathsf{D}_{X_C}|$.
- Each scalar value $z_C \in \mathsf{D}_{Z_C}$ represents a vector of values $x_C \in \mathsf{D}_{X_C}$, and let $x_i(z_C)$ represent the value of x_i associated with z_C, and let $z_C(x_C)$ represent the value of z_C corresponding to vector x_C.
- Remove all edges between variables in x_C and add pairwise factors (and edges) of the form $\psi_{z_C,x_i}(z_C, x_i)$ for $i \in C$ where $\psi_{z_C,x_i}(z_C, x_i) = \mathbf{1}\{x_i = x_i(z_C)\}$.

Pairwise MRF representing higher order MRF

- transform an MRF with higher order potentials to an MRF with only pairwise potentials, but with more variables.
- Suppose we have $\psi_C(x_C)$ where $|C| > 2$. Define a new (single, scalar) variable z_C where $z_C \in \mathsf{D}_{Z_C}$ and where $|\mathsf{D}_{Z_C}| = |\mathsf{D}_{X_C}|$.
- Each scalar value $z_C \in \mathsf{D}_{Z_C}$ represents a vector of values $x_C \in \mathsf{D}_{X_C}$, and let $x_i(z_C)$ represent the value of x_i associated with z_C, and let $z_C(x_C)$ represent the value of z_C corresponding to vector x_C.
- Remove all edges between variables in x_C and add pairwise factors (and edges) of the form $\psi_{z_C,x_i}(z_C, x_i)$ for $i \in C$ where $\psi_{z_C,x_i}(z_C, x_i) = \mathbf{1}\{x_i = x_i(z_C)\}$.
- Create new unary factor $\psi_Z(z_C) = \psi(x_1(z_C), x_2(z_C), \dots)$.

Pairwise MRF representing higher order MRF

- transform an MRF with higher order potentials to an MRF with only pairwise potentials, but with more variables.
- Suppose we have $\psi_C(x_C)$ where $|C| > 2$. Define a new (single, scalar) variable z_C where $z_C \in \mathsf{D}_{Z_C}$ and where $|\mathsf{D}_{Z_C}| = |\mathsf{D}_{X_C}|$.
- Each scalar value $z_C \in \mathsf{D}_{Z_C}$ represents a vector of values $x_C \in \mathsf{D}_{X_C}$, and let $x_i(z_C)$ represent the value of x_i associated with z_C, and let $z_C(x_C)$ represent the value of z_C corresponding to vector x_C.
- Remove all edges between variables in x_C and add pairwise factors (and edges) of the form $\psi_{z_C,x_i}(z_C, x_i)$ for $i \in C$ where $\psi_{z_C,x_i}(z_C, x_i) = \mathbf{1}\{x_i = x_i(z_C)\}$.
- Create new unary factor $\psi_Z(z_C) = \psi(x_1(z_C), x_2(z_C), \dots)$.
- Then model of the form

$$\dots \psi_C(x_C) \dots$$

has same function as a model but of the form

$$\dots \psi_Z(z_C) \prod_{i \in C} \psi_{z_C,x_i}(z_C, x_i) \dots$$

uses only pairwise factors.

Choices for dealing with higher order factors in MRFs

So, to deal with MRFs with higher order factors, we can:

1. transform MRF to have only pairwise interactions, add more variables, we can keep using BP on MRF edges (as done above), makes the math a bit easier, does not change fundamental computational cost. Possible since for any given p, we know the interaction terms.

For the remainder of this term, we'll assume we've done the pair-wise transformation (i.e., option 1 above).

Choices for dealing with higher order factors in MRFs

So, to deal with MRFs with higher order factors, we can:

1. transform MRF to have only pairwise interactions, add more variables, we can keep using BP on MRF edges (as done above), makes the math a bit easier, does not change fundamental computational cost. Possible since for any given p, we know the interaction terms.
2. Alternatively, we can define BP on factor graphs.

For the remainder of this term, we'll assume we've done the pair-wise transformation (i.e., option 1 above).

Choices for dealing with higher order factors in MRFs

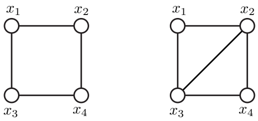
So, to deal with MRFs with higher order factors, we can:

1. transform MRF to have only pairwise interactions, add more variables, we can keep using BP on MRF edges (as done above), makes the math a bit easier, does not change fundamental computational cost. Possible since for any given p, we know the interaction terms.
2. Alternatively, we can define BP on factor graphs.
3. Alternatively, could define BP directly on the maxcliques of the MRF (but maxcliques are not easy to get in a MRF when not triangulated).

For the remainder of this term, we'll assume we've done the pair-wise transformation (i.e., option 1 above).

Reparameterization

- We start with a general $p \in \mathcal{F}(G, \mathcal{M}^{(\mathsf{f})})$ in terms of factors that might not alone have any inherent meaning or normalization.
- Goal: We *reparamterize* p so that the factor decomposition is the same but the factors are now marginals – marginal reparameterization
- Tree graph is such that we can reparameterize so that the edges and nodes are true marginals. e.g., $\phi_i(x_i) = \sum_{x_{V \setminus \{i\}}} p(x)$.
- Can we always do this? Only when graph is triangulated and we do it in terms of cliques and separators. When graph is not triangulated, not possible in general to do this. Eg., 4-cycle.



Reparameterization

- In a tree, we achieve true marginal reparameterization by sending messages according to MPP until all messages are sent in both directions.
- Alternatively, we could, say, initialize all messages to unity $\mu_{i \to j}(x_j) = 1$ or some other set of values, and sending *all* messages in parallel. Each parallel send of all message is considered one step.
- Let D be the diameter of the tree (length of longest path).
- Once we have done D steps, we will have "converged." Any additional messages will not change the state.
- If we have a tree, we have achieved marginal reparameterization.

State representation

- Consider the set of messages $\{\mu_{i\to j}(x_j)\}_{i,j}$ as a large state vector μ^t with $2|E(G)|r$ scalar elements.
- Each sent message moves the state vector from μ^t at time t to μ^{t+1} at next time step.
- A parallel message (sending multiple messages at the same time) moves the state vector as well.
- Convergence means that any set or subset of messages sent in parallel is such that $\mu^{t+1} = \mu^t$.

Messages as matrix multiply

$$\mu_{i\to j}(x_j) \propto \sum_{x_i} \psi_{i,j}(x_i, x_j)\psi_i(x_i) \prod_{k\in\delta(i)\setminus\{j\}} \mu_{k\to i}(x_i) \tag{10.15}$$

$$= \sum_{x_i} \psi'_{i,j}(x_i, x_j)\mu_{\neg j\to i}(x_i) \tag{10.16}$$

$$= (\psi'_{i,j})^T \mu_{\neg j\to i} \tag{10.17}$$

- Here, $\psi'_{i,j}$ is a matrix and $\mu_{\neg j\to i}$ is a column vector.
- Going from state μ^t to μ^{t+1} is like matrix-vector multiply — group messages from μ^t together into one vector representing $\mu_{\neg j\to i}$ for each $(i, j) \in E$, do the matrix-vector update, and store result in new state vector μ^{t+1}.
- If G is tree, μ^t will converged after D steps.

Belief Propagation and Cycles

What if graph has cycles?

- MPP causes deadlock since there is no way to start sending messages
- Like before, we can assume that messages have an initial state, e.g., $\mu_{i \to j}(x_j) = 1$ for all $(i, j) \in E(G)$ - note this is bi-directional. This breaks deadlock.
- We can then start sending messages. Will we converge after D steps? What does D even mean here?

Belief Propagation and Cycles

What if graph has cycles?

- MPP causes deadlock since there is no way to start sending messages
- Like before, we can assume that messages have an initial state, e.g., $\mu_{i \to j}(x_j) = 1$ for all $(i, j) \in E(G)$ - note this is bi-directional. This breaks deadlock.
- We can then start sending messages. Will we converge after D steps? What does D even mean here?
- No, in fact we could oscillate forever.

Sources for Today's Lecture

- Most of this material comes from a variety of sources. Best place to look is in our standard reading material.