CPSC 440: Advanced Machine Learning

Inference in Graphical Models

Mark Schmidt

University of British Columbia

Winter 2022

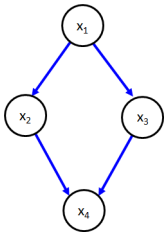
Inference in General DAGs

- If we try to generalize the CK equations to DAGs we obtain

$$p(x_j = s) = \sum_{x_{\mathsf{pa}(j)}} p(x_j = s, x_{\mathsf{pa}(j)}) = \sum_{x_{\mathsf{pa}(j)}} \underbrace{p(x_j = s \mid x_{\mathsf{pa}(j)})}_{\text{given}} p(x_{\mathsf{pa}(j)}).$$

- What goes wrong if nodes have multiple parents?
  - The expression $p(x_{\mathsf{pa}(j)})$ is a joint distribution depending on multiple variables.

- Consider the non-tree graph:



Inference in General DAGs

- We can compute $p(x_4)$ in this non-tree using:

$$\begin{aligned} p(x_4) &= \sum_{x_3}\sum_{x_2}\sum_{x_1} p(x_1, x_2, x_3, x_4) \\ &= \sum_{x_3}\sum_{x_2}\sum_{x_1} p(x_4 \mid x_2, x_3)p(x_3 \mid x_1)p(x_2 \mid x_1)p(x_1) \\ &= \sum_{x_3}\sum_{x_2} p(x_4 \mid x_2, x_3) \underbrace{\sum_{x_1} p(x_3 \mid x_1)p(x_2 \mid x_1)p(x_1)}_{M_{23}(x_2,x_3)} \end{aligned}$$

- Dependencies between $\{x_1, x_2, x_3\}$ mean our message depends on two variables.

$$\begin{aligned} p(x_4) &= \sum_{x_3}\sum_{x_2} p(x_4 \mid x_2, x_3)M_{23}(x_2, x_3) \\ &= \sum_{x_3} M_{34}(x_3, x_4), \end{aligned}$$

Inference in General DAGs

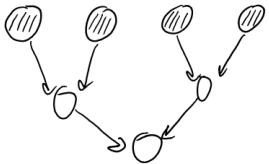
- With 2-variable messages, our cost increases to $O(dk^3)$.
- If we add the edge $x_1 -> x_4$, then the cost is $O(dk^4)$.

(the same cost as enumerating all possible assignments)

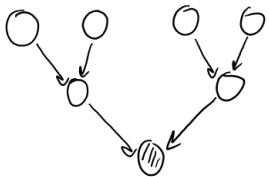
- Unfortunately, cost is not as simple as counting number of parents.
  - Even if each node has 2 parents, we may need huge messages.
  - Decoding is NP-hard and computing marginals is #P-hard in general.
  - We'll see later that maximum message size is "treewidth" of a particular graph.
- On the other hand, ancestral sampling is easy:
  - We can obtain Monte Carlo estimates of solutions to these NP-hard problems.

Conditional Sampling in DAGs

- What about conditional sampling in DAGs?
  - Could be easy or hard depending on what we condition on.
- For example, easy if we condition on the first variables in the order:
  - Just fix these and run ancestral sampling.



- Hard to condition on the last variables in the order:
  - Conditioning on descendent makes ancestors dependent.



Outline

1 DAG Inference

2 Structure Learning

3 More UGMs

4 Treewidth

5 ICM

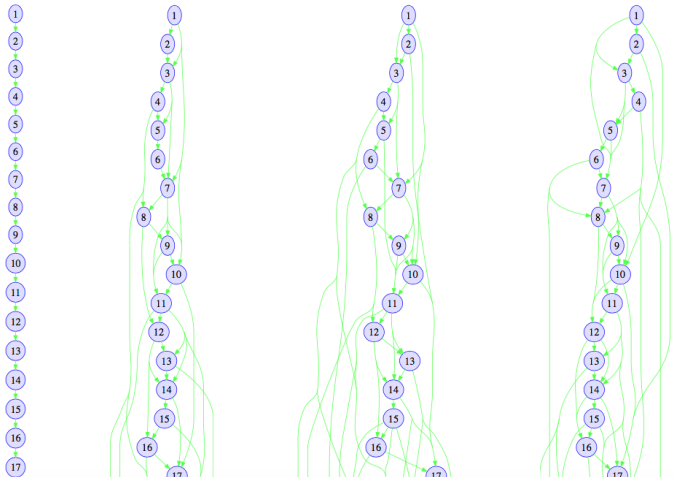
6 Block Inference

DAG Structure Learning

- Structure learning is the problem of choosing the graph.
  - Input is data X.
  - Output is a graph G.
- The "easy" case is when we're given the ordering of the variables.
  - So the parents of j must be chosen from $\{1, 2, \ldots, j-1\}$.
- Given the ordering, structure learning reduces to feature selection:
  - Select features $\{x_1, x_2, \ldots, x_{j-1}\}$ that best predict "label" x_j.
  - We can use any feature selection method to solve these d problems.

Example: Structure Learning in Rain Data Given Ordering

- Structure learning in rain data using L1-regularized logistic regression.
  - For different λ values, assuming chronological ordering.



DAG Structure Learning without an Ordering

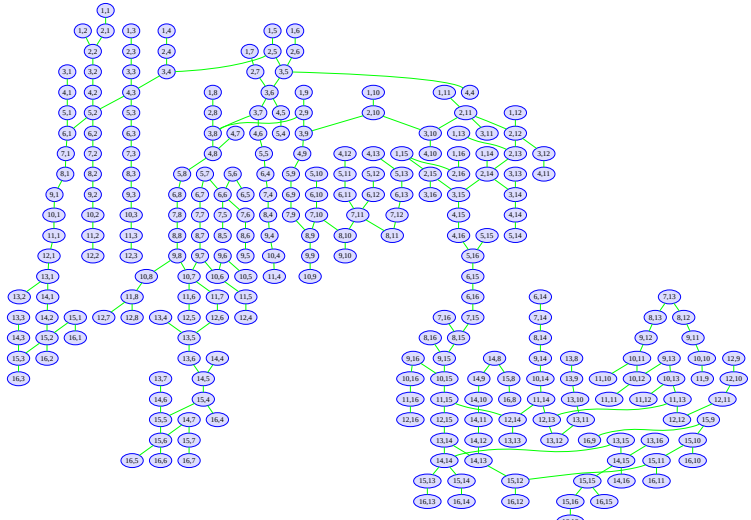
- Without an ordering, a common approach is "search and score"
  - Define a score for a particular graph structure (like BIC or other L0-regularizers).
  - Search through the space of possible DAGs.
    - "DAG-Search": at each step greedily add, remove, or reverse an edge.

- May have equivalent graphs with the same score (don't trust edge direction).
  - Do not interpret causally a graph learned from data.

- Structure learning is NP-hard in general, but finding the optimal tree is poly-time:
  - For symmetric scores, can be found by minimum spanning tree ("Chow-Liu").
    - Score is symmetric if score($x_j \to x_{j'}$) is the same as score($x_{j'} \to x_j$).
  - For asymetric scores, can be found by minimum spanning arborescence.

Structure Learning on USPS Digits

An optimal tree on USPS digits (16 by 16 images of digits).



20 Newsgroups Data

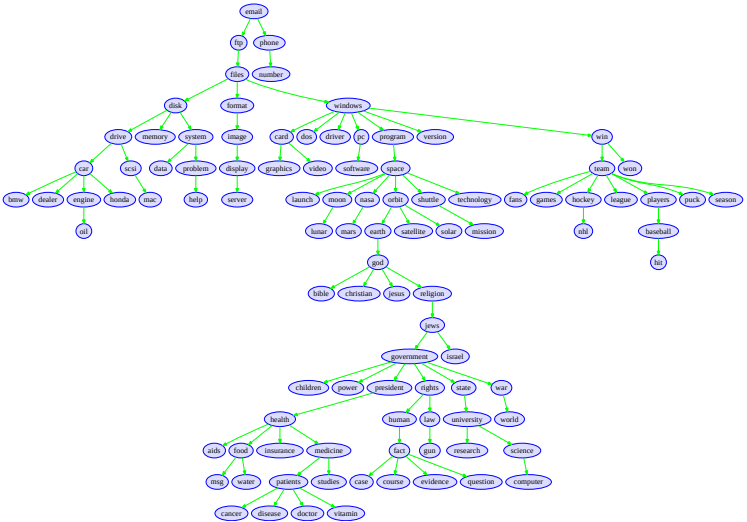
- Data containing presence of 100 words from newsgroups posts:

car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

- Structure learning should give some relationship between word occurrences.

Structure Learning on News Words

Optimal tree on newsgroups data:



"Constraint-Based" DAG Structure Learning

- Another common structure learning approach is "constraint-based":
  - Based on performing a sequence of conditional independence tests.
  - Prune edge between x_i and x_j if you find variables S making them independent,

$$x_i \perp x_j \mid x_S.$$

  - Challenge is considering exponential number of sets x_S (heuristic: "PC algorithm").
  - Assumes "faithfulness" (all independences are reflected in graph).
    - Otherwise it's weird (a duplicated feature would be disconnected from everything.)

Outline

1 DAG Inference

2 Structure Learning

3 More UGMs

4 Treewidth

5 ICM

6 Block Inference

Gaussians as Undirected Graphical Models

- Multivariate Gaussian can be written as

$$p(x) \propto \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \propto \exp\left(-\frac{1}{2}x^T\Sigma^{-1}x + x^T\underbrace{\Sigma^{-1}\mu}_{v}\right),$$

and writing it in summation notation we can see that it's a pairwise UGM:

$$p(x) \propto \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}x_i x_j(\Sigma^{-1})_{ij} + \sum_{i=1}^{d}x_i v_i\right)$$

$$= \left(\prod_{i=1}^{d}\prod_{j=1}^{d}\underbrace{\exp\left(-\frac{1}{2}x_i x_j(\Sigma^{-1})_{ij}\right)}_{\phi_{ij}(x_i,x_j)}\right)\left(\prod_{i=1}^{d}\underbrace{\exp\left(x_i v_i\right)}_{\phi_i(x_i)}\right)$$

- Above we include all edges. You can "remove" edges by setting $(\Sigma^{-1})_{ij} = 0$.
- "Gaussian graphical model" (GGM) or "Gaussian Markov random field" (GMRF).

General Pairwise UGM

- For general discrete x_i a generalization of Ising models is

$$p(x_1, x_2, \ldots, x_d) = \frac{1}{Z} \exp\left(\sum_{i=1}^{d} w_{i,x_i} + \sum_{(i,j) \in E} w_{i,j,x_i,x_j} \right),$$

  which can represent any "positive" pairwise UGM (meaning $p(x) > 0$ for all x).

- Interpretation of weights for this UGM:
  - If $w_{i,1} > w_{i,2}$ then we prefer $x_i = 1$ to $x_i = 2$.
  - If $w_{i,j,1,1} > w_{i,j,2,2}$ then we prefer $(x_i = 1, x_j = 1)$ to $(x_i = 2, x_j = 2)$.

- As before, we can use parameter tieing:
  - We could use the same w_{i,x_i} for all positions i.
  - Ising model corresponds to a particular parameter tieing of the w_{i,j,x_i,x_j}.

Label Propagation (Graph-Based Semi-Supervised) as a UGM

- Consider modeling the probability of a vector of labels $\bar{y} \in \mathbb{R}^t$ using

$$p(\bar{y}^1, \bar{y}^2, \ldots, \bar{y}^t) \propto \exp\left(-\sum_{i=1}^{n}\sum_{j=1}^{t} w_{ij}(y^i - \bar{y}^i)^2 - \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t} \bar{w}_{ij}(\bar{y}^i - \bar{y}^j)^2\right).$$

- Decoding in this model is the label propagation problem.

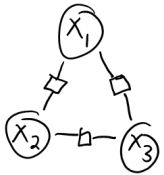
- This is a pairwise UGM:

$$\phi_j(\bar{y}^j) = \exp\left(-\sum_{i=1}^{n} w_{ij}(y^i - \bar{y}^j)^2\right), \quad \phi_{ij}(\bar{y}^i, \bar{y}^j) = \exp\left(-\frac{1}{2}\bar{w}_{ij}(\bar{y}^i - \bar{y}^j)^2\right).$$

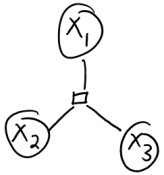
Factor Graphs

- Factor graphs are a way to visualize UGMs that distinguishes different orders.
  - Use circles for variables, squares to represent dependencies.

- Factor graph of $p(x_1, x_2, x_3) \propto \phi_{12}(x_1, x_2)\phi_{13}(x_1, x_3)\phi_{23}(x_2, x_3)$:



- Factor graph of $p(x_1, x_2, x_3) \propto \phi_{123}(x_1, x_2, x_3)$:



Other Graphical Models

- Factor graphs: we use a square between variables that appear in same factor.
  - Can distinguish between a 3-way factor and 3 pairwise factors.
- Chain-graphs: DAGs where each block can be a UGM.
- Ancestral-graph:
  - Generalization of DAGs that is closed under conditioning.
- Structural equation models (SEMs): generalization of DAGs that allows cycles.

Outline

1 DAG Inference

2 Structure Learning

3 More UGMs

4 Treewidth

5 ICM

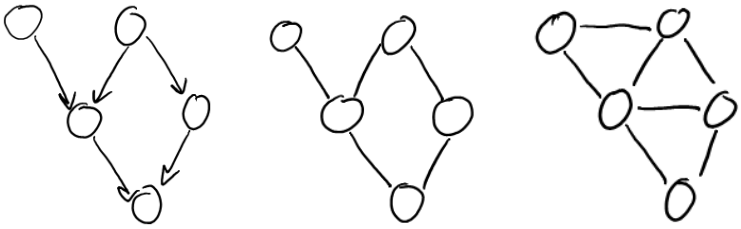
6 Block Inference

Moralization: Converting DAGs to UGMs

- To address the NP-hard problems, DAGs and UGMs use same techniques.
- We'll focus on UGMs, but we can convert DAGs to UGMs:

$$p(x_1, x_2, \dots, x_d) = \prod_{j=1}^{d} p(x_j | x_{\text{pa}(j)}) = \prod_{j=1}^{d} \underbrace{\phi_j(x_j, x_{\text{pa}(j)})}_{=p(x_j \mid x_{\text{pa}(j)})},$$

  which is a UGM with $Z = 1$.
- Graphically: we drop directions and "marry" parents (moralization).

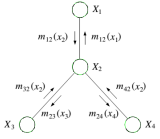


- May no longer see some independences, but doesn't change computational cost.

Easy Cases: Chains, Trees and Forests

- The forward-backward algorithm still works for chain-structured UGMs:
  - We compute the forward messages M and the backwards messages V.
  - With both M and V we can [conditionally] decode/marginalize/sample.

- Belief propagation generalizes this to trees (undirected graphs with no cycles):
  - Pick an arbitrary node as the "root", and order the nodes going away from the root.
    - Pass messages starting from the "leaves" going towards the root.
  - "Root" is like the last node in a Markov chain.
    - Backtrack from root to leaves to do decoding/sampling.
    - Send messages from the root going to the leaves to compute all marginals.



https://www.quora.com/

Probabilistic-graphical-models-what-are-the-relationships-between-sum-product-algorithm-belief-propagation-and-junction-tree-

Easy Cases: Chains, Trees and Forests

- Recall the CK equations in Markov chains:

$$M_c(x_c) = \sum_{x_p} p(x_c \mid x_p) M_p(x_p).$$

- For chain-structure UGMs we would have:

$$M_c(x_c) \propto \sum_{x_p} \phi(x_p)\phi(x_p, x_c) M_p(x_p).$$

- In tree-structured UGMs, parent p in the ordering may have multiple parents.
- Message coming from "neighbour" i that itself has neighbours j and k would be

$$M_{ic}(x_c) \propto \sum_{x_i} \phi_i(x_i)\phi_{ic}(x_i, x_c) M_{ji}(x_i) M_{ki}(x_i),$$

- Univariate marginals are proportional to $\phi_i(x_i)$ times all "incoming" messages.
  - The "forward" and "backward" Markov chain messages are a special case.
  - Replace $\sum_{x_i}$ with $\max_{x_i}$ for decoding.
    - "Sum-product" and "max-product" algorithms.

Exact Inference in UGMs

- For general graphs, the cost of message passing depends on
  1. Graph structure.
  2. Variable order.
- To see the effect of the order, consider Markov chain inference with bad ordering:

$$
\begin{aligned}
p(x_5) &= \sum_{x_5}\sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)p(x_5 \mid x_4) \\
&= \sum_{x_5}\sum_{x_1}\sum_{x_4}\sum_{x_3}\sum_{x_2} p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)p(x_5 \mid x_4) \\
&= \sum_{x_5}\sum_{x_1} p(x_1) \sum_{x_3}\sum_{x_4} p(x_4 \mid x_3)p(x_5 \mid x_4) \underbrace{\sum_{x_2} p(x_2 \mid x_1)p(x_3 \mid x_2)}_{M_{13}(x_1,x_3)}
\end{aligned}
$$

- So even though we have a chain, we have an M with k^2 values instead of k.
  - Increases cost to $O(dk^3)$ instead of $O(dk^2)$.
  - Inference can be exponentially more expensive with the wrong ordering.

Exact Inference in UGMs

- For general graphs, the cost of message passing depends on
  1. Graph structure.
  2. Variable order.
- As a non-tree example, consider computing Z in a simple 4-node cycle:

$$
\begin{aligned}
Z &= \sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1} \phi_{12}(x_1,x_2)\phi_{23}(x_2,x_3)\phi_{34}(x_3,x_4)\phi_{14}(x_1,x_4) \\
&= \sum_{x_4}\sum_{x_3} \phi_{34}(x_3,x_4) \sum_{x_2} \phi_{23}(x_2,x_3) \sum_{x_1} \phi_{12}(x_1,x_2)\phi_{14}(x_1,x_4) \\
&= \sum_{x_4}\sum_{x_3} \phi_{34}(x_3,x_4) \sum_{x_2} \phi_{23}(x_2,x_3) M_{24}(x_2,x_4) \\
&= \sum_{x_4}\sum_{x_3} \phi_{34}(x_3,x_4) M_{34}(x_3,x_4) = \sum_{x_4} M_4(x_4).
\end{aligned}
$$

- We again have an M with k^2 values instead of k.
  - We can do inference tasks with this graph, but it costs $O(dk^3)$ instead of $O(dk^2)$.

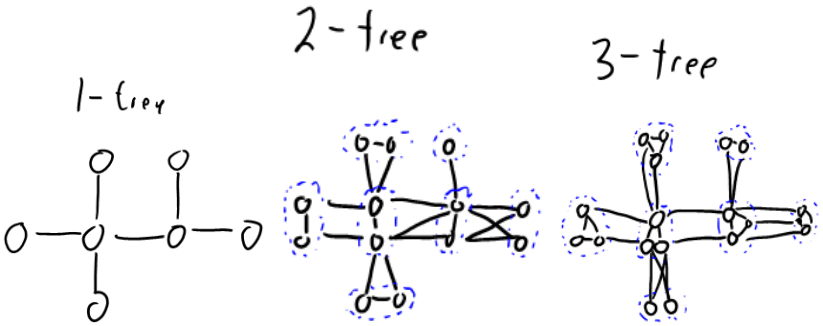
Variable Order and Treewidth

- Cost of message passing in general graphs is given by $O(dk^{\omega+1})$.
  - Here, ω is the number of dimensions of the largest message.
  - For trees, $\omega = 1$ so we get our usual cost of $O(dk^2)$.

- The minimum value of ω across orderings for a given graph is called treewidth.
  - In terms of graph: "minimum size of largest clique, minus 1, over all triangulations".
    - Also called "graph dimension" or "ω-tree".
  - Intuitively, you can think of low treewidth as being "close to a tree".
    - Trees have a treewidth of 1, and a single loop has a treewidth of 2.

Treewidth Examples

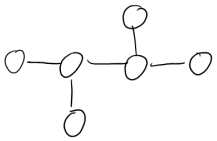
- Examples of k-trees:



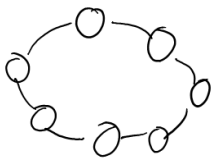
- 2-tree and 3-tree are trees if you use dotted circles to group nodes.

Treewidth Examples

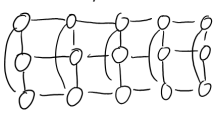
- Trees have $\omega = 1$, so with the right order inference costs $O(dk^2)$.



- A big loop has $\omega = 2$, so cost with the right ordering is $O(dk^3)$.



- The below grid-like structure has $\omega = 3$, so cost is $O(dk^4)$.



Variable Order and Treewidth

- Junction trees generalize belief propagation to general graphs (requires ordering).
  - This is the algorithm that achieves the $O(dk^{\omega+1})$ runtime.
- Computing ω and the optimal ordering is NP-hard.
  - But various heuristic ordering methods exist.
- An m_1 by m_2 lattice has $\omega = \min\{m_1, m_2\}$.
  - So you can do exact inference on "wide chains" with Junction tree.
  - But for 28 by 28 MNIST digits it would cost $O(784 \cdot 2^{29})$.
- Some links if you want to read about treewidth:
  - https://www.win.tue.nl/~nikhil/courses/2015/2WO08/treewidth-erickson.pdf
  - https://math.mit.edu/~apost/courses/18.204-2016/18.204_Gerrod_Voigt_final_paper.pdf
- For some graphs $\omega = (d-1)$ so there is no gain over brute-force enumeration.
  - Many graphs have high treewidth so we need approximate inference.

Outline

1 DAG Inference

2 Structure Learning

3 More UGMs

4 Treewidth

5 ICM

6 Block Inference

Iterated Conditional Mode (ICM)

- The iterated conditional mode (ICM) algorithm for approximate decoding:
  - On each iteration k, choose a variable j_t.
  - Maximie the joint probability in terms of x_{j_t} (with other variables fixed),

$$x_j^{t+1} \in \underset{c}{\text{argmax}}\, p(x_1^t, \ldots, x_{j-1}^t, x_j = c, x_{j+1}^t, \ldots, x_d^t).$$

- Equivalently, iterations correspond to finding mode of conditional $p(x_j \mid x_{-j}^t)$,

$$x_j^{t+1} \in \underset{c}{\text{argmax}}\, p(x_j = c \mid x_{-j}^t),$$

where x_{-j} means "x_i for all i except x_j": $x_1, x_2, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d$.

ICM in Action

- Start with some initial value: $x^0 = \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix}$.
- Select random j like $j = 3$.
- Set j to maximize $p(x_3 \mid x^0_{-3})$: $x^1 = \begin{bmatrix} 2 & 2 & 1 & 1 \end{bmatrix}$.
- Select random j like $j = 1$.
- Set j to maximize $p(x_1 \mid x^1_{-1})$: $x^2 = \begin{bmatrix} 3 & 2 & 1 & 1 \end{bmatrix}$.
- Select random j like $j = 2$.
- Set j to maximize $p(x_2 \mid x^2_{-2})$: $x^3 = \begin{bmatrix} 3 & 2 & 1 & 1 \end{bmatrix}$.
- ...
- Repeat until you can no longer improve by single-variable changes.
  - Intead of random, could cycle through the variables in order.
  - Or you could greedily choose the variable that increases the probability the most.

Optimality and Globalization of ICM

- Does ICM find the global optimum?
- Decoding is usually non-convex, so doesn't find global optimum.
  - ICM is an approximate decoding method.
- There exist many globalization methods that can improve its performance:
  - Restarting with random initializations.
  - Global optimization methods:
    - Simulated annealing, genetic algorithms, ant colony optimization, GRASP, etc.

Using the Unnormalized Objective

- How can you maximize $p(x)$ in terms of x_j if evaluating it is NP-hard?
- Let's define the unnormalized probability $\tilde{p}$ as

$$\tilde{p}(x) = \prod_{c \in \mathcal{C}} \phi_c(x_c).$$

- So the normalized probability is given by

$$p(x) = \frac{\tilde{p}(x)}{Z}.$$

- In UGMs evaluating Z is hard but evaluating $\tilde{p}(x)$ is easy.
- And for decoding we only need unnormalized probabilities,

$$\operatorname*{argmax}_x p(x) \equiv \operatorname*{argmax}_x \frac{\tilde{p}(x)}{Z} \equiv \operatorname*{argmax}_x \tilde{p}(x),$$

so we can decode based on $\tilde{p}$ without knowing Z.

ICM Iteration Cost

- How much does ICM cost?

- Consider a pairwise UGM,

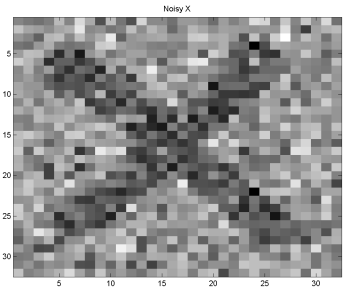
$$\tilde{p}(x) = \left(\prod_{j=1}^{d} \phi_j(x_j)\right)\left(\prod_{(i,j)\in E} \phi_{ij}(x_i, x_j)\right).$$

- Each ICM update would:
  1. Set $M_j(x_j = s)$ to product of terms in $\tilde{p}(x)$ involving x_j, with x_j set to s.
  2. Set x_j to the largest value of $M_j(x_j)$.

- The variable x_j has k values and appears in at most d factors here.
  - You can compute the k values of these d factors in $O(dk)$ to find the largest.
  - If you only have m nodes in "Markov blanket", this reduces to $O(mk)$.
    - We will define "Markov blanket" in a couple slides.

ICM in Action

Consider using a UGM for binary image denoising:



We have

- Unary potentials ϕ_j for each position.
- Pairwise potentials ϕ_{ij} for neighbours on grid.
- Parameters are trained as CRF (later).

Goal is to produce a noise-free binary image (show video).

Digression: Closure of UGMs under Conditioning

- UGMs are closed under conditioning:
  - If $p(x)$ is a UGM, then $p(x_A \mid x_B)$ can be written as a UGM (for partition A and B).
- Conditioning on x_2 and x_3 in a chain,

gives a UGM defined on x_1 and x_4 that is disconnected:

- Graphically, we "erase the black nodes and their edges".
- Notice that inference in the conditional UGM may be mucher easier.

Digression: Closure of UGMs under Conditioning

- Mathematically, a 4-node pairwise UGM with a chain structure assumes

$$p(x_1, x_2, x_3, x_4) \propto \phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4).$$

- Conditioning on x_2 and x_3 gives UGM over x_1 and x_4.

$$p(x_1, x_4 \mid x_2, x_3) = \frac{1}{Z'}\phi'_1(x_1)\phi'_4(x_4),$$

where new potentials "absorb" the shared potentials with observed nodes:

$$\phi'_1(x_1) = \phi_1(x_1)\phi_{12}(x_1, x_2), \quad \phi'_4(x_4) = \phi_4(x_4)\phi_{34}(x_3, x_4).$$

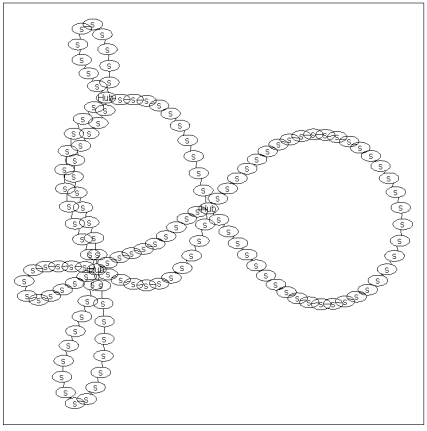
Conditioning in UGMs

- Conditioning on x_2 and x_3 in 4-node chain-UGM gives

$$
\begin{aligned}
p(x_1, x_4|x_2, x_3) &= \frac{p(x_1, x_2, x_3, x_4)}{p(x_2, x_3)} \\
&= \frac{\frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_1(x_1, x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \frac{1}{Z}\phi_1(x_1')\phi_2(x_2)\phi_3(x_3)\phi_4(x_4')\phi_1(x_1', x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4')} \\
&= \frac{\frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_1(x_1, x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4)}{\frac{1}{Z}\phi_2(x_2)\phi_3(x_3)\phi_2(x_2, x_3)\sum_{x_1', x_4'} \phi_1(x_1')\phi_4(x_4')\phi_1(x_1', x_2)\phi_3(x_3, x_4')} \\
&= \frac{\phi_1(x_1)\phi_4(x_4)\phi_1(x_1, x_2)\phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \phi_1(x_1')\phi_4(x_4')\phi_1(x_1', x_2)\phi_3(x_3, x_4')} \\
&= \frac{\phi_1'(x_1)\phi_4'(x_4)}{\sum_{x_1', x_4'} \phi_1'(x_1')\phi_4'(x_4')}
\end{aligned}
$$

Simpler Inference in Conditional UGMs

- Consider the following graph which could describe bus stops:



- If we condition on the "hubs", the graph forms a forest (and inference is easy).
  - Simpler inference after conditioning is used by many approximate inference methods.

Digression: Local Markov Property and Markov Blanket

- Approximate inference methods often use conditional $p(x_j \mid x_{-j})$,
  - where x^k_{-j} means "x^k_i for all i except x^k_j": $x^k_1, x^k_2, \ldots, x^k_{j-1}, x^k_{j+1}, \ldots, x^k_d$.

- In UGMs, the conditional simplifies due to conditional independence,

$$p(x_j \mid x_{-j}) = p(x_j \mid x_{\text{nei}(j)}),$$

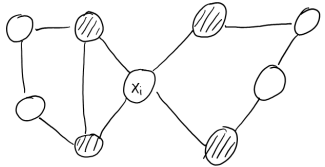
  this local Markov property means conditional only depends on neighbours.

- We say that the neighbours of x_j are its "Markov blanket".

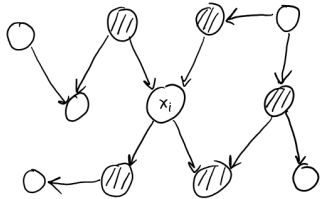
- Markov blanket is the set nodes that make you independent of all other nodes.

Digression: Local Markov Property and Markov Blanket

- In UGMs the Markov blanket is the neighbours.



- Markov blanket in DAGs: parents, children, co-parents (parents of same children):



Outline

1 DAG Inference

2 Structure Learning

3 More UGMs

4 Treewidth

5 ICM

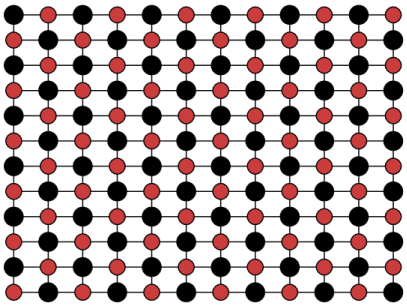
6 Block Inference

Block-Structured Approximate Inference

- Basic approximate inference methods like ICM and Gibb sampling:
  - Update one x_j at a time.
  - Efficient because conditional UGM is 1 node.
- Better approximate inference methods use block updates:
  - Update a block of x_j values at once.
  - Efficient if conditional UGM allows exact inference.
- If we choose the blocks cleverly, this works substantially better.

Block-Structured Approximate Inference

- Consider a lattice-structure and the following two blocks ("red-black ordering"):

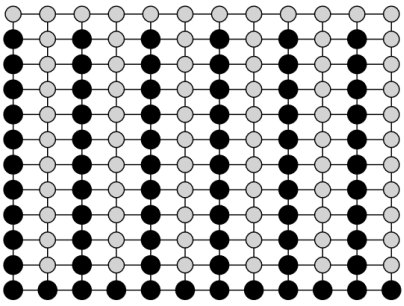


- Given black nodes, conditional UGM on red nodes is a disconnected graph.
  - "I can optimally update the red nodes given the black nodes" (and vice versa).
    - You update $d/2$ nodes at once for cost of this is $O(dk)$, and easy to parallelize.

- Minimum number of blocks to disconnect the graph is graph colouring.

Block-Structured Approximate Inference

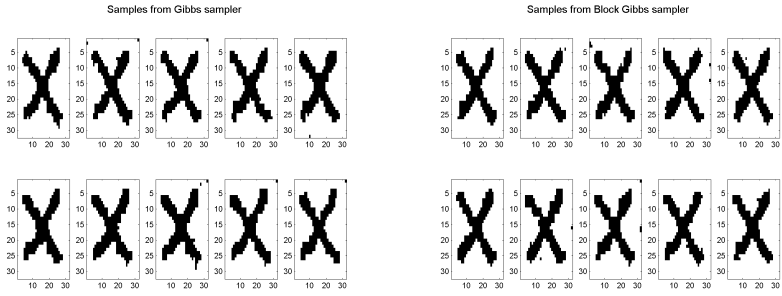
- We could also consider general forest-structured blocks:



- We can still optimally update the black nodes given the gray nodes in $O(dk^2)$.
  - This works much better than "one at a time".

Block Gibbs Sampling in Action

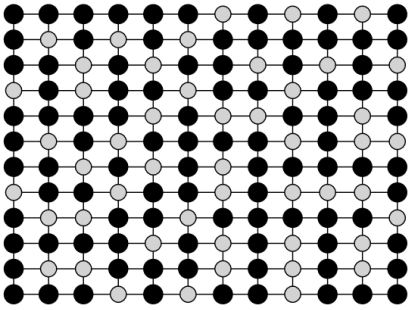
- Gibbs vs. tree-structured block-Gibbs samples:



- With block sampling, the samples are far less correlated.
- We can also do tree-structured block ICM.
  - Harder to get stuck if you get to update entire trees.

Block-Structured Approximate Inference

- Or we could define a new tree-structured block on each iteration:



- The above block updates around two thirds of the nodes optimally.

(Here we're updating the black nodes.)

Block ICM Based on Graph Cuts

- Consider a binary pairwise UGM with "attractive" potentials,

$$\log \phi_{ij}(1,1) + \log \phi_{ij}(2,2) \geq \log \phi_{ij}(1,2) + \log \phi_{ij}(2,1).$$

- In words: "neighbours prefer to have similar states".

- In this setting exact decoding can be formulated as a max-flow/min-cut problem.
  - Can be solved in polynomial time.

- This is widely-used computer vision:
  - Want neighbouring pixels/super-pixels/regions to be more likely to get same label.

Graph Cut Example: "GrabCut"

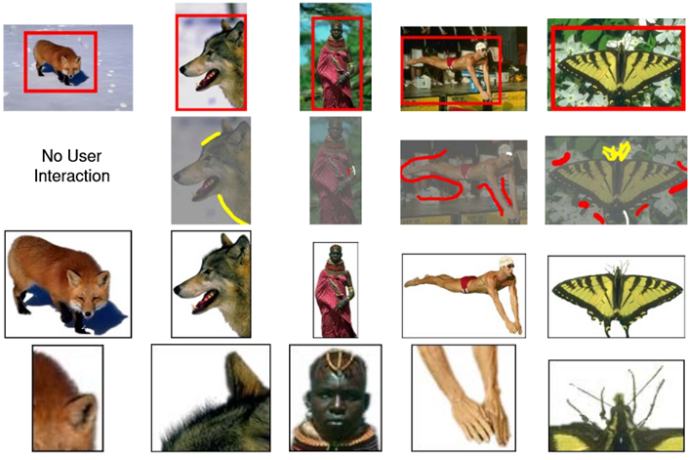
Figure 1: **Three examples of GrabCut.** The user drags a rectangle loosely around an object. The object is then extracted automatically.

http://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

1. User draws a box around the object they want to segment.
2. Fit Gaussian mixture model to pixels inside the box, and to pixels outside the box.
3. Construct a pairwise UGM using:
   - $\phi_i(x_i)$ set to GMM probability of pixel i being in class x_i.
   - $\phi_{ij}(x_i, x_j)$ set to Ising potential times RBF based on spatial/colour distance.
     - Use $w_{ij} > 0$ so the model is "attractive".
4. Perform exact decoding in the binary attractive model using graph cuts.

Graph Cut Example: "GrabCut"

- GrabCut with extra user interaction:



http://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

Alpha-Beta Swap and Alpha-Expansions: ICM with Graph Cuts

- If we have more than 2 states, we can't use graph cuts.
- Alpha-beta swaps are an approximate decoding method for "pairwise attractive",

$$\log \phi_{ij}(\alpha, \alpha) + \log \phi_{ij}(\beta, \beta) \geq \log \phi_{ij}(\alpha, \beta) + \log \phi_{ij}(\beta, \alpha).$$

  - Each step choose an α and β, optimally "swaps" labels among these nodes.
- Alpha-expansions are another variation based on a slightly stronger assumption,

$$\log \phi_{ij}(\alpha, \alpha) + \log \phi_{ij}(\beta_1, \beta_2) \geq \log \phi_{ij}(\alpha, \beta_1) + \log \phi_{ij}(\beta_2, \alpha).$$

  - Steps choose label α, and consider replacing the label of any node not labeled α.

Alpha-Beta Swap and Alpha-Expansions: ICM with Graph Cuts

- These don't find global optima in general, but make huge moves:

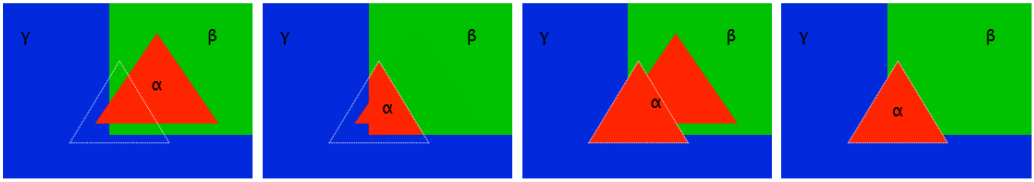


Figure 1: From left to right: Initial labeling, labeling after $\alpha\beta$-swap, labeling after α-expansion, labeling after α-expansion β-shrink. The optimal labeling of the α pixels is outlined by a white triangle, and is achieved from the initial labeling by one ~~α-expansion β-shrink move~~. ex-Swap move

- A somewhat-related MCMC method is the Swendson-Wang algorithm.

Example: Photomontage

- Photomontage: combining different photos into one photo:

http://vision.middlebury.edu/MRF/pdf/MRF-PAMI.pdf

- Here, x_i corresponds to identity of original image at position i.

Example: Photomontage

- Photomontage: combining different photos into one photo:

http://vision.middlebury.edu/MRF/pdf/MRF-PAMI.pdf