CPSC 440: Advanced Machine Learning

Approximate Inference

Mark Schmidt

University of British Columbia

Winter 2021

Last Lectures: Directed and Undirected Graphical Models

- We've discussed the most common classes of graphical models:
  - DAG models represent probability as ordered product of conditionals,

$$p(x) = \prod_{j=1}^{d} p(x_j \mid x_{\mathsf{pa}(j)}),$$

    and are also known as "Bayesian networks" and "belief networks".

  - UGMs represent probability as product of non-negative potentials ϕ_c,

$$p(x) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(x_c), \quad \text{with} \quad Z = \sum_x \prod_{c \in \mathcal{C}} \phi_c(x_c),$$

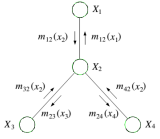
    and are also known as "Markov random fields" and "Markov networks".
    - "Partition function" Z makes all inference tasks hard in non-forest UGMs.

- Same exact/approximate inference methods work for both cases.
  - Can convert from DAG to UGM via moralization.

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

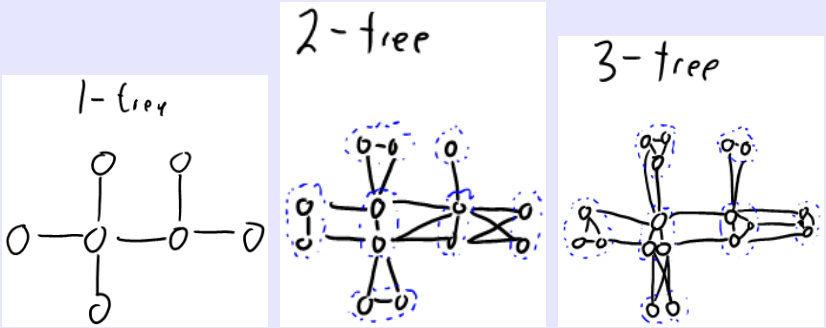
- Cost of message passing in general graphs is given by $O(dk^{\omega+1})$.
  - Here, ω is the number of dimensions of the largest message.
  - For trees, $\omega = 1$ so we get our usual cost of $O(dk^2)$.

- 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

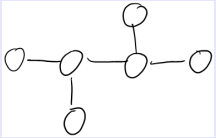
- Examples of k-trees:



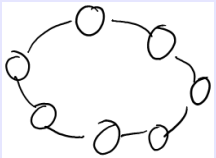
- 2-tree and 3-tree are trees if you use dotted circles to group nodes.

Treewidth Examples

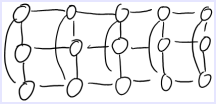
- Trees have $\omega = 1$, so with the right order inference costs $O(dk^2)$.



- 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

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,

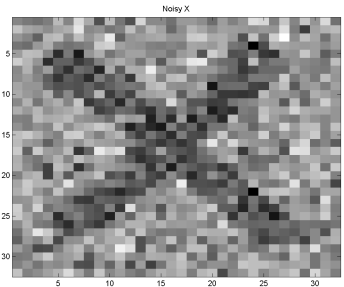
$$\tilde{p}(x) = \left(\prod_{j=1}^{d} \phi_j(x_j)\right)\left(\prod_{(i,j)\in E} \phi_{ij}(x_i, x_j)\right).$$

- 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 (tedious: bonus slide)

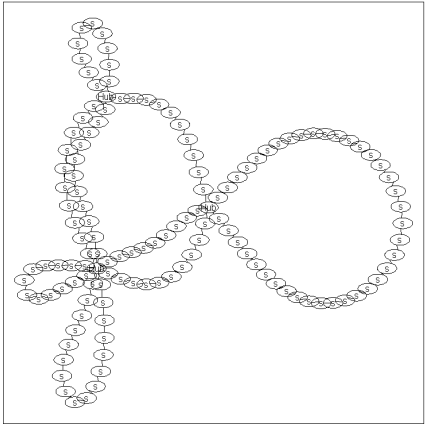
$$p(x_1, x_4 \mid x_2, x_3) = \frac{1}{Z'}\phi'_1(x_1)\phi'_4(x_4),$$

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

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

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

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

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

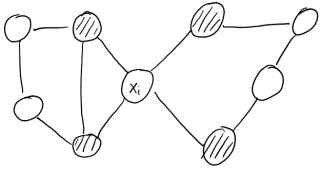
  this local Markov property means conditional only depends on neighbours.

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

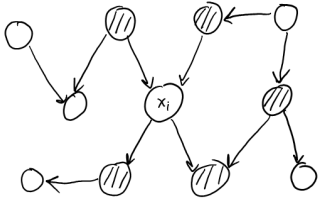
- Markov blanket is the set nodes that make you independent of all other nodes.

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):



Summary

- Message passing can be used for inference in UGMs.
  - Belief propagation for trees.
  - Cost might be exponential for unfavourable graphs/ordering.
    - Exponential in "treewidth" of graph.
- Conditioning in UGMs leads to a smaller/simpler UGM.
- Iterated conditional mode is coordinate descent for decoding UGMs.
  - Fast but doesn't obtain global optimum in general.
- Next time: our first MCMC method.

Conditioning in UGMs

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

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