

# CPSC 540: Machine Learning

## Approximate Inference

Mark Schmidt

University of British Columbia

Winter 2018

## 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,

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

and are also known as “Bayesian networks” and “belief networks”.

- **UGMs** represent probability as product of **non-negative potentials**  $\phi_c$ ,

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

and are also known as “Markov random fields” and “Markov networks”.

- We saw how to write Gaussians as UGMs, today we focus on **discrete**  $x_j$ .

## Discrete DAGs vs. Discrete UGMs

- Common **inference tasks** in graphical models:
  - 1 Compute  $p(x)$  for an assignment to the variables  $x$ .
  - 2 Generate a **sample**  $x$  from the distribution.
  - 3 Compute **univariate marginals**  $p(x_j)$ .
  - 4 Compute **decoding**  $\operatorname{argmax}_x p(x)$ .
  - 5 Compute **univariate conditional**  $p(x_j|x_{j'})$ .
- With discrete  $x_i$ , all of the above are easy in **tree-structured graphs**.
  - For DAGs, a tree-structured has **at most one parent**.
  - For UGMs, a tree-structured graph has **no cycles**.
- With discrete  $x_i$ , the above may be harder for **general graphs**:
  - In DAGs the first two are easy, the others are NP-hard.
  - In **UGMs all of these are NP-hard**.

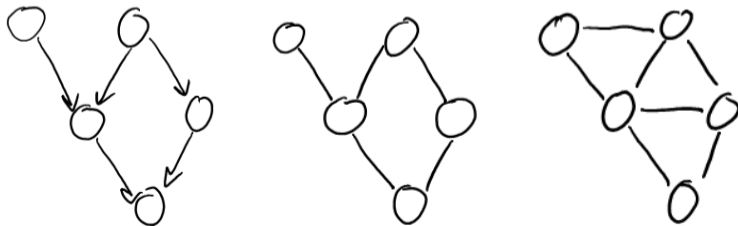
## Moralization: Converting DAGs to UGMs

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

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

which is a UGM with  $Z = 1$ .

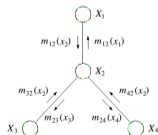
- Graphically: we drop directions and “marry” parents ([moralization](#)).



- May lose some conditional independences, but doesn't change computational cost.

## Easy Cases: Chains, Trees and Forests

- The **forward-backward** still **works for chain-structured UGMs**:
  - We compute the forward messages  $M$  and the backwards messages  $V$ .
  - With both  $M$  and  $V$  we can compute [conditionally] decode/marginalize/sample.
- **Belief propagation** generalizes this to **trees**:
  - 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/>

## Easy Cases: Chains, Trees and Forests

- In pairwise UGM, belief propagation “message” from parent  $p$  to child  $c$  is given by

$$M_{pc}(x_c) \propto \sum_{x_p} \phi_i(x_p) \phi_{pc}(x_p, x_c) M_{jp}(x_p) M_{kp}(x_p),$$

assuming that parent  $p$  has parents  $j$  and  $k$ .

- 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

- Message passing is also favourable in some other graph structures.
- For example, computing  $Z$  in a simple 4-node cycle could be done using:

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

- Message-passing cost depends on **graph structure** and the **order of the sums**.

## Exact Inference in UGMs

- To see the effect of the order, consider chain-structured UGM with a stupid order:

$$\begin{aligned}
 Z &= \sum_{x_5} \sum_{x_4} \sum_{x_3} \sum_{x_2} \sum_{x_1} \prod_{j=2}^5 \phi(x_j, x_{j-1}) \\
 &= \sum_{x_5} \sum_{x_3} \sum_{x_2} \sum_{x_4} \sum_{x_1} \prod_{j=2}^5 \phi(x_j, x_{j-1}) \\
 &= \sum_{x_5} \sum_{x_3} \sum_{x_2} \sum_{x_4} \prod_{j=3}^5 \phi(x_j, x_{j-1}) \underbrace{\sum_{x_1} \phi(x_2, x_1)}_{M_2(x_2)} \\
 &= \sum_{x_5} \sum_{x_3} \sum_{x_2} \phi(x_3, x_2) \underbrace{\sum_{x_4} \phi(x_4, x_3) \phi(x_5, x_4)}_{M_{235}(x_2, x_3, x_5)} M_2(x_2).
 \end{aligned}$$

- So even though we have a chain, we have an  $M$  with  $k^3$  values instead of  $k$ .

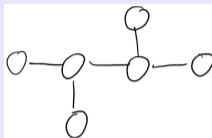


## Variable Order and Treewidth

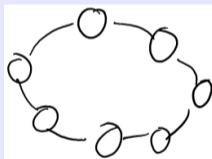
- So cost of message passing depends on
  - 1 Graph structure.
  - 2 Variable order.
- Cost of message passing is given by  $O(dk^{\omega+1})$ .
  - Here,  $\omega$  is the **size of the largest message**.
  - For trees,  $\omega = 1$  so we get our usual cost of  $O(dk^2)$ .
- The **minimum value of  $\omega$**  across orderings for a given graph is called **treewidth**.
  - In terms of graph: “minimum size of largest clique, minus 1, over all triangulations”.
  - An  $m_1$  by  $m_2$  lattice has  $\omega = \min\{m_1, m_2\}$ .
    - For 28 by 28 MNIST digits it would cost  $O(784 * 2^{29})$ .
  - For some graphs  $\omega = (d - 1)$  so there is no gain over brute-force enumeration.
- **Junction trees** generalize belief propagation to general graphs (require ordering).
- Computing  $\omega$  and the optimal ordering is NP-hard.
  - But various heuristic ordering methods exist.

## Variable Order and Treewidth

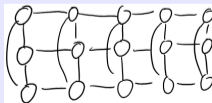
- Trees have  $\omega = 1$ , so with the right order inference costs  $O(dk^2)$ .



- 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  $O(dk^4)$ .



- Many graphs have high treewidth so we need **approximate inference**.

# Outline

- 1 Exact Inference in UGMs
- 2 ICM and Gibbs Sampling

## Iterated Conditional Mode (ICM)

- The **iterated conditional mode (ICM)** algorithm for **approximate decoding**:
  - On each iteration  $k$ , **choose a variable  $j_k$** .
  - **Optimize  $x_{j_k}$**  with the other variables held fixed.
- So ICM is **coordinate optimization**.
- Iterations correspond to finding **mode of conditional  $p(x_j | x_{-j}^k)$** ,

$$x_j^{k+1} \leftarrow \max_c p(x_j = c | x_{-j}^k),$$

where  $x_{-j}^k$  means “ $x_i^k$  for all  $i$  except  $x_j^k$ ”.

- 3 main issues:
  - ① How can we do this if evaluating  $p(x)$  is NP-hard?
  - ② Is coordinate optimization efficient for this problem?
  - ③ Does it find the global optimum?

## ICM Issue 1: Intractable Objective

- How can you optimize  $p(x)$  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 probability is given by

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

- Note that 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 decoded based on  $\tilde{p}$  without knowing  $Z$ .

## ICM Issue 2: Efficiency

- Is coordinate optimization efficient for this problem?
- Consider a **pairwise UGM**,

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

or

$$\log p(x) = \sum_{j=1}^d \log \phi_j(x_j) + \sum_{(i,j) \in E} \log \phi_{ij}(x_i, x_j) + \text{constant}.$$

which is a special case of

$$f(x) = \sum_{j=1}^d f_j(x_j) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j),$$

which is one of the problem where **coordinate optimization is  $n$ -times faster**.

## Digression: Local Markov Property and Markov Blanket

- In UGMs, **conditional independence is determined by reachability**.
  - $A \perp B \mid C$  if all paths from  $A$  to  $B$  are blocked by  $C$ .

- This implies a **local Markov property**,

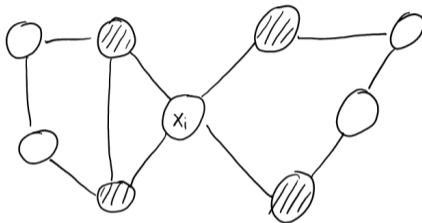
$$p(x_j \mid x_{1:d}) = p(x_j \mid x_{\text{nei}(j)}),$$

that we're independent of all non-neighbours given neighbours in the graph.

- We say that the **neighbours of  $x_j$  are its “Markov blanket”**.

## Digression: Local Markov Property and Markov Blanket

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



- In UGMs the Markov blanket is the neighbours.
- Graphically, **ICM is efficient because update only depends on Markov blanket.**
  - And even if graph is fully-connected, update only depends on edges to neighbours.



## Pseudo-Code for ICM

- Consider a **pairwise UGM**:

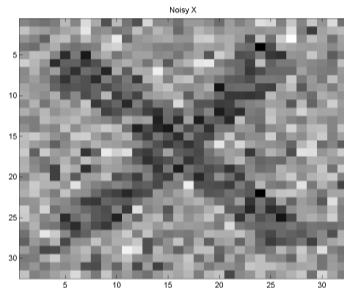
$$p(x_1, x_2, \dots, x_d) \propto \left( \prod_{i=1}^d \phi_i(x_i) \right) \left( \prod_{(i,j) \in E} \phi_{ij}(x_i, x_j) \right),$$

- For node  $i$  with 2 neighbours  $j$  and  $k$ , ICM update would be:

- 1 Compute  $M_i(x_i) = \phi_i(x_i) \underbrace{\phi_{ij}(x_i, x_j) \phi_{ik}(x_i, x_k)}_{\text{edges in Markov blanket}}$  for all  $x_i$ .
- 2 Set  $x_i$  to the largest value of  $M_i(x_i)$ .

## ICM in Action

Consider using a UGM for binary image denoising:



We have

- Unary potentials  $\phi_j$  for each position.
- Pairwise potentials  $\phi_{ij}$  for neighbours on grid.
- Parameters are trained as CRF (later).

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

## ICM Issue 3: Non-Convexity

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

## Coordinate Sampling

- What about **approximate sampling**?
- In DAGs, ancestral sampling conditions on sampled values of parents,

$$x_j \sim p(x_j \mid x_{\text{pa}(j)}).$$

- In ICM, we approximately decode a UGM by **iteratively maximizing an**  $x_{j_t}$ ,

$$x_j \leftarrow \max_{x_j} p(x_j \mid x_{-j}).$$

- We can approximately sample from a UGM by **iteratively sampling an**  $x_{j_t}$ ,

$$x_j \sim p(x_j \mid x_{-j}),$$

and this **coordinate-wise sampling** algorithm is called **Gibbs sampling**.

# Gibbs Sampling

- **Gibbs sampling** starts with some  $x$  and then repeats:
  - ① Choose a variable  $j$  uniformly at random.
  - ② Update  $x_j$  by sampling it from its conditional,

$$x_j \sim p(x_j \mid x_{-j}).$$

- Analogy: **sampling version of coordinate optimization**:
  - Transformed  $d$ -dimensional sampling into 1-dimensional sampling.
- Gibbs sampling is probably the **most common multi-dimensional sampler**.

## Gibbs Sampling

- For discrete  $x_j$  the conditionals needed for Gibbs sampling have a simple form,

$$p(x_j = c \mid x_{-j}) = \frac{p(x_j = c, x_{-j})}{p(x_{-j})} = \frac{p(x_j = c, x_{-j})}{\sum_{x_j=c'} p(x_j = c', x_{-j})} = \frac{\tilde{p}(x_j = c, x_{-j})}{\sum_{x_j=c'} \tilde{p}(x_j = c', x_{-j})}$$

where we use **unnormalized  $\tilde{p}$**  since  $Z$  is the same in numerator/denominator.

- Note that **this expression is easy to evaluate**: just summing values of 1 variable  $x_j$ .
- And in UGMs it further simplifies to only depend on the Markov blanket,

$$p(x_j \mid x_{-j}) = p(x_j \mid x_{\text{MB}(j)}).$$

- For node  $i$  with 2 neighbours  $j$  and  $k$ , Gibbs sampling step would be:

① Compute  $M_i(x_i) = \phi_i(x_i) \underbrace{\phi_{ij}(x_i, x_j)\phi_{ik}(x_i, x_k)}_{\text{edges in Markov blanket}}$  for all  $x_i$ .

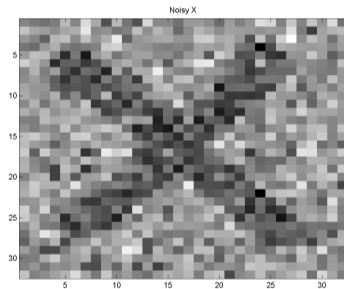
② Sample  $x_i$  proportional to  $M_i(x_i)$ .

## Gibbs Sampling in Action

- Start with some initial value:  $x^0 = [2 \ 2 \ 3 \ 1]$ .
- Select random  $j$  like  $j = 3$ .
- Sample variable  $j$ :  $x^1 = [2 \ 2 \ 1 \ 1]$ .
- Select random  $j$  like  $j = 1$ .
- Sample variable  $j$ :  $x^2 = [3 \ 2 \ 1 \ 1]$ .
- Select random  $j$  like  $j = 2$ .
- Sample variable  $j$ :  $x^3 = [3 \ 2 \ 1 \ 1]$ .
- ...
- Use the samples to form a Monte Carlo estimator.

## Gibbs Sampling in Action: UGMs

Back to image denoising...



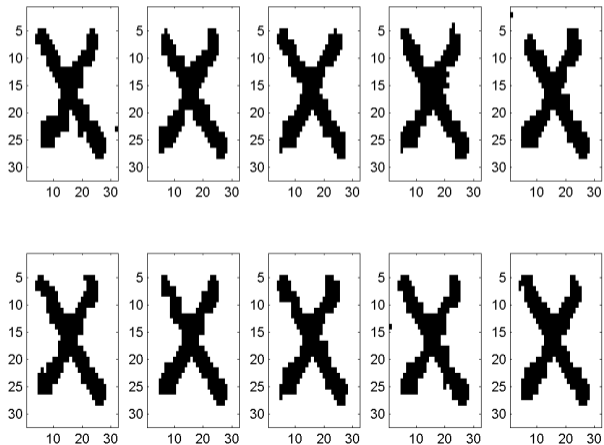
(show videos)



## Gibbs Sampling in Action: UGMs

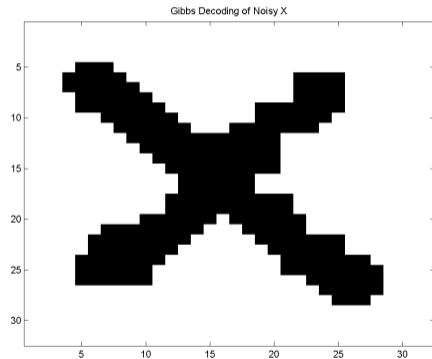
Gibbs samples after every  $100d$  iterations:

Samples from Gibbs sampler



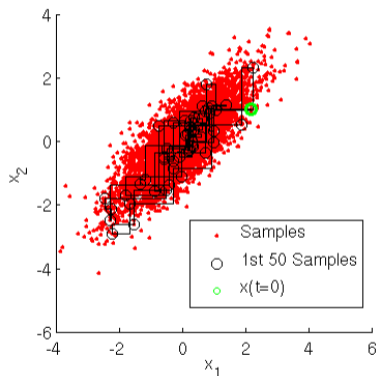
## Gibbs Sampling in Action: UGMs

Estimates of marginals and decoding based on Gibbs sampling:



## Gibbs Sampling in Action: Multivariate Gaussian

- Gibbs sampling works for general distributions.
  - E.g., sampling from multivariate Gaussian by univariate Gaussian sampling.



<https://theclevermachine.wordpress.com/2012/11/05/mcmc-the-gibbs-sampler>

- Video: <https://www.youtube.com/watch?v=AEwY6QXWoUg>

## Gibbs Sampling as a Markov Chain

- Why would Gibbs sampling work?
- Key idea: Gibbs sampling **generates a sample from a homogeneous Markov chain**.
  - If we pick a random  $j$ , we have the same transition distribution at each time.
  - If we cycle through the  $j$ , we consider  $d$ th sample as coming from Markov chain.
- Previously we discussed **stationary distribution** of Markov chain:

$$\pi(s) = \sum_{s'} q(x^t = s \mid x^{t-1} = s') \pi(s'),$$

with transition probabilities  $q$ .

- A sufficient condition for Gibbs sampling to converge to stationary:

$$p(x_j \mid x_{-j}) > 0 \quad \text{for all } j,$$

although weaker conditions exist.

## Markov Chain Monte Carlo (MCMC)

- Stationary distribution  $\pi$  of Gibbs sampling is the target distribution:

$$\pi(x) = p(x),$$

so for large  $k$  a sample  $x^k$  will be distributed according to  $p(x)$ .

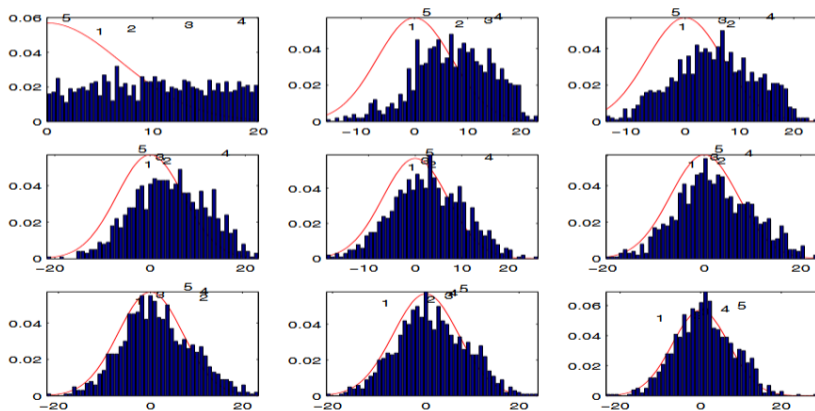
- So we can use it as a Markov Chain Monte Carlo (MCMC) method:
  - Design a Markov chain that has  $\pi(x) = p(x)$ .
  - Use these samples within a Monte Carlo estimator,

$$\mathbb{E}[g(x)] \approx \frac{1}{n} \sum_{t=1}^n g(x^t).$$

- Law of large numbers can be generalized to show this converges as  $n \rightarrow \infty$ .
  - But convergence rate is slower since we're generating dependent samples.

# Markov Chain Monte Carlo

From top left to bottom right: histograms of 1000 independent Markov chains with a normal distribution as target distribution.



## Summary

- **Moralization of DAGs** to do decoding/inference/sampling as a UGM.
- **Markov blanket** is set of nodes that make  $x_j$  independent of all others.
- **Message passing** can be used for inference in UGMs.
  - Belief propagation for trees.
  - Cost might be exponential for unfavourable graphs/ordering.
- **Iterated conditional mode** is coordinate descent for decoding UGMs.
  - Fast but doesn't obtain global optimum in general.
- **Gibbs sampling** is coordinate-wise sampling.
  - Special case of Markov chain Monte Carlo method.
- Next time: reproducing the Spaceballs beaming experiment.

## Conditional Independence and Local Markov Property

- In UGMs, **conditional independence is determined by reachability**.
  - $A \perp B \mid C$  if all paths from  $A$  to  $B$  are blocked by  $C$ .

- The independence assumptions in DAGs were defined by

$$p(x_j \mid x_{1:j-1}) = p(x_j \mid x_{\text{pa}(j)}),$$

that we're independent of previous non-parents given parents.

- In UGMs there is no order and we instead have a **local Markov property**,

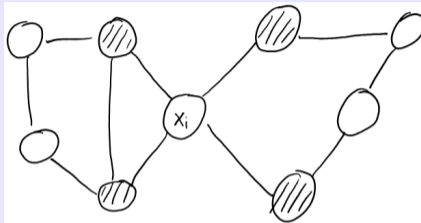
$$p(x_j \mid x_{1:d}) = p(x_j \mid x_{\text{nei}(j)}),$$

that we're independent of all non-neighbours given neighbours in the graph.

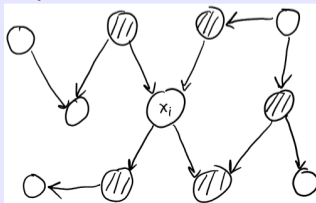


## Markov Blanket

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



- In UGMs the Markov blanket is the neighbours.
- Markov blanket in DAGs is all parents, children, and **co-parents**:



## Decomposable Graphical Models

- Probabilities whose conditional independences that can be represented as DAGs *and* UGMs are called **decomposable**.
  - Includes chains, trees, and fully-connected graphs.
- These models allow some efficient operations in UGMs by writing them as DAGs:
  - Computing  $p(x)$ .
  - Ancestral sampling.
  - Fitting parameters independently.