# CPSC 540: Machine Learning

## More DAGs, Undirected Graphical Models

Mark Schmidt

University of British Columbia

Winter 2017

# Admin

- Assignment 4:
  - Due March 20.

# Last Two 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,

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

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

# Last Two 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,

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

    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 special cases, today we focus on discrete $x_j$.

# Last Time: Conditional Independence in UGMs

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

# Last Time: Conditional Independence in UGMs

- In UGMs, conditional independence in 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

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

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

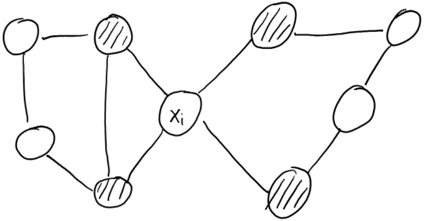
- In UGMs there is no order and we instead have a local Markov property,

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

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

# Markov Blanket

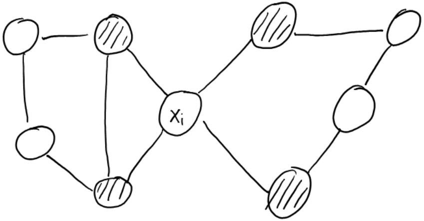
- Markov blanket is the set nodes that make you independent of all other nodes.



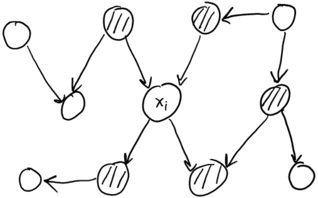

  - In UGMs the Markov blanket is the neighbours.

# Markov Blanket

- 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:



# Outline


1. Complexity of Inference in Graphical Models
2. ICM and Gibbs Sampling
3. Variational Inference

# Inference in Discrete Graphical Models

- 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 $\text{argmax}_x\, p(x)$.
  5. Compute univariate conditional $p(x_j | x_{j'})$.

# Inference in Discrete Graphical Models

- 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 $\text{argmax}_x\, p(x)$.
  5. Compute univariate conditional $p(x_j|x_{j'})$.

- 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.

# Inference in Discrete Graphical Models

- 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 $\text{argmax}_x\, p(x)$.
  5. Compute univariate conditional $p(x_j | x_{j'})$.

- 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.

- 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:

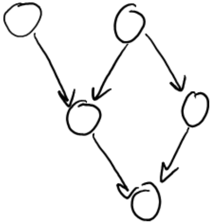
$$p(x) = \prod_{j=1}^{d} p(x_j | x_{\mathsf{pa}(j)}) = \prod_{j=1}^{d} \phi_j(x_j, x_{\mathsf{pa}(j)}).$$

# 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:

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

- Graphically: we drop directions and "marry" parents (moralization).

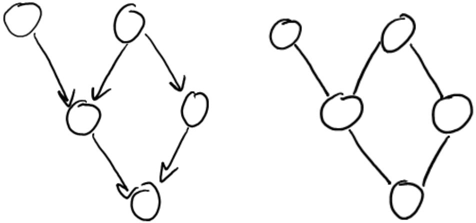


# Moralization: Converting DAGs to UGMs

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

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

- Graphically: we drop directions and "marry" parents (moralization).

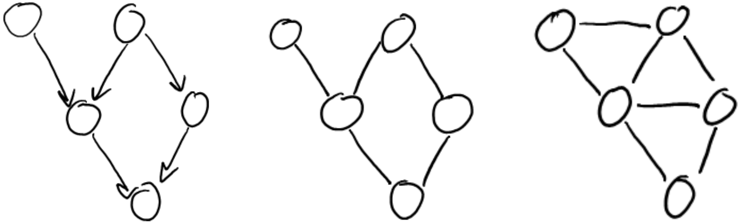


# Moralization: Converting DAGs to UGMs

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

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

- Graphically: we drop directions and "marry" parents (moralization).



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

# Moralization: Converting DAGs to UGMs

- Models that can be represented as DAGs or UGMs are called decomposable.
  - Includes chains, trees, and fully-connected graphs.
- These models allow some efficient operations.
  - E.g., we can write them as DAGs and do ancestral sampling.
  - But this is a restricted model class that we won't talk much about.

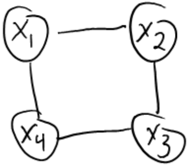
# Moralization: Converting DAGs to UGMs

- Models that can be represented as DAGs or UGMs are called decomposable.
  - Includes chains, trees, and fully-connected graphs.
- These models allow some efficient operations.
  - E.g., we can write them as DAGs and do ancestral sampling.
  - But this is a restricted model class that we won't talk much about.

- We can perform the inference in general UGMs with message passing.
  - The algorithms for general graphs are almost identical....

# Exact Inference in UGMs

- For example, consider a UGM that is a simple 4-node cycle:

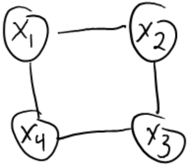



- We can compute $Z$ using

$$Z = \sum_{x_4} \sum_{x_3} \sum_{x_2} \sum_{x_1} \phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4)\phi_{14}(x_1, x_4)$$

# Exact Inference in UGMs

- For example, consider a UGM that is a simple 4-node cycle:

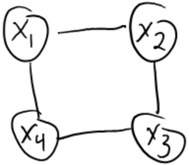



- We can compute $Z$ using

$$Z = \sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1} \phi_{12}(x_1,x_2)\phi_{23}(x_2,x_3)\phi_{34}(x_3,x_4)\phi_{14}(x_1,x_4)$$
$$= \sum_{x_4}\phi_{34}(x_3,x_4)\sum_{x_3}\sum_{x_2}\phi_{23}(x_2,x_3)\sum_{x_1}\phi_{12}(x_1,x_2)\phi_{14}(x_1,x_4)$$

# Exact Inference in UGMs

- For example, consider a UGM that is a simple 4-node cycle:

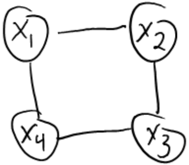



- We can compute $Z$ using

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

# Exact Inference in UGMs

- For example, consider a UGM that is a simple 4-node cycle:




- We can compute $Z$ using

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

# Exact Inference in UGMs

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

# Exact Inference in UGMs

- Message-passing costs depends on graph structure and the order of the sums.
- Consider chain-structured UGM with sums in a different order:

$$Z = \sum_{x_5} \sum_{x_4} \sum_{x_3} \sum_{x_2} \sum_{x_1} \prod_{j=2}^{d} \phi(x_j, x_{j-1})$$

# Exact Inference in UGMs

- Message-passing costs depends on graph structure and the order of the sums.
- Consider chain-structured UGM with sums in a different order:

$$Z = \sum_{x_5}\sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1}\prod_{j=2}^{d}\phi(x_j, x_{j-1})$$

$$= \sum_{x_5}\sum_{x_3}\sum_{x_2}\sum_{x_4}\sum_{x_1}\prod_{j=2}^{d}\phi(x_j, x_{j-1})$$

# Exact Inference in UGMs

- Message-passing costs depends on graph structure and the order of the sums.
- Consider chain-structured UGM with sums in a different order:

$$Z = \sum_{x_5}\sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1}\prod_{j=2}^{d}\phi(x_j, x_{j-1})$$

$$= \sum_{x_5}\sum_{x_3}\sum_{x_2}\sum_{x_4}\sum_{x_1}\prod_{j=2}^{d}\phi(x_j, x_{j-1})$$

$$= \sum_{x_5}\sum_{x_3}\sum_{x_2}\sum_{x_4}\prod_{j=3}^{d}\phi(x_j, x_{j-1})\underbrace{\sum_{x_1}\phi(x_2, x_1)}_{M_2(x_2)}$$

# Exact Inference in UGMs

- Message-passing costs depends on graph structure and the order of the sums.
- Consider chain-structured UGM with sums in a different order:

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

- 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.

# Variable Order and Treewidth

- So cost of message passing depends on
  1. Graph structure.
  2. Variable order.

- Cost of for the best ordering is given by:
  - $O(dk^{\omega+1})$, where $\omega$ is the treewidth of the graph.

# Variable Order and Treewidth

- So cost of message passing depends on
  1. Graph structure.
  2. Variable order.

- Cost of for the best ordering is given by:
  - $O(dk^{\omega+1})$, where $\omega$ is the treewidth of the graph.

- Treewidth $\omega$ is "minimum size of largest clique over all triangulations".
  - For chains, $\omega = 1$ (by going through the chain in order).

# Variable Order and Treewidth

- So cost of message passing depends on
  1. Graph structure.
  2. Variable order.

- Cost of for the best ordering is given by:
  - $O(dk^{\omega+1})$, where $\omega$ is the treewidth of the graph.

- Treewidth $\omega$ is "minimum size of largest clique over all triangulations".
  - For chains, $\omega = 1$ (by going through the chain in order).
  - An $m_1$ by $m_2$ lattice has $\omega = \min\{m_1, m_2\}$.
    - For 28 by 28 MNIST digits it would cost $2^{29}$.

# Variable Order and Treewidth

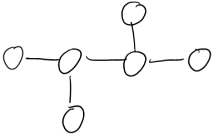
- So cost of message passing depends on
  1. Graph structure.
  2. Variable order.

- Cost of for the best ordering is given by:
  - $O(dk^{\omega+1})$, where $\omega$ is the treewidth of the graph.

- Treewidth $\omega$ is "minimum size of largest clique over all triangulations".
  - For chains, $\omega = 1$ (by going through the chain in order).
  - An $m_1$ by $m_2$ lattice has $\omega = \min\{m_1, m_2\}$.
    - For 28 by 28 MNIST digits it would cost $2^{29}$.
  - In the worst case, $\omega = (d-1)$ so there is no gain.
  - Computing $\omega$ and the optimal ordering is NP-hard.
    - But various heuristic ordering methods exist.

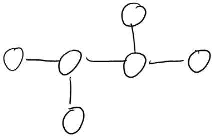
# Variable Order and Treewidth

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

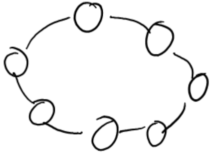


# Variable Order and Treewidth

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

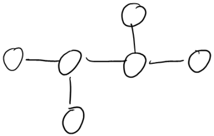


- A big loop has $\omega = 2$, so cost can be $O(dk^3)$.

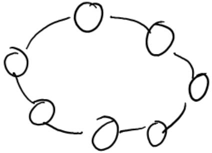


# Variable Order and Treewidth

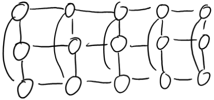
- Trees have $\omega = 1$, so with the right order inference costs $O(dk^2)$.



- A big loop has $\omega = 2$, so cost can be $O(dk^3)$.



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



# Belief Propagation and Junction Trees

- Recall the forward-backward algorithm in Markov chains:
  - We compute the forward messages and the backwards messages.
  - With both types of messages we can compute all univariate marginals.

# Belief Propagation and Junction Trees

- Recall the forward-backward algorithm in Markov chains:
  - We compute the forward messages and the backwards messages.
  - With both types of messages we can compute all univariate marginals.

- Belief propagation is generalization to trees:
  - We start at an arbitrary "root", and pass messages away from it.
  - We also start from the leaves, pass messages towards root.

# Belief Propagation and Junction Trees

- Recall the forward-backward algorithm in Markov chains:
  - We compute the forward messages and the backwards messages.
  - With both types of messages we can compute all univariate marginals.
- Belief propagation is generalization to trees:
  - We start at an arbitrary "root", and pass messages away from it.
  - We also start from the leaves, pass messages towards root.
- Generalization to general graphs is the junction tree method.
- Unfortunately, low tree width models are very restricted.
  - This has motivated a ton of work on approximate inference...

# Outline

1 Complexity of Inference in Graphical Models

2 ICM and Gibbs Sampling

3 Variational Inference

# Iterated Conditional Mode (ICM)

- The iterated conditional mode (ICM) algorithm for approximate decoding:
  - On each iteration $t$, choose a variable $j_t$.
  - Optimize $x_{j_t}$ with the other variables held fixed.

# Iterated Conditional Mode (ICM)

- The iterated conditional mode (ICM) algorithm for approximate decoding:
  - On each iteration $t$, choose a variable $j_t$.
  - Optimize $x_{j_t}$ with the other variables held fixed.

- A special case of coordinate optimization.
- Iterations correspond to finding mode of conditional $p(x_j | x_{-j})$.

# Iterated Conditional Mode (ICM)

- The iterated conditional mode (ICM) algorithm for approximate decoding:
  - On each iteration $t$, choose a variable $j_t$.
  - Optimize $x_{j_t}$ with the other variables held fixed.

- A special case of coordinate optimization.
- Iterations correspond to finding mode of conditional $p(x_j|x_{-j})$.

- 3 main issues:
  1. How can you optimize $p(x)$ if evaluating it 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?

# ICM Issue 1: Intractable Objective

- How can you optimize $p(x)$ if evaluating it is NP-hard?

- Note that it's easy to evaluate unnormalized probability.

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

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

- And for decoding we only need unnormalized probabilities,

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

# ICM Issue 1: Intractable Objective

- How can you optimize $p(x)$ if evaluating it is NP-hard?
- Note that it's easy to evaluate unnormalized probability.

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

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

- And for decoding we only need unnormalized probabilities,

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

- To update $x_j$ we actually only need consider $\phi_c$ involving $x_j$
  - We only care about $x_{-j}$ in the Markov blanket (neighbours in the graph).

# ICM Issue 2: Efficiency

- Is coordinate optimization efficient for this problem?

# 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

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

# 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 our problems where coordinate optimization is efficient.

# ICM Issue 3: Non-Convexity

- Does it find the global optimum?
- Negative log-probability is usually non-convex, so doesn't find global optimum.

# ICM Issue 3: Non-Convexity

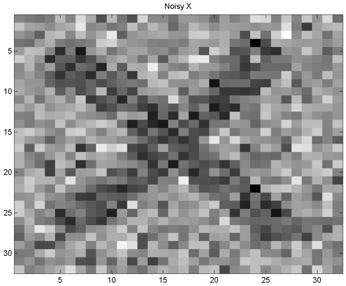
- Does it find the global optimum?
- Negative log-probability is usually non-convex, so doesn't find global optimum.
- There exist many globalization methods that can improve its performance:
    - Restarting with random initializations.

# ICM Issue 3: Non-Convexity

- Does it find the global optimum?
- Negative log-probability is usually non-convex, so doesn't find global optimum.
- There exist many globalization methods that can improve its performance:
  - Restarting with random initializations.
  - Simulated annealing, genetic algorithms, ant colony optimization, etc.
  - See the book/class of Holger Hoos on stochastic local search methods.

# ICM in Action

Consider using a UGM for 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 image (show video).

# Coordinate Sampling

- What about approximate sampling?

# Coordinate Sampling

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

$$x_j \sim p(x_j | x_{\mathsf{pa}(j)}).$$

# Coordinate Sampling

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

$$x_j \sim p(x_j | x_{\mathsf{pa}(j)}).$$

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

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

# Coordinate Sampling

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

$$x_j \sim p(x_j | x_{\mathsf{pa}(j)}).$$

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

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

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

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

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,

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

# 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,

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

- 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 UGMs these conditionals needed for Gibbs sampling have a simple form,

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

because the $Z$ is the same in the numerator and deonimator terms.

# Gibbs Sampling

- For UGMs these conditionals needed for Gibbs sampling have a simple form,

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

  because the $Z$ is the same in the numerator and deonimator terms.

- And UGMs it further simplifies due to the local Markov property,

$$p(x_j|x_{-j}) = p(x_j|x_{\mathsf{MB}(j)}).$$

- Thus these iterations are very cheap:
  - We're just sampling a discrete variable given its Markov blanket.

# Gibbs Sampling in Action

- Start with some initial value: $x^0 = \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix}$.

# Gibbs Sampling in Action

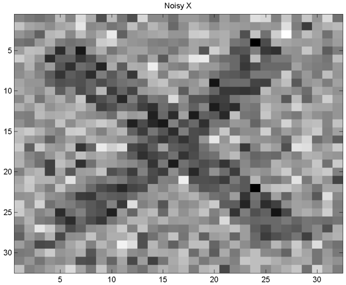
- Start with some initial value: $x^0 = \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix}$.
- Select random $j$ like $j = 3$.
- Sample variable $j$: $x^2 = \begin{bmatrix} 2 & 2 & {\color{red}1} & 1 \end{bmatrix}$.

# Gibbs Sampling in Action

- Start with some initial value: $x^0 = \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix}$.
- Select random $j$ like $j = 3$.
- Sample variable $j$: $x^2 = \begin{bmatrix} 2 & 2 & \color{red}{1} & 1 \end{bmatrix}$.
- Select random $j$ like $j = 1$.
- Sample variable $j$: $x^3 = \begin{bmatrix} \color{red}{3} & 2 & 1 & 1 \end{bmatrix}$.
- Select random $j$ like $j = 2$.
- Sample variable $j$: $x^4 = \begin{bmatrix} 3 & \color{red}{2} & 1 & 1 \end{bmatrix}$.
- ...
- Use the samples to form Monte Carlo estimators.

# Gibbs Sampling in Action: UGMs

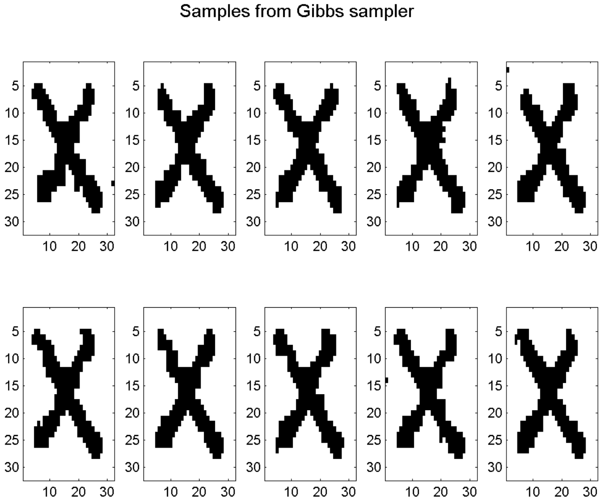
Back to image denoising...




(show videos)

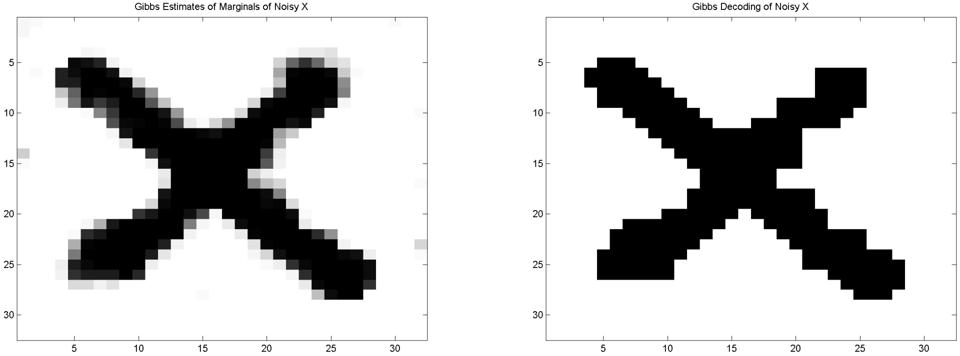
# Gibbs Sampling in Action: UGMs

Gibbs samples after every $100d$ iterations:



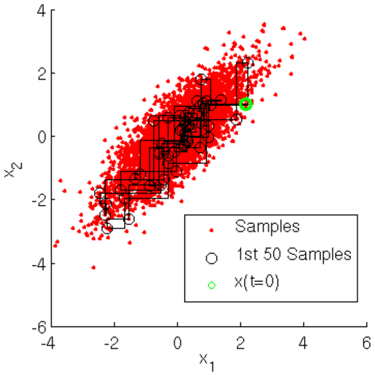
# Gibbs Sampling in Action: UGMs

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

# Gibbs Sampling and Markov Chains

- Why would Gibbs sampling work?

# Gibbs Sampling and Markov Chains

- Why would Gibbs sampling work?

- The samples follow a homogeneous Markov chain.

# Gibbs Sampling and Markov Chains

- Why would Gibbs sampling work?

- The samples follow a homogeneous Markov chain.
- Under weak conditions, homogenous chains converge to an invariant distribution,

$$\pi(s) = \sum_{s'} p(x^t = s | x^{t-1} = s')\pi(s').$$

  - $p(x_j | x_{-j}) > 0$ is sufficient for Gibbs sampling.
  - A weaker condition is "irreducible and aperiodic".

# Gibbs Sampling and Markov Chains

- Why would Gibbs sampling work?
- The samples follow a homogeneous Markov chain.
- Under weak conditions, homogenous chains converge to an invariant distribution,

$$\pi(s) = \sum_{s'} p(x^t = s | x^{t-1} = s')\pi(s').$$

  - $p(x_j | x_{-j}) > 0$ is sufficient for Gibbs sampling.
  - A weaker condition is "irreducible and aperiodic".
- Invariant distribution $\pi$ of Gibbs sampling is the original distribution $p$.
  - If we stop it after a really long time, the final Gibbs sample will come from $p(x)$.
- A special case of Markov chain Monte Carlo (MCMC) methods.

# Markov Chain Monte Carlo (MCMC)

- Markov chain Monte Carlo (MCMC): given target $p$, design transitions such that

$$\frac{1}{n}\sum_{t=1}^{n} f(x^t) \to \sum_{x} f(x)p(x) \quad \text{and/or} \quad x^n \sim p,$$

  as $n \to \infty$.
- We are generating dependent samples whose average converges to expectation.

# Markov Chain Monte Carlo (MCMC)

- Markov chain Monte Carlo (MCMC): given target $p$, design transitions such that

$$\frac{1}{n}\sum_{t=1}^{n} f(x^t) \to \sum_x f(x)p(x) \quad \text{and/or} \quad x^n \sim p,$$

  as $n \to \infty$.
- We are generating dependent samples whose average converges to expectation.
- There are many transitions that will yield target as invariant distribution.
  - Typically easy to design sampler, but hard to characterize rate of convergence.

# Markov Chain Monte Carlo (MCMC)

- Markov chain Monte Carlo (MCMC): given target $p$, design transitions such that

$$\frac{1}{n}\sum_{t=1}^{n} f(x^t) \rightarrow \sum_{x} f(x)p(x) \quad \text{and/or} \quad x^n \sim p,$$

  as $n \rightarrow \infty$.
- We are generating dependent samples whose average converges to expectation.
- There are many transitions that will yield target as invariant distribution.
  - Typically easy to design sampler, but hard to characterize rate of convergence.
- Gibbs sampling satisfies the above under very weak conditions.

# Markov Chain Monte Carlo (MCMC)

- Markov chain Monte Carlo (MCMC): given target $p$, design transitions such that

$$\frac{1}{n}\sum_{t=1}^{n} f(x^t) \to \sum_{x} f(x)p(x) \quad \text{and/or} \quad x^n \sim p,$$

  as $n \to \infty$.
- We are generating dependent samples whose average converges to expectation.
- There are many transitions that will yield target as invariant distribution.
  - Typically easy to design sampler, but hard to characterize rate of convergence.
- Gibbs sampling satisfies the above under very weak conditions.
- Typically, we don't take all samples:
  - Burn in: throw away the initial samples when we haven't converged to stationary.
  - Thinning: only keep every $k$ samples, since they will be highly correlated.

# Markov Chain Monte Carlo (MCMC)

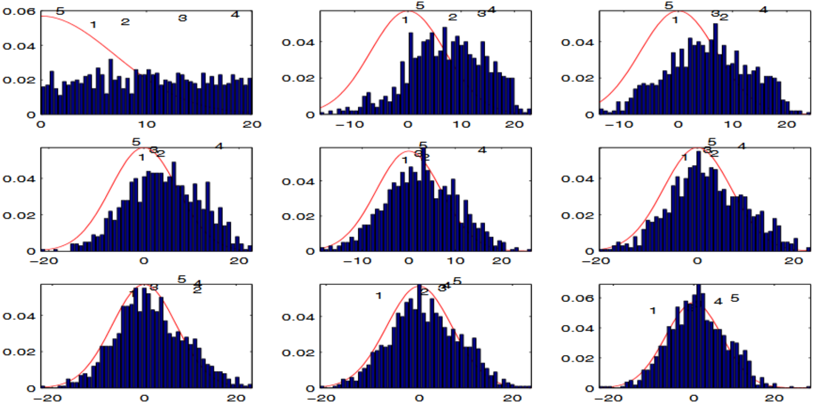
- Markov chain Monte Carlo (MCMC): given target $p$, design transitions such that

$$\frac{1}{n}\sum_{t=1}^{n} f(x^t) \to \sum_{x} f(x)p(x) \quad \text{and/or} \quad x^n \sim p,$$

  as $n \to \infty$.
- We are generating dependent samples whose average converges to expectation.
- There are many transitions that will yield target as invariant distribution.
  - Typically easy to design sampler, but hard to characterize rate of convergence.
- Gibbs sampling satisfies the above under very weak conditions.
- Typically, we don't take all samples:
  - Burn in: throw away the initial samples when we haven't converged to stationary.
  - Thinning: only keep every $k$ samples, since they will be highly correlated.
- It can very hard to diagnose if we reached invariant distribution.
  - Recent work showed that this is P-space hard (*not* polynomial-time).

# Markov Chain Monte Carlo

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



http://www.cs.ubc.ca/~arnaud/stat535/slides10.pdf

# Outline


1. Complexity of Inference in Graphical Models
2. ICM and Gibbs Sampling
3. Variational Inference

# Monte Carlo vs. Variational Inference

Two main strategies for approximate inference:

1. Monte Carlo methods:
   - Approximate $p$ with empirical distribution over samples,
     $$p(x) \approx \frac{1}{n}\sum_{i=1}^{n}\mathcal{I}[x^i = x].$$
   - Turns inference into sampling.

# Monte Carlo vs. Variational Inference

Two main strategies for approximate inference:

1. Monte Carlo methods:
   - Approximate $p$ with empirical distribution over samples,
   $$p(x) \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{I}[x^i = x].$$
   - Turns inference into sampling.
2. Variational methods:
   - Approximate $p$ with "closest" distribution $q$ from a tractable family,
   $$p(x) \approx q(x).$$

# Monte Carlo vs. Variational Inference

Two main strategies for approximate inference:

1. Monte Carlo methods:
    - Approximate $p$ with empirical distribution over samples,

$$p(x) \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{I}[x^i = x].$$

    - Turns inference into sampling.
2. Variational methods:
    - Approximate $p$ with "closest" distribution $q$ from a tractable family,

$$p(x) \approx q(x).$$

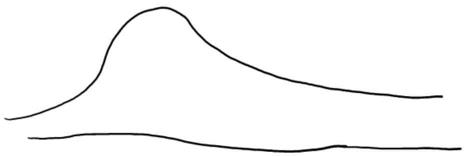
    - E.g., Gaussian, independent Bernoulli, or tree UGM.

(or mixtures of these simple distributions)

    - Turns inference into optimization.

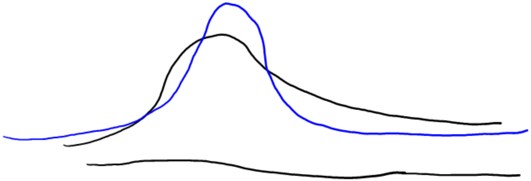
# Variational Inference Illustration

- Approximate non-Gaussian $p$ by a Gaussian $q$:



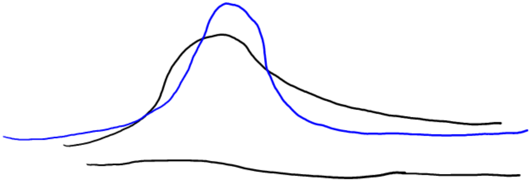
# Variational Inference Illustration

- Approximate non-Gaussian $p$ by a Gaussian $q$:

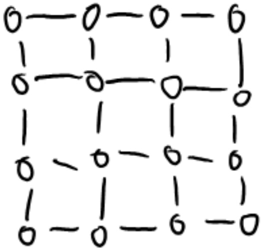


# Variational Inference Illustration

- Approximate non-Gaussian $p$ by a Gaussian $q$:

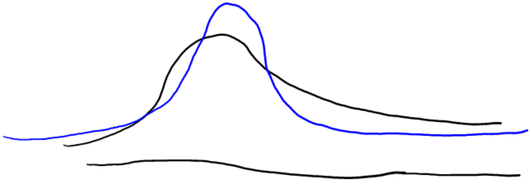


- Approximate loopy UGM by independent distribution

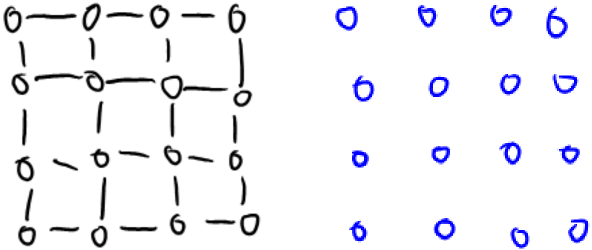


# Variational Inference Illustration

- Approximate non-Gaussian $p$ by a Gaussian $q$:

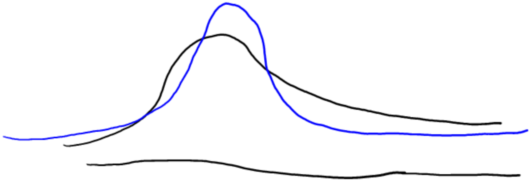


- Approximate loopy UGM by independent distribution

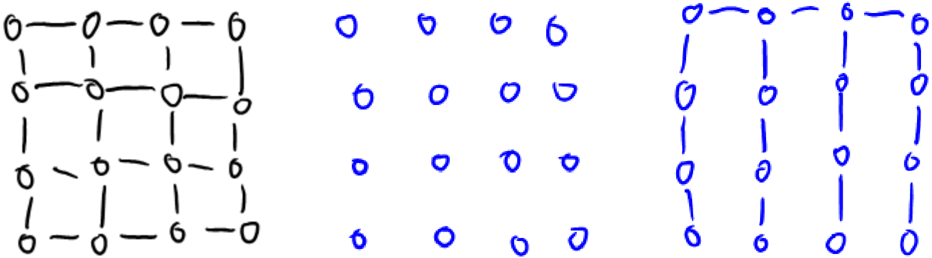


# Variational Inference Illustration

- Approximate non-Gaussian $p$ by a Gaussian $q$:



- Approximate loopy UGM by independent distribution or tree-structed UGM:



# Minimizing Reverse KL) Divergence

- Most common variational method:
  - Minimize (reverse) Kullback-Leibler (KL) divergence between $q$ and $p$,

$$\mathsf{KL}(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}.$$

  - KL divergence is a common measure of similarity between distributions.
    - Also called information gain: "information lost when $q$ is approximated by $p$?".

# Minimizing Reverse KL) Divergence

- Most common variational method:
  - Minimize (reverse) Kullback-Leibler (KL) divergence between $q$ and $p$,

$$\mathsf{KL}(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}.$$

  - KL divergence is a common measure of similarity between distributions.
    - Also called information gain: "information lost when $q$ is approximated by $p$?".
  - KL would be more natural, but reverse KL only needs unnormalized distribution $\tilde{p}$,

$$\mathsf{KL}(q||p) = \sum_x q(x) \log q(x) - \sum_x q(x) \log p(x)$$

# Minimizing Reverse KL) Divergence

- Most common variational method:
  - Minimize (reverse) Kullback-Leibler (KL) divergence between $q$ and $p$,

$$\text{KL}(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}.$$

  - KL divergence is a common measure of similarity between distributions.
    - Also called information gain: "information lost when $q$ is approximated by $p$?".
  - KL would be more natural, but reverse KL only needs unnormalized distribution $\tilde{p}$,

$$\begin{aligned}\text{KL}(q||p) &= \sum_x q(x) \log q(x) - \sum_x q(x) \log p(x) \\ &= \sum_x q(x) \log q(x) - \sum_x q(x) \log \tilde{p}(x) + \sum_x q(x) \log(Z)\end{aligned}$$

# Minimizing Reverse KL) Divergence

- Most common variational method:
  - Minimize (reverse) Kullback-Leibler (KL) divergence between $q$ and $p$,

$$\mathsf{KL}(q||p) = \sum_x q(x) \log \frac{q(x)}{p(x)}.$$

  - KL divergence is a common measure of similarity between distributions.
    - Also called information gain: "information lost when $q$ is approximated by $p$?".

  - KL would be more natural, but reverse KL only needs unnormalized distribution $\tilde{p}$,

$$\begin{aligned}\mathsf{KL}(q||p) &= \sum_x q(x) \log q(x) - \sum_x q(x) \log p(x) \\ &= \sum_x q(x) \log q(x) - \sum_x q(x) \log \tilde{p}(x) + \sum_x q(x) \log(Z) \\ &= \sum_x q(x) \log \frac{q(x)}{\tilde{p}(x)} + \underbrace{\log(Z)}_{\text{const. in } q},\end{aligned}$$

which since KL is non-negative gives a lower bound on $\log(Z)$.

# Mean Field Variational Approximation

- Consider minimizing reverse KL with independent $q$,

$$q(x) = \prod_{j=1}^{d} q_j(x_j).$$

# Mean Field Variational Approximation

- Consider minimizing reverse KL with independent $q$,

$$q(x) = \prod_{j=1}^{d} q_j(x_j).$$

- If we fix $q_{-j}$ and optimize the functional $q_j$ we obtain (not obvious)

$$\log q_j(x_j) = \mathbb{E}_{q_{-j}}[\log \tilde{p}(x)] + \text{constant},$$

so we can update $q_j$ function-wise by setting them to mean in log-space.

# Mean Field Variational Approximation

- Consider minimizing reverse KL with independent $q$,

$$q(x) = \prod_{j=1}^{d} q_j(x_j).$$

- If we fix $q_{-j}$ and optimize the functional $q_j$ we obtain (not obvious)

$$\log q_j(x_j) = \mathbb{E}_{q_{-j}}[\log \tilde{p}(x)] + \text{constant},$$

  so we can update $q_j$ function-wise by setting them to mean in log-space.

- This is called the mean field approximation.

- Once you've fit $q$, you use the independent distribution instead of $p$.

# Summary

- Markov blanket is set of nodes that make $x_j$ independent of all others.

# Summary

- Markov blanket is set of nodes that make $x_j$ independent of all others.
- Moralization of DAGs to do decoding/inference/sampling as a UGM.

# Summary

- Markov blanket is set of nodes that make $x_j$ independent of all others.
- Moralization of DAGs to do decoding/inference/sampling as a UGM.
- Iterated conditional mode is coordinate descent for decoding UGMs.

# Summary

- Markov blanket is set of nodes that make $x_j$ independent of all others.
- Moralization of DAGs to do decoding/inference/sampling as a UGM.
- Iterated conditional mode is coordinate descent for decoding UGMs.
- Gibbs sampling is coordinate-wise sampling.
    - Special case of Markov chain Monte Carlo method.

# Summary

- Markov blanket is set of nodes that make $x_j$ independent of all others.
- Moralization of DAGs to do decoding/inference/sampling as a UGM.
- Iterated conditional mode is coordinate descent for decoding UGMs.
- Gibbs sampling is coordinate-wise sampling.
    - Special case of Markov chain Monte Carlo method.
- Variational methods approximate $p$ with a simpler distribution $q$.
    - Mean field approximation minimizes KL divergence with independent $q$.

Next time: deep graphical models and finally being able to model digits.