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,
    \[
    p(x) = \prod_{j=1}^{d} p(x_j \mid x_{pa(j)}),
    \]
    and are also known as “Bayesian networks” and “belief networks”.

  - **UGMs** represent probability as product of non-negative potentials \( \phi_c \),
    \[
    p(x) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c), \quad \text{with} \quad Z = \sum_x \prod_{c \in C} \phi_c(x_c),
    \]
    and are also known as “Markov random fields” and ”Markov networks”.

- We discussed inference tasks (for both by converting to UGMs) in discrete \( x_j \).
  - Cost of message passing is exponential in treewidth of graph.
  - Motivates considering approximate inference methods today.
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,

  ![Graph](image)

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

  ![Graph](image)

  Graphically, we “erase the black nodes and their edges”.

- Notice that inference in the **conditional UGM** may be much 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 many approximate inference methods.
Approximate inference methods often use conditional \( p(x_j \mid x_{-j}) \),

where \( x_{-j} \) 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”.
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.

- Markov blanket in DAGs is all parents, children, and **co-parents**.
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** $p(x_j \mid x_{\neg j})$,
  $$x_j^{k+1} \leftarrow \max_c p(x_j = c \mid x_{\neg j}^k),$$

- 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 \( p(x) \) coordinate-wise if evaluating it is NP-hard?

- Let’s define the unnormalized probability \( \tilde{p} \) as
  \[
  \tilde{p}(x) = \prod_{c \in C} \phi_c(x_c).
  \]

- 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,
  \[
  \arg\max_x p(x) \equiv \arg\max_x \frac{\tilde{p}(x)}{Z} \equiv \arg\max_x \tilde{p}(x),
  \]
  so we can decode based on \( \tilde{p} \) without knowing \( Z \).
ICM Issue 2: Efficiency

- Is coordinate optimization efficient for this problem?

- Consider a pairwise UGM,

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

or

\[ \log \tilde{p}(x) = \sum_{j=1}^{d} \log \phi_j(x_j) + \sum_{(i,j) \in E} \log \phi_{ij}(x_i, x_j) , \]

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 problems where coordinate optimization is \( n \)-times faster.
Consider a pairwise UGM:

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

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

1. Compute $M_i(x_i) = \phi_i(x_i) \phi_{ij}(x_i, x_j) \phi_{ik}(x_i, x_k)$ for all $x_i$.

2. Set $x_i$ to the largest value of $M_i(x_i)$. 

edges in Markov blanket
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.
Outline

1. Iterated Conditional Mode
2. Gibbs Sampling
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 \( x_{jt} \),
  \[ x_j \leftarrow \max_{x_j} p(x_j \mid x_{-j}). \]

- We can approximately sample from a UGM by iteratively sampling an \( x_{jt} \),
  \[ x_j \sim p(x_j \mid x_{-j}), \]
  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,

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

- 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 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

- 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 over values of $x_j$.

- And in UGMs it further simplifies to only depend on the Markov blanket,

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

since the other terms cancel in the numerator/denominator.
Gibbs Sampling in Action: UGMs

- For node $i$ with 2 neighbours $j$ and $k$, Gibbs sampling step would be:
  1. Compute $M_i(x_i) = \phi_i(x_i) \phi_{ij}(x_i, x_j) \phi_{ik}(x_i, x_k)$ for all $x_i$.
  - edges in Markov blanket
  2. Sample $x_i$ proportional to $M_i(x_i)$.

(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.

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.

The “Gibbs sampling Markov chain” for sampling from a 4-variable binary UGM:
- The states are the possible configurations of the four variables:
  - \( s = [0, 0, 0, 0] \), \( s = [0, 0, 0, 1] \), \( s = [0, 0, 1, 0] \), etc.
- The initial probability \( q \) is set to 1 for the initial state, and 0 for the others:
  - If you start at \( s = [1, 1, 0, 1] \), then \( q(x^1 = [1, 1, 0, 1]) = 1 \) and \( q(x^1 = [0, 0, 0, 0]) = 0 \).
- The transition probabilities \( q \) are based on variable we choose and UGM:
  - If we are at \( s = [1, 1, 0, 1] \) and choose coordinate randomly we have:
    \[
    q(x_t^{t+1} = [0, 0, 1, 1] \mid x_t = [1, 1, 0, 1]) = 0 \quad \text{(Gibbs only updates on variable)}
    \]
    \[
    q(x_t^{t+1} = [1, 0, 0, 1] \mid x_t = [1, 1, 0, 1]) = \frac{1}{d} \quad \text{uniform from UGM}
    \]
  - Not homogeneous if cycling through the \( j \), but homogeneous over every \( d \) samples.
Gibbs Sampling as a Markov Chain

Why would Gibbs sampling work?

- Key idea: Gibbs sampling generates a sample from a homogeneous 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 $q$ (of the Gibbs sampling Markov chain).

A sufficient condition for Gibbs Markov chain to have unique stationary:

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

- Allows Gibbs sampling to be used in Markov Chain Monte Carlo (MCMC):
  - Design a Markov chain that has $\pi(x) = p(x)$.
  - Use these samples within a Monte Carlo estimator,

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

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

MCMC sampling from a Gaussian:

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

- Basic idea of Markov Chain Monte Carlo (MCMC) method:
  - Design a Markov chain that has \( \pi(x) = p(x) \).
  - Use these samples within a Monte Carlo estimator,

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

- In practice, we often don’t take all samples in our Monte Carlo estimate:
  - **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.
MCMC Implementation Issues

- Two common ways that MCMC is applied:
  1. Sample from a huge number of Markov chains for a long time, use final states.
     - Great for parallelization.
     - No need for thinning, if chains are independently initialized.
     - Need to worry about burn in.
  2. Sample from one Markov chain for a really long time, use states across time.
     - Less worry about burn in.
     - Need to worry about thinning.

- It can very hard to diagnose if we reached stationary distribution.
  - Recent work showed that this is P-space hard (not polynomial-time even if P=NP).
  - Various heuristics exist.
Summary

- **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.

- **Gibbs sampling** is coordinate-wise sampling.
  - Special case of Markov chain Monte Carlo method.

- Next time: reproducing the Spaceballs beaming experiment.
Conditioning in UGMs

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

\[
p(x_1, x_4|x_2, x_3) = \frac{p(x_1, x_2, x_3, x_4)}{p(x_2, x_3)}
\]

\[
= \frac{\frac{1}{Z} \phi_1(x_1) \phi_2(x_2) \phi_3(x_3) \phi_4(x_4) \phi_1(x_1, x_2) \phi_2(x_2, x_3) \phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \frac{1}{Z} \phi_1(x_1') \phi_2(x_2) \phi_3(x_3) \phi_4(x_4) \phi_1(x_1', x_2) \phi_2(x_2, x_3) \phi_3(x_3, x_4')}
\]

\[
= \frac{\frac{1}{Z} \phi_1(x_1) \phi_2(x_2) \phi_3(x_3) \phi_2(x_2, x_3) \sum_{x_1', x_4'} \phi_1(x_1') \phi_4(x_4') \phi_1(x_1', x_2) \phi_3(x_3, x_4')}{\sum_{x_1', x_4'} \phi_1(x_1') \phi_4(x_4') \phi_1(x_1', x_2) \phi_3(x_3, x_4')}
\]

\[
= \frac{\phi_1(x_1) \phi_4(x_4) \phi_1(x_1, x_2) \phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \phi_1(x_1') \phi_4(x_4') \phi_1(x_1', x_2) \phi_3(x_3, x_4')}
\]

\[
= \frac{\phi_1(x_1) \phi_4'(x_4)}{\sum_{x_1', x_4'} \phi_1(x_1') \phi_4'(x_4')}
\]