CPSC 440: Advanced Machine Learning

More Approximate Inference

Mark Schmidt

University of British Columbia

Winter 2021

Last Time: ICM for Approximate Decoding

- We discussed iterate conditional mode (ICM) for decoding.
- Start with a guess for x, and at iteration t:
  - Optimize the variable x_j for some variable j, with others held fixed.
- Fast but not guaranteed to find local optimum ("approximate" decoding).
- Works with "unnormalized" probability $\tilde{p}(x)$.
- Can be implemented efficiently: update only depends on Markov blanket.
  - Markov blanket is the nodes that make you conditionally independent of all others.

(Show video)

Coordinate Sampling

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

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

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

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

- 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 ICM:
  - Transforms 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: $x^0 = \begin{bmatrix} 2 & 2 & 3 & 1 \end{bmatrix}$.
- Select random j like $j = 3$.
- Sample variable j: $x^1 = \begin{bmatrix} 2 & 2 & 1 & 1 \end{bmatrix}$.
- Select random j like $j = 1$.
- Sample variable j: $x^2 = \begin{bmatrix} 3 & 2 & 1 & 1 \end{bmatrix}$.
- Select random j like $j = 2$.
- Sample variable j: $x^3 = \begin{bmatrix} 3 & 2 & 1 & 1 \end{bmatrix}$.
- ...
- 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.

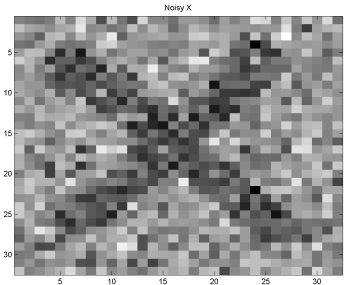
- And in UGMs it further simplifies to only depend on the Markov blanket,

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

since the other terms cancel in the numerator/denominator.

Gibbs Sampling in Action: UGMs

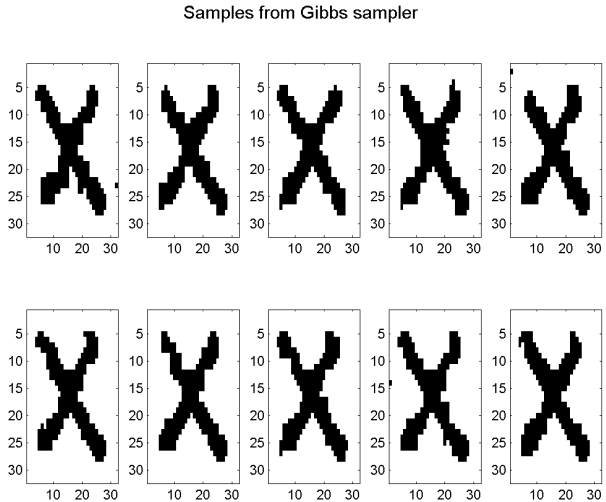
- Each ICM update would:
  1. Set $M_j(x_j = s)$ to product of terms in $\tilde{p}(x)$ involving x_j, with x_j set to s.
  2. Sample x_j proportional to $M_j(x_j)$.



(show videos)

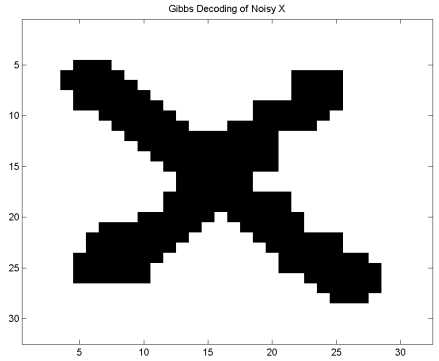
Gibbs Sampling in Action: UGMs

Gibbs samples after every $100d$ iterations:



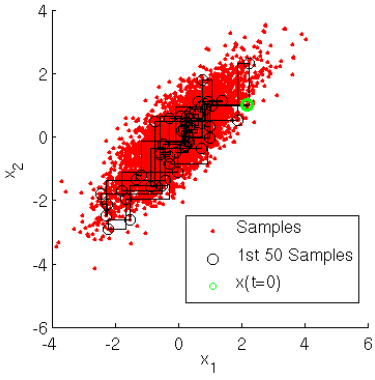
Gibbs Sampling in Action: UGMs

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

- 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:

$$q(x^{t+1} = [0\ 0\ 1\ 1] \mid x^t = [1\ 1\ 0\ 1]) = 0 \quad \text{(Gibbs only updates on variable)}$$

$$q(x^{t+1} = [1\ 0\ 0\ 1] \mid x^t = [1\ 1\ 0\ 1]) = \underbrace{\frac{1}{d}}_{\text{uniform}} \underbrace{p(x_2 = 0 \mid x_1 = 1, x_3 = 0, x_4 = 1)}_{\text{from UGM}}.$$

    - Not homogeneous if cycling, but homogeneous if add "last variable" to state.

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:

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

Markov Chain Monte Carlo (MCMC)

- Stationary distribution π 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,

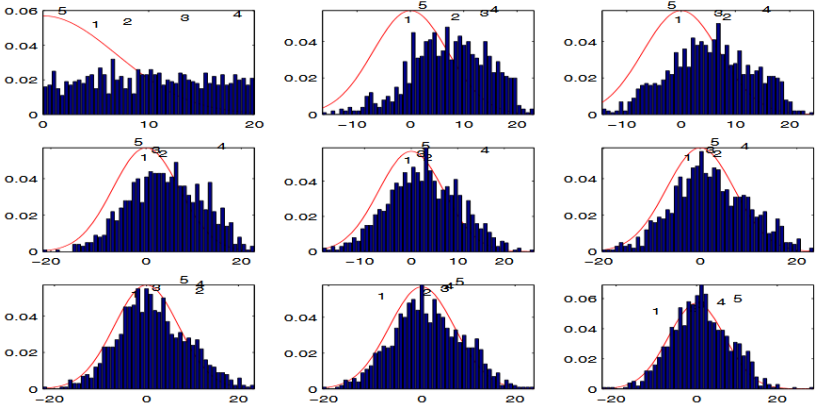
$$\mathbb{E}[g(x)] \approx \frac{1}{n} \sum_{t=1}^{n} g(x^i).$$

- Law of large numbers can be generalized to show this converges as $n \to \infty$.
  - "Ergodic theroem".
  - But convergence 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.



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

MCMC Implementation Issues

- 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.

- 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 have reached stationary distribution.
  - Recent work showed that this is P-space hard (*not* polynomial-time even if P=NP).
  - Various heuristics exist.

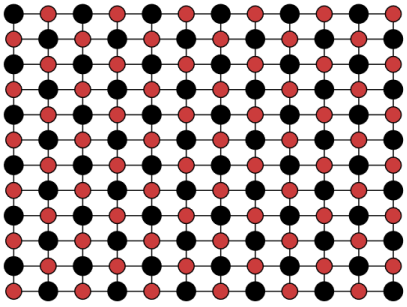
Outline

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"):

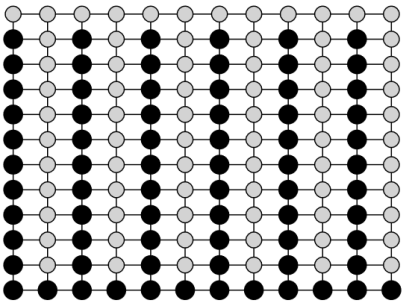


- Given black nodes, conditional UGM on red nodes is a disconnected graph.
  - "I can optimally update the red nodes given the black nodes" (and vice versa).
    - You update $d/2$ nodes at once for cost of this is $O(dk)$, and easy to parallelize.

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

Block-Structured Approximate Inference

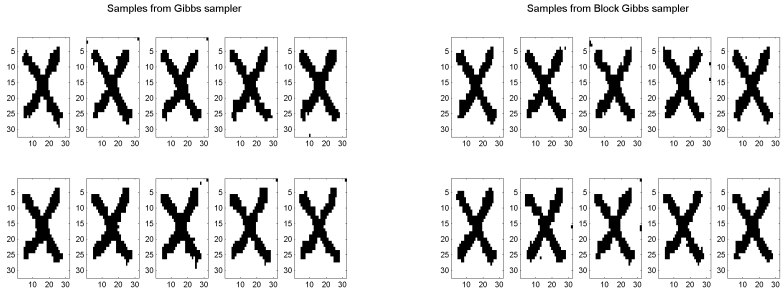
- We could also consider general forest-structured blocks:



- 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

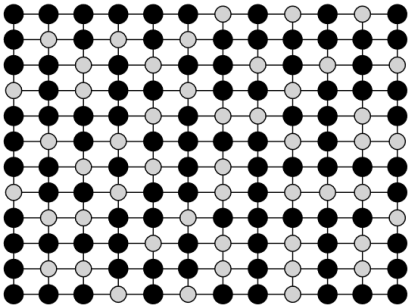
- Gibbs vs. tree-structured block-Gibbs samples:



- 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 UGMs 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"

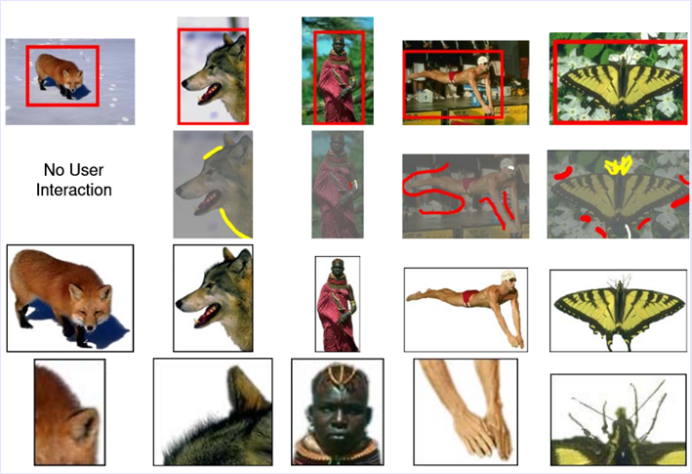
Figure 1: **Three examples of GrabCut.** The user drags a rectangle loosely around an object. The object is then extracted automatically.

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:

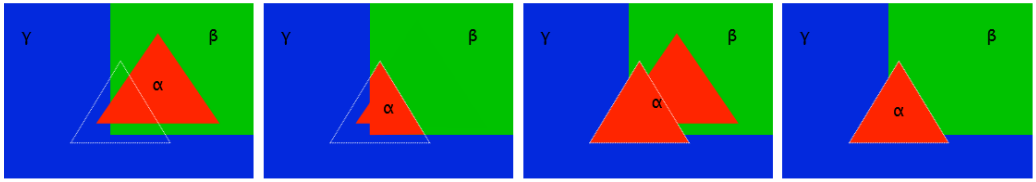


Figure 1: From left to right: Initial labeling, labeling after $\alpha\beta$-swap, labeling after α-expansion, labeling after α-expansion β-shrink. The optimal labeling of the α pixels is outlined by a white triangle, and is achieved from the initial labeling by one α-expansion β-shrink move.

- 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

Summary

- Gibbs sampling is coordinate-wise sampling.
  - Special case of Markov chain Monte Carlo (MCMC) method.
- Block approximate inference works better than single-variable methods.
  - Blocks could be defined by trees or to implement graph cuts.
- Next time: learning in UGMs.