CPSC 540: Machine Learning

More Approximate Inference

Mark Schmidt

University of British Columbia

Winter 2019

Last Time: Approximate Inference

- We've discussed approximate inference in two settings:
  1. Inference in graphical models (sum over x values).

$$E[f(x \mid w)] = \sum_x f(x)p(x \mid w)dx.$$

  2. Inference in Bayesian models (integrate over posterior values).

$$E[f(\theta)] = \int_\theta f(\theta)p(\theta \mid x)d\theta.$$

- Our previous approach was Monte Carlo methods.
  - Gibbs sampling (special case of MCMC).
- Inverse transform can be used for conjugate models.
- Rejection sampling or importance sampling for non-conjugate.
  - Can be used to model whole distribution, or to model conditionals in Gibbs.

Limitations of Simple Monte Carlo Methods

- The basic ingredients of our previous sampling methods:
    - Inverse CDF, rejection sampling, importance sampling.
    - Sampling in higher-dimensions: ancestral sampling, Gibbs sampling.
- These work well in low dimensions or for posteriors with analytic properties.
- But we want to solve high-dimensional integration problems in other settings:
    - Deep belief networks and Boltzmann machines.
    - Bayesian graphical models and Bayesian neural networks.
    - Hierarchical Bayesian models.
- Our previous methods tend not to work in complex situations:
    - Inverse CDF may not be available.
    - Conditionals needed for ancestral/Gibbs sampling may be hard to compute.
    - Rejection sampling tends to reject almost all samples.
    - Importance sampling tends to give almost zero weight to all samples.

Dependent-Sample Monte Carlo Methods

- We want an algorithm whose samples get better over time.
- Two main strategies for generating dependent samples:
  - Sequential Monte Carlo:
    - Importance sampling where proposal q_t changes over time from simple to posterior.
    - AKA sequential importance sampling, annealed importance sampling, particle filter.
    - "Particle Filter Explained without Equations": `https://www.youtube.com/watch?v=aUkBa1zMKv4`
  - Markov chain Monte Carlo (MCMC).
    - Design Markov chain whose stationary distribution is the posterior.
- These are the main tools to sample from high-dimensional distributions.

Markov Chain Monte Carlo

- We've previously discussed Markov chain Monte Carlo (MCMC).
    1. Based on generating samples from a Markov chain q.
    2. Designed so stationary distribution π of q is target distribution p.

- If we run the chain long enough, it gives us samples from p.

- Gibbs sampling is an example of an MCMC method.
    - Sample x_j conditioned on all other variables x_{-j}.

- Note that before we were sampling states according to a UGM, in Bayesian models we're sampling parameters according to the posterior.

Limitations of Gibbs Sampling

- Gibbs sampling is nice because it has no parameters:
  - You just need to decide on the blocks and figure out the conditionals.
- But it isn't always ideal:
  - Samples can be very correlated: slow progress.
  - Conditionals may not have a nice form:
    - If Markov blanket is not conjugate, need rejection sampling (or numerical CDF).
- Generalization that can address these is Metropolis-Hastings:
  - Oldest algorithm among the "10 Best of the 20th Century".

Warm-Up to Metropolis-Hastings: “Stupid MCMC”

- Consider finding the expected value of a fair di:
  - For a 6-sided di, the expected value is 3.5.
- Consider the following “stupid MCMC” algorithm:
  - Start with some initial value, like “4”.
  - At each step, roll the di and generate a random number u:
    - If $u < 0.5$, “accept” the roll and take the roll as the next sample.
    - Othewise, “reject” the roll and take the old value (“4”) as the next sample.

Warm-Up to Metropolis-Hastings: “Stupid MCMC”

- Example:
  - Start with “4”, so record “4”.
  - Roll a 6 and generate 0.234, so record 6.
  - Roll a 3 and generate 0.612, so record 6.
  - Roll a 2 and generate 0.523, so record 6.
  - Roll a 3 and generate 0.125, so record 3.
- So our samples are 4,6,6,6,3,. . .
  - If you run this long enough, you will spend 1/6 of the time on each number.
  - So the dependent samples from this Markov chain could be used within Monte Carlo.
- It is “stupid” since you should just accept every sample (they are IID samples).
  - It works but it is twice as slow.

A Simple Example of Metropolis-Hastings

- Consider a loaded di that rolls a 6 half the time.
  - All others are equally likely, so they have probability 1/10.

- Consider the following "less stupid" MCMC algorithm:
  - At each step, we start with an old state x.
  - Generate a random number $\hat{x}$ uniformly between 1 and 6 (roll a fair di), and generate a random number u in the interval $[0, 1]$.
  - "Accept" this roll if

$$u < \frac{p(\hat{x})}{p(x)}.$$

  - So if we roll $\hat{x} = 6$, we accept it: $u < 1$ (''always move to higher probability'').
  - If $x = 2$ and roll $\hat{x} = 1$, accept it: $u < 1$ ("always move to same probability").
  - If $x = 6$ and roll $\hat{x} = 1$, we accept it with probability $1/5$.
    - We prefer high probability states, but sometimes move to low probability states.

- This has right probabilities as the stationary distribution (not yet obvious).
  - And accepts most samples.

Metropolis Algorithm

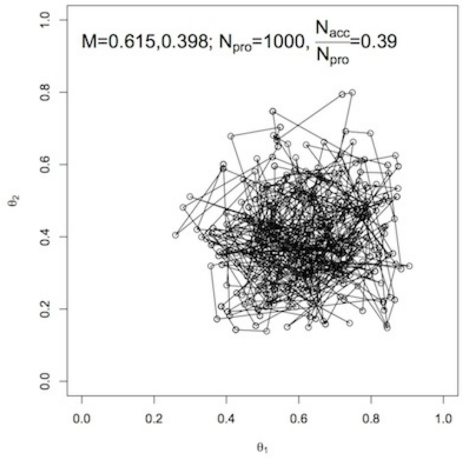
- The Metropolis algorithm for sampling from a continuous target $p(x)$:
    - On each iteration add zero-mean Gaussian noise to x^t to give proposal $\hat{x}^t$.
    - Generate u uniformly between 0 and 1.
    - "Accept" the sample and set $x^{t+1} = \hat{x}^t$ if

$$u \leq \frac{\tilde{p}(\hat{x}^t)}{\tilde{p}(x^t)}, \qquad \frac{\text{(probability of proposed)}}{\text{(probability of current)}}$$

    - Otherwise "reject" the sample and use x^t again as the next sample x^{t+1}.

- A random walk, but sometimes rejecting steps that decrease probability:
    - A valid MCMC algorithm on continuous densities, but convergence may be slow.
    - You can implement this even if you don't know normalizing constant.

Metropolis Algorithm in Action



Pseudo-code:

```
eps = randn(d,1)
xhat = x + eps
u = rand()
if u < ( p(xhat) / p(x) )
 set x = xhat
otherwise
  keep x
```

[illegible]

Metropolis Algorithm Analysis

- Markov chain with transitions $q_{ss'} = q(x^t = s' \mid x^{t-1} = s)$ is reversible if

$$\pi(s)q_{ss'} = \pi(s')q_{s's},$$

  for some distribution π (this condition is called detailed balance).

- Assuming we reach stationary, reversibility implies π is stationary distribution.
  - By summing reversibility condition over all s values we get

$$\sum_s \pi(s)q_{ss'} = \sum_s \pi(s')q_{s's}$$

$$\sum_s \pi(s)q_{ss'} = \pi(s')\underbrace{\sum_s q_{s's}}_{=1}$$

$$\sum_s \pi(s)q_{ss'} = \pi(s') \qquad \text{(stationary condition).}$$

- Metropolis is reversible (bonus slide) so has correct stationary distribution.

Metropolis-Hastings

- Gibbs and Metropolis are special cases of Metropolis-Hastings.
    - Uses a proposal distribution $q(\hat{x} \mid x)$, giving probability of proposing $\hat{x}$ at x.
        - In Metropolis, q is a zero-mean Gaussian.

- Metropolis-Hastings accepts a proposed $\hat{x}^t$ if

$$u \leq \frac{\tilde{p}(\hat{x}^t)q(x^t \mid \hat{x}^t)}{\tilde{p}(x^t)q(\hat{x}^t \mid x^t)},$$

where extra terms ensure reversibility for asymmetric q:
    - E.g., if you are more likely to propose to go from x^t to $\hat{x}^t$ than the reverse.

- This again works under very weak conditions, such as $q(\hat{x}^t \mid x^t) > 0$.
    - You can make performance much better/worse with an appropriate q.

Metropolis-Hastings Example: Rolling Dice with Coins

- Conisder the following random walk on the numbers 1-6:
  - If $x = 1$, always propose 2.
  - If $x = 2$, 50% of the time propose 1 and 50% of the time propose 3.
  - If $x = 3$, 50% of the time propose 2 and 50% of the time propose 4.
  - If $x = 4$, 50% of the time propose 3 and 50% of the time propose 5.
  - If $x = 5$, 50% of the time propose 4 and 50% of the time propose 6.
  - If $x = 6$, always propose 5.

- "Flip a coin: go up if it's heads and go down it it's tails".
  - The PageRank "random surfer" applied to this graph:

$$\begin{tikzcd}
\textcircled{1} \arrow[r, bend left] & \textcircled{2} \arrow[l, bend left] \arrow[r, bend left] & \textcircled{3} \arrow[l, bend left] \arrow[r, bend left] & \textcircled{4} \arrow[l, bend left] \arrow[r, bend left] & \textcircled{5} \arrow[l, bend left] \arrow[r, bend left] & \textcircled{6} \arrow[l, bend left]
\end{tikzcd}$$

Metropolis-Hastings Example: Rolling Dice with Coins

- Suppose we want to sample from a fair 6-sided di.
  - p(x=1) = p(x=2) = p(x=3) = p(x=4) = p(x=5) = p(x=6) = 1/6.
  - But don't have a di or a computer and can only flip coins.

- Use random walk as transitions q in Metropolis-Hastings.
  - $q(\hat{x} = 2 \mid x = 1) = 1$, $q(\hat{x} = 1 \mid x = 2) = \frac{1}{2}$, $q(\hat{x} = 2 \mid x = 3) = 1/2, \ldots$
  - If x is in the "middle" (2-5), we'll always accept the random walk.
    - If $x = 3$ and we propose $\hat{x} = 2$, then:

$$u < \frac{p(\hat{x} = 2)}{p(x = 3)} \frac{q(x = 3 \mid \hat{x} = 2)}{q(\hat{x} = 2 \mid x = 3)} = \frac{1/6}{1/6} \frac{1/2}{1/2} = 1.$$

    - If $x = 2$ and we propose $\hat{x} = 1$, then we test $u < 2$ which is also always true.

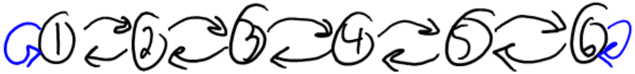
    - If x is at the end $(1$ or $6)$, you accept with probability $1/2$:

$$u < \frac{p(\hat{x} = 2)}{p(x = 1)} \frac{q(x = 1 \mid \hat{x} = 2)}{q(\hat{x} = 2 \mid x = 1)} = \frac{1/6}{1/6} \frac{1/2}{1} = \frac{1}{2}.$$

Metropolis-Hastings Example: Rolling Dice with Coins

- So Metropolis-Hastings modifies random walk probabilities:
  - If you're at the end (1 or 6), stay there half the time.
  - This accounts for the fact that 1 and 6 have only one neighbour.
    - Which means they aren't visited as often by the random walk.

- Could also be viewed as a random surfer in a different graph:



- You can think of Metropolis-Hastings as the modification that "makes the random walk have the right probabilities".
  - For any (reasonable) proposal distribution q.

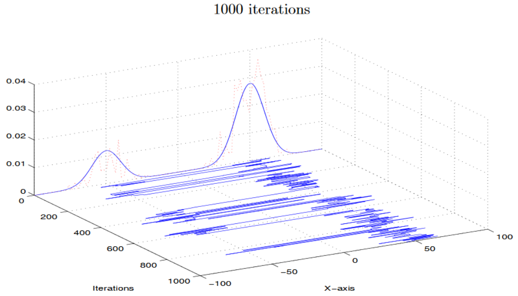
Metropolis-Hastings

- Simple choices for proposal distribution q:
  - Metropolis originally used random walks: $x^t = x^{t-1} + \epsilon$ for $\epsilon \sim \mathcal{N}(0, \Sigma)$.
  - Hastings originally used independent proposal: $q(x^t \mid x^{t-1}) = q(x^t)$.
  - Gibbs sampling updates single variable based on conditional:
    - In this case the acceptance rate is 1 so we never reject.
  - Mixture model for q: e.g., between big and small moves.
  - "Adaptive MCMC": tries to update q as we go: needs to be done carefully.
  - "Particle MCMC": use particle filter to make proposal.

- Unlike rejection sampling, we don't want acceptance rate as high as possible:
  - High acceptance rate may mean we're not moving very much.
  - Low acceptance rate definitely means we're not moving very much.
  - Designing q is an "art".

Mixture Proposal Distribution

Metropolis-Hastings for sampling from mixture of Gaussians:



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

- With a random walk q we may get stuck in one mode.
- We could have proposal be mixture between random walk and “mode jumping”.

Advanced Monte Carlo Methods

- Some other more-powerful MCMC methods:
  - Block Gibbs sampling improves over single-variable Gibb sampling.
  - Collapsed Gibbs sampling (Rao-Blackwellization): integrate out variables that are not of interest.
    - E.g., integrate out hidden states in Bayesian hidden Markov model.
    - E.g., integrate over different components in topic models.
    - Provably decreases variance of sampler (if you can do it, you should do it).
  - Auxiliary-variable sampling: introduce variables to sample bigger blocks:
    - E.g., introduce z variables in mixture models.
    - Also used in Bayesian logistic regression (beginning with Albert and Chib).

Advanced Monte Carlo Methods

- Trans-dimensional MCMC:
  - Needed when dimensionality of problem can change on different iterations.
  - Most important application is probably Bayesian feature selection.
- Hamiltonian Monte Carlo:
  - Faster-converging method based on Hamiltonian dynamics.
- Population MCMC:
  - Run multiple MCMC methods, each having different “move” size.
  - Large moves do exploration and small moves refine good estimates.
    - With mechanism to exchange samples between chains.

Outline

Monte Carlo vs. Variational Inference

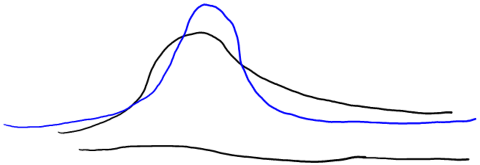
Two main strategies for approximate inference:

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.

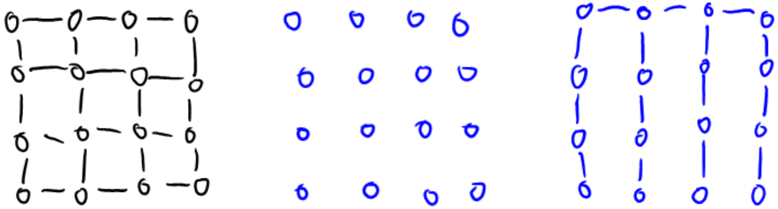
       (or mixtures of these simple distributions)
   - Turns inference into optimization.

Variational Inference Illustration

- Approximate non-Gaussian p by a Gaussian q:



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



- Variational methods try to find simple distribution q that is closets to target p.
  - This isn't consistent like MCMC, but can be very fast.

Laplace Approximation

- A classic variational method is the Laplace approximation.
  1. Find an x that maximizes $p(x)$,

$$x^* \in \underset{x}{\text{argmin}}\{-\log p(x)\}.$$

  2. Computer second-order Taylor expansion of $-\log p(x)$ at x^*.

$$-\log p(x) \approx f(x^*) + \underbrace{\nabla f(x^*)}_{0}{}^T(x - x^*) + \frac{1}{2}(x - x^*)^T\nabla^2 f(x^*)(x - x^*).$$

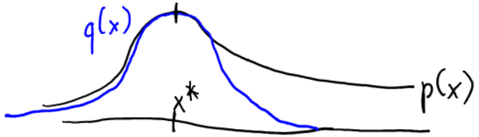
  3. Find Gaussian distribution q where $-\log q(x)$ has same Taylor expansion.

$$-\log q(x) = f(x^*) + \frac{1}{2}(x - x^*)\nabla^2 f(x^*)(x - x^*),$$

  so q follows a $\mathcal{N}(x^*, \nabla^2 f(x^*)^{-1})$ distribution.
    - This is the same approximation used by Newton's method in optimization.

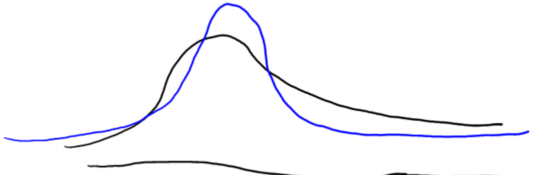
Laplace Approximation

- So Laplace approximation replaces complicated $p(x)$ with Gaussian $q(x)$.
  - Centered at mode and agreeing with 1st/2nd-derivatives of log-likelihood:



- Now you only need to compute Gaussian integrals (linear algebra for many f).
  - Very fast: just solve an optimization (compared to super-slow MCMC).
  - Bad approximation if posterior is heavy-tailed, multi-modal, skewed, etc.

- It might not even give you the "best" Gaussian approximation:



Kullback-Leibler (KL) Divergence

- How do we define "closeness" between a distribution p and q?
- A common measure is Kullback-Leibler (KL) divergence between p and q:

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

  - Replace sum with integral for continuous families of q distributions.
- Also called information gain: "information lost when p is approximated by q".
  - If p and q are the same, we have $KL(p \,||\, q) = 0$ (no information lost).
  - Otherwise, $KL(p \,||\, q)$ grows as it becomes hard to predict p from q.
- Unfortunately, this requires summing/integrating over p.
  - The problem we are trying to solve.

Minimizing Reverse KL Divergence

- Instead of using KL, most variational methods minimize reverse KL,

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

which just swaps all p and q values in the definition (KL is not commutative).
    - Not intuitive: "how much information is lost when we approximate q by p".

- But, reverse KL only needs unnormalized distribution $\tilde{p}$,

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

- By non-negativiy of KL this also gives a lower bound on $\log(Z)$.
    - Called the ELBO ("evidence lower bound").

Coordinate Optimization: Mean Field Approximation

- This "variational lower bound" still seems difficult to work with.
  - But with appropriate q we can do coordinate optimization.
- Consider minimizing reverse KL with independent q,

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

where we choose q to be conjugate (usually discrete or Gaussian).

- If we fix q_{-j} and optimize the functional q_j we obtain (see Murphy's book)

$$q_j(x_j) \propto \exp\left(\mathbb{E}_{q_{-j}}[\log \tilde{p}(x)]\right),$$

which we can use to update q_j for a particular j.

Coordinate Optimization: Mean Field Approximation

- Each iteration we choose a j and set q based on mean (of neighbours),

$$q_j(x_j) \propto \exp\left(\mathbb{E}_{q_{-j}}[\log \tilde{p}(x)]\right).$$

- This improves the (non-convex) reverse KL on each iteration.

- Applying this update is called:
  - Mean field method (graphical models).
  - Variational Bayes (Bayesian inference).

3 Coordinate-Wise Algorithms

- ICM is a coordinate-wise method for approximate decoding:
  - Choose a coordinate i to update.
  - Maximize x_i keeping other variables fixed.
- Gibbs sampling is a coordinate-wise method for approximate sampling:
  - Choose a coordinate i to update.
  - Sample x_i keeping other variables fixed.
- Mean field is a coordinate-wise method for approximate marginalization:
  - Choose a coordinate i to update.
  - Update $\underbrace{q_i(x_i)}_{\text{for all } x_i}$ keeping other variables fixed ($q_i(x_i)$ approximates $p_i(x_i)$).

3 Coordinate-Wise Algorithms

- Consider a pairwise UGM:

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

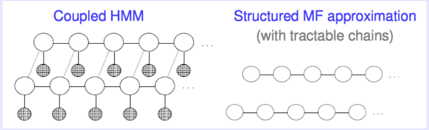
- ICM for updating a node i with 2 neighbours (j and k).
  1. Compute $M_i(x_i) = \phi_i(x_i)\phi_{ij}(x_i, x_j)\phi_{ik}(x_i, x_k)$ for all x_i.
  2. Set x_i to the largest value of $M_i(x_i)$.

- Gibbs for updating a node i with 2 neighbours (j and k).
  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. Sample x_i proportional to $M_i(x_i)$.

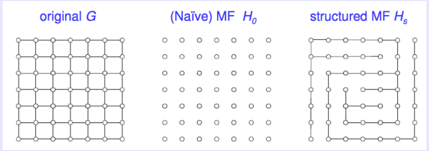
- Mean field for updating a node i with 2 neighbours (j and k).
  1. Compute $M_i(x_i) = \exp\left(\sum_{x_j} q_j(x_j)\log\phi_{ij}(x_i, x_j) + \sum_{x_k} q_k(x_k)\log\phi_{ik}(x_i, x_k)\right)$.
  2. Set $q_i(x_i)$ proportional to $\phi_i(x_i)M_i(x_i)$.

Structure Mean Field

- Common variant is structured mean field: q function includes some of the edges.



http://courses.cms.caltech.edu/cs155/slides/cs155-14-variational.pdf



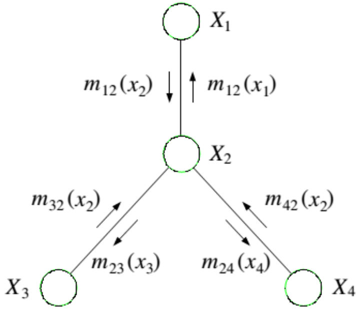
http://courses.cms.caltech.edu/cs155/slides/cs155-14-variational.pdf

- Original LDA article proposed a structured mean field approximation.

Previously: Belief Propagation

- We've discussed belief propagation for forest-structured UGMs.

(undirected graphs with no loops, which must be pairwise)



https://www.quora.com/

Probabilistic-graphical-models-what-are-the-relationships-between-sum-product-algorithm-belief-propagation-and-junction-tree-

- Defines "messages" that can be sent along each edge.
  - Generalizes forward-backward algorithm.

Loopy Belief Propagation

- In pairwise UGM, belief propagation "message" from parent p to child c is gven 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.
  - We get marginals by multiplying all incoming messages with local potentials.

- Loopy belief propagation: a "hacker" approach to approximate marginals:
  - Choose an edge ic to update.
  - Update messages $M_{ic}(x_c)$ keeping all other messages fixed.
  - Repeat until "convergence".
    - We approximate marginals by multiplying all incoming messages with local potentials.

- Empirically much better than mean field, we've spent 20 years figuring out why.

Discussion of Loopy Belief Propagation

- Loopy BP decoding is used for "error correction" in WiFi and Skype.
    - Called "turbo codes" in information theory.
- Loopy BP is not optimizing an objective function.
    - Convergence of loopy BP is hard to characterize: does not converge in general.
- If it converges, loopy BP finds fixed point of "Bethe free energy":
    - Better approximation than mean field, but not a lower/upper bound.
- Recent works give convex variants that upper bound Z.
    - Tree-reweighted belief propagation.
    - Variations that are guaranteed to converge.
- Messages only have closed-form update for conjugate models.
    - Can approximate non-conjugate models using expectation propagation.

Summary

- Markov chain Monte Carlo generates a sequence of *dependent samples*:
  - But asymptotically these samples come from the posterior.
- Metropolis-Hastings allows arbitrary "proposals".
  - With good proposals works much better than Gibbs sampling.

- Variational methods approximate p with a simpler distribution q.
  - Mean field approximation minimizes KL divergence with independent q.
  - Loopy belief propagation is a heuristic that often works well.

- Next time: non-parametric Bayes new generative deep learning methods.
  - I may go over time.

Metropolis Algorithm Analysis

- Metropolis algorithm has $q_{ss'} > 0$ (sufficient to guarantee stationary distribution is unique and we reach it) and satisfies detailed balance with target distribution p,
$$p(s)q_{ss'} = p(s')q_{s's}.$$
- We can show this by defining transition probabilities
$$q_{ss'} = \min\left\{1, \frac{\tilde{p}(s')}{\tilde{p}(s)}\right\},$$
and observing that
$$\begin{aligned} p(s)q_{ss'} &= p(s)\min\left\{1, \frac{\tilde{p}(s')}{\tilde{p}(s)}\right\} = p(s)\min\left\{1, \frac{\frac{1}{Z}\tilde{p}(s')}{\frac{1}{Z}\tilde{p}(s)}\right\} \\ &= p(s)\min\left\{1, \frac{p(s')}{p(s)}\right\} = \min\left\{p(s), p(s')\right\} \\ &= p(s')\min\left\{1, \frac{p(s)}{p(s')}\right\} = p(s')q_{s's}. \end{aligned}$$