CS281: Advanced ML **November 6, 2017**

Lecture 17: Loopy BP, Gibbs Sampling, and VI with Gradients

Lecturer: Sasha Rush *Scribes: Patrick Varin, Lillian Pentecost, André Snoeck, Baojia Tong*

17.1 Loopy BP

17.1.1 History

The history of Loopy-BP began in 1988 with Judea Pearl who tried to analyze the behavior of the BP algorithm (which gives exact marginal inference on tree) on graphs that are not trees, like the Ising model. Remember the message passing algorithms is

$$m_{s\to t}(x_t) = \sum_{x_s} \psi(x_s)\psi(x_s, x_t) \prod_{u\in \text{NBR}(s)-t} m_{u\to s}(x_s)$$
$$bel_s(x_s) \propto \psi(x_s) \prod_{t\in \text{NBR}(s)} m_{t\to s}(x_s)$$

Note that this algorithm doesn't require an ordering on the nodes, so it can naturally extend to cyclic graphs. Around 1998, a decade after Pearl raised the question of the BP algorithm on general graphs, papers studying *Turbo coding* or *Low density parity check* (LDPC) codes used the BP algorithm on cyclical graphs with empirical success, motivating more research. This research drew parallels between loopy-BP and variational inference.

17.1.2 Implementations

We can implement loopy belief propagation either *synchronously* or *asynchronously*

- Synchronous Updates: All of the nodes are updated togethers, so that every node at iteration t depends on it's Markov blanket at $t-1$
  - parallelizable
  - usually takes more updates
- Asynchronous Updates: The nodes are given an order and updated sequentially
  - sequential

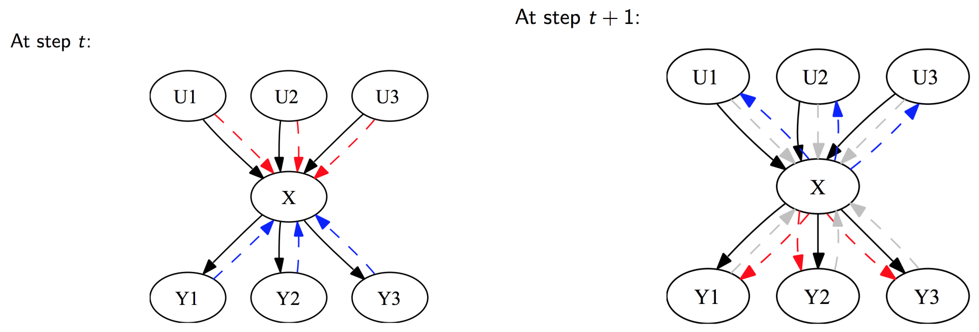

Figure 17.1: Message passing example: left (incoming), right (outgoing)

- usually converges in fewer updates

These updates look similar to mean field variational inference with some differences in performance.

- Loopy BP is exact on trees, mean-field variational inference is not.
- Loopy BP is not guaranteed to converge, where VI will.
- If Loopy BP converges it is not guaranteed to converge to a local optimum.

Loopy BP is generally more costly than mean field VI, because it stores node beliefs in addition to edge messages, whereas mean field VI only stores node parameters. If the graph is large and dense, then Loopy BP might be very costly.
In practice, because Loopy BP contains more information about the problem than mean field VI it tends to work better.

17.2 Gibbs Sampling (Preview)

Gibbs Sampling is a different method for approximate inference. It is a special case of Markov Chain Monte Carlo (MCMC), which will be covered in more detail in coming lectures.
The main idea of Gibbs Sampling is to re-estimate the posteriors for x_i assuming that we have we have explicit point estimates for all the other nodes in the graph (x_{-i}):

$$\tilde{x}_i \sim P(x_i|\tilde{x}_{-i})$$

Naturally, we only care about the Markov blanket when estimating a particular node. Gibbs sampling is not a parallel algorithm, the samples have to be generated sequentially. However, we can go through our graph in any particular order and update each node accordingly.

17.2.1 Example: Gibbs Sampling Topic Modeling

In the following example, we present the Gibbs update associated to the topic modeling that we defined in Lecture 16. Remember that

$$\begin{aligned}
\beta_k &\sim Dir(\eta) && k \text{ topics}\\
\pi_n &\sim Dir(\alpha) && n \text{ documents}\\
z_{ni} &\sim Cat(\pi_n) && \text{Draw topic for each word and each document}\\
w_{ni} &\sim Dir(\beta_{z_{ni}}) && \text{Draw a word from a topic}
\end{aligned}$$

and the full joint distribution is given by

$$p(\beta_k, \pi_n, \{z\}, \{w\}) = \prod_k p(\beta_k) \prod_n p(\pi_n) \prod_{n,i} p(z_{ni}|\pi_n) p(w_{ni}|z_{ni})$$

See Figure 17.2 for reference.
The updates for Gibbs sampling are similar to those of Mean Field (see Lecture 16), albeit much simpler. We need to compute the conditional probability for each of the following

$$\begin{aligned}
z_{ni} = k|\pi_n, w_{ni} &\propto \pi_{nk}\beta_{k,w_{ni}}\\
\pi_n|\alpha, z_{ni} &= Dir(a_k + \sum_i \mathbb{1}(z_{ni} = k))\\
\beta_k = d|\eta, z_{ni} &= Dir(\eta_d + \sum_i \mathbb{1}(z_{ni} = k, w_{ni} = d))
\end{aligned}$$

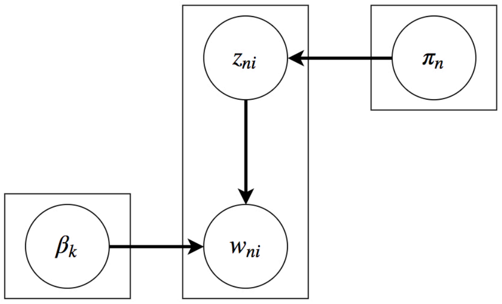

Figure 17.2: Graphical model associated to topic modeling example

The last two terms are essentially the full posterior that we saw in lecture 2. Although these updates are rather simple, it is used in many applications given it works with a wide variety of models. Typically, one would draw one sample per update. It is not clear whether drawing multiple samples or running more epochs helps the algorithm converge faster.

17.3 Variational Inference with Gradients

17.3.1 Introduction

Variational inference with gradients is also referred to as black box variational inference or stochastic variational inference (SVI) and is actively gaining traction in the ML commnity today. For example, UberAI just released a new framework on top of pytorch called *pyro* to support SVI. One reason why this is attractive is because SVI integrates well with autograd and deep learning.

17.3.2 Aside on Monte Carlo Sampling

If we consider some expectation over a distribution p of some function $f(z)$, as given below for discrete z, we often find this is intractable to compute.

$$\mathbb{E}_{z\sim p}[f(z)] = \sum_{z'} p(z')f(z')$$

We can approximate this expectation by sampling some N number of $\tilde{z}$, with each sample represented by $\tilde{z}_n$, which results in the following expression for the expectation:

$$\mathbb{E}_{z\sim p}[f(z)] \approx \frac{1}{N}\sum_{n=1}^{N} f(\tilde{z}_n)$$

This is much simpler to compute, and it can be shown that the resulting $\tilde{\mu}$ satisfies that $(\tilde{\mu} - \mu) \sim \mathcal{N}(0, \frac{\sigma^2}{N})$, which implies that the variance of the approximation with respect to the true mean decreases with the number of samples N.

17.3.3 Rough Idea

We will apply gradient optimization to the lower bound on $p(x)$ that we use to derive the variational objective. We use the formulation of the GMM and Mean Field approximation with variational parameters, developed in previous lectures, as an example. This is potentially more attractive than coordinate ascent because for a very large dataset compared to the number of classes (i.e. $n >> k$ in the GMM MF configuration) coordinate ascent requires (1) iterating through all data to perform updates to the variational

parameters and (2) storing variational parameters per-example (the number of the parameters grows with the size of the dataset).
The two wishes/ideas of this approach will be to (1) update global parameters (λ_k) on mini-batches to avoid the need to iterate over all samples (all λ_n) to perform update and (2) avoid storage of a dedicated λ_n for each n. The remainder of the lecture will focus on idea (1) above.

17.3.4 Deriving the Variational Objective

Letting $\theta = z_n, \mu_k$, we would like $p(x)$ for the GMM MF configuration previously developed. We aim to maximize the following lower bound (also from previous lecture) in order to approach $p(x)$, where q_λ is any q with variational parameters λ.

$$p(x) \geq \mathbb{E}_q[\log \frac{p(x,\theta)}{q_\lambda(\theta)}]$$

$$\max_q \mathbb{E}_q[\log \frac{p(x,\theta)}{q_\lambda(\theta)}]$$

Expressing $p(x,\theta) = p(\theta)p(x|\theta)$ and distributing the log, we find the variational objective, and we can interpret each of the three terms:

$$\max_q -\mathbb{E}_q[\log q_\lambda(\theta)] + \mathbb{E}_q[\log p(\theta)] + \mathbb{E}_q[\log p(x|\theta)]$$

In the expression above, we can recognize that the first term is just an entropy term, the second term includes the prior to effectively find a q close to $p(\theta)$ based on cross-entropy, and the third term contains the likelihood to reflect how well θ predicts the data. Furthermore, we see that the first two terms together form the negative inverse KL divergence of p and q, so we further simplify to the expression below:

$$L(\lambda) = \max_q -KL(q(\theta)||p(\theta)) + \mathbb{E}_q(\log p(x|\theta))$$

In the expression above, the KL divergence will act to tend q towards the prior $p(\theta)$ and the expectation term will give weight in q to parameters that tend to explain x.

17.3.5 Optimization via SGD

Next, we aim to apply optimization via SGD, and the $L(\lambda)$ given above becomes a loss function; we will maximize the value of this function by computing the gradient with respect to the parameters we care about (λ). Here, we will take a mini-batch of x values and compute $\nabla_\lambda L(\lambda)$. However, we notice that the expectation term in $L(\lambda)$ over q must be approximated, and the two methods to compute an approximation are (1) reinforcement and (2) reparameterization. This lecture only covers method (2), reparameterization, which also appears on HW4.
Reparameterization requires known q of a certain type and removes q from the expectation. Beginning with the form of the expectation, we can reformulate in terms of a change of variable from distribution q to the standard normal (letting $\theta = Az + b$) and then apply the Monte Carlo approximation to simplify the expression and make it computationally tractable:

$$\mathbb{E}_{q_\lambda}[p(x|\theta)] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[p(x|f_\lambda(z))]$$

$$\nabla_\lambda \mathbb{E}_{z\sim\mathcal{N}(0,1)}[p(x|f_\lambda(z))] = \mathbb{E}_{z\sim\mathcal{N}(0,1)}[\nabla_\lambda p(x|f_\lambda(z))]$$

$$\nabla_\lambda \mathbb{E}_{q_\lambda}[p(x|\theta)] \approx \frac{1}{N}\sum_{n=1}^{N} \nabla_\lambda p(x|f_\lambda(\tilde{z}_n))$$