Compass Seminar

Massively Parallel Bayesian Inference

Bayesian Inference

Want to compute the posterior

$$P(z|x) = \frac{P(x, z')}{P(x)} = \frac{P(x|z')P(z')}{\int_{\mathcal{Z}} P(x, z'')dz''}$$

where:

- x are data
- $z' \in \mathcal{Z}$ are latent variables with prior $P(z')$ (N.B. this is often high-dimensional with complicated correlations.)
- $P(z')$ is a prior over our latents
- $P(x, z')$ is our generative model
- $P(x|z')$ is our likelihood function
- $\int_{\mathcal{Z}} P(x, z'')dz'' = P(x)$ is our (usually intractable) normalising constant

For example, x may be records of sightings of bird species in various US states and $P(x, z') = P(x|z')P(z')$ is some model of how population affects sightings based on latents z' such as bird population, visibility, quality of bird-watchers' recordings &c.

For example, x may be covid infections per day, $P(x, z') = P(x|z')P(z')$ is some model of population interactions which takes latent 'settings' z' (how many people are infected, wearing masks, isolating &c.).

Because $P(x)$ is usually intractable, we often resort to Monte Carlo techniques that can approximate the posterior. One of the simplest such methods is importance sampling:

1. Sample K latent variables from a proposal distribution Q:

   $$z = (z^1, \ldots, z^K) \sim Q(z).$$

   (We'll assume each z^k is iid, so $Q(z) = \prod_{k=1}^K Q(z^k)$.)
2. Compute the importance weights:

   $$r_k(z) = \frac{P(x, z^k)}{Q(z^k)}.$$

3. Approximate the normalising constant using the 'global' estimator:

$$\mathcal{P}_{\text{global}}(z) = \frac{1}{K}\sum_{k=1}^{K} r_k(z)$$

which is unbiased

$$\mathbb{E}_{z\sim Q}[\mathcal{P}_{\text{global}}(z)] = \mathbb{E}_{z\sim Q}\left[\frac{1}{K}\sum_{k=1}^{K}\frac{P(x,z^k)}{Q(z^k)}\right] = \mathbb{E}_{z\sim Q}\left[\frac{P(x,z^k)}{Q(z^k)}\right] = \int_{\mathcal{Z}^{(k)}} P(x,z^k)dz^k = P(x).$$

Note that, in addition to giving an estimator for $P(x)$, the importance weights can be used to:

- Draw samples from the posterior: from our samples $z = (z^1, \ldots, z^K)$, sample z^k with probability $\propto r_k(z)$. (Note that this is *self-normalised* importance sampling, which is slightly biased... (but consistent, i.e. unbiased in the limit as $K \to \infty$).)
- Compute posterior moments:

$$m_{\text{global}}(z) = \frac{1}{K}\sum_{k=1}^{K}\frac{r_k(z)}{\mathcal{P}_{\text{global}}(z)}m(z^k)$$

which gives (another) consistent estimator, i.e.

$$\lim_{K\to\infty}\mathbb{E}_{z\sim Q}[m_{\text{global}}(z)] = m_{\text{post}} = \mathbb{E}_{p(z|x)}[m(z)] = \int_{\mathcal{Z}} m(z)P(z|x)dz$$

The Problem

Consider our latents z' as a collection of n random variables $z' = (z'_1, z'_2, \ldots, z'_n)$. As shown by Chatterjee & Diaconis (2018), for a high-quality posterior approximation via importance sampling, as the number of latent variables, n, in a model increases, the number of samples, K, must increase with $O(e^{D_{\text{KL}}(P||Q)}) \approx O(e^n)$. (This is a lot!)

But, if we have K joint samples of our n latent variables, how about we use all possible K^n combinations of the samples we have?

(E.g. the model $p(x|z) = \mathcal{N}(z_1, z_2)$ where $n = 2$ (so with independent priors on z_1, z_2 the full generative model would be something like $p(x,z) = \mathcal{N}(z_1, z_2)P(z_1)P(z_2)$).)

E.g. the model with three latent variables $z = (z_1, z_2, z_3)$ where the prior is given by

$$\begin{gathered} z_1 \sim \mathcal{N}(0,1) \\ z_2 \sim \mathcal{N}(z_1,1) \\ z_3 \sim \mathcal{N}(z_2,1) \\ \text{hence,}\quad P(z=(z_1,z_2,z_3)) = \mathcal{N}(z_1;0,1)\mathcal{N}(z_2;z_1,1)\mathcal{N}(z_3;z_2,1) \end{gathered}$$

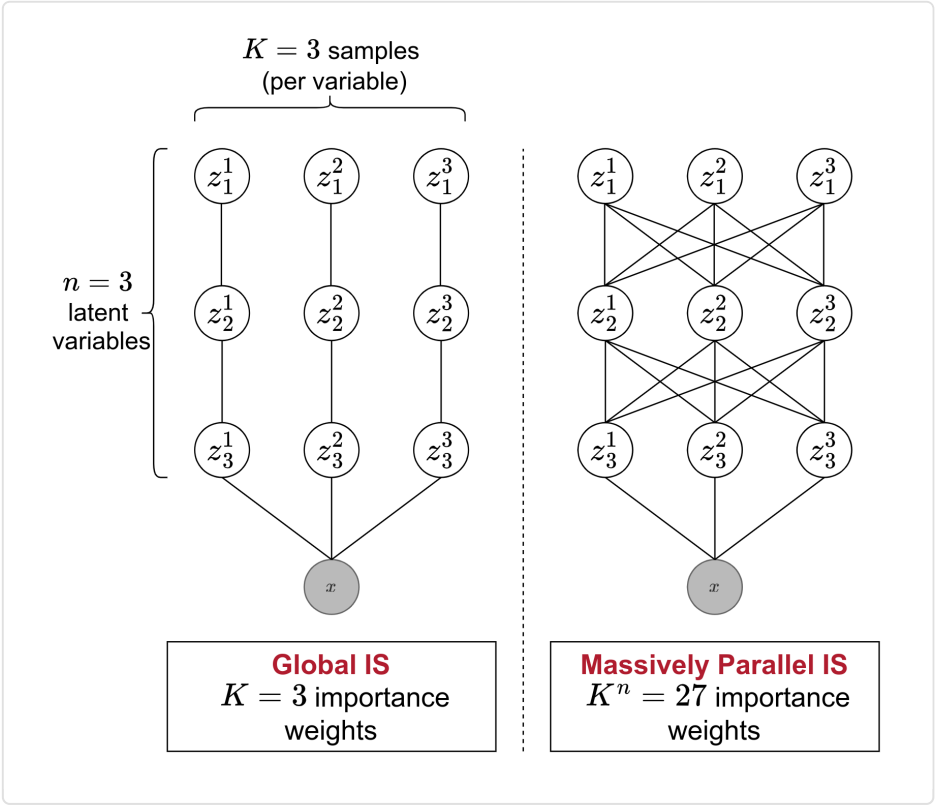
and we assume the likelihood is given by

$$\begin{aligned} x &\sim \mathcal{N}(z_3, 1) \\ P(x|z = (z_1, z_2, z_3)) &= \mathcal{N}(x; z_3, 1). \end{aligned}$$

We could use a proposal such as

$$\begin{aligned} z_1 &\sim \mathcal{N}(0, 2) \\ z_2 &\sim \mathcal{N}(z_1, 2) \\ z_3 &\sim \mathcal{N}(z_2, 2) \\ \text{hence,} \quad Q(z = (z_1, z_2, z_3)) &= \mathcal{N}(z_1; 0, 2)\mathcal{N}(z_2; z_1, 2)\mathcal{N}(z_3; z_2, 2). \end{aligned}$$

Then we can construct $K = 3$ importance weights $r_1(z), r_2(z), r_3(z)$ using regular ('global') importance weighting as shown on the left of the following diagram.



Then, we can construct a 'massively parallel' approximation of the normalising constant:

$$\mathcal{P}_{\mathrm{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z)$$

where

$$r_{\mathbf{k}}(z) = \frac{P(x, z^{\mathbf{k}})}{Q_{\mathrm{MP}}(z, \mathbf{k})}$$

and $\mathcal{K} = \{1, \ldots, K\}, \mathbf{k} = (k_1, \ldots, k_n) \in \mathcal{K}^n$ and $z^{\mathbf{k}} = (z_1^{k_1}, \ldots, z_n^{k_n})$.

Here we use a massively parallel proposal distribution Q_{MP} and likelihood $P(x, z^{\mathbf{k}})$ that factorise over the latent variables via (potentially different) hierarchical models:

$$Q_{\mathrm{MP}}(z, \mathbf{k}) = \prod_{i=1}^{n} Q_{\mathrm{MP}}(z_i^{k_i} | z_j \text{ for } j \in \mathrm{qa}(i)),$$

$$P(x, z^{\mathbf{k}}) = P\left(x | z_j^{k_j} \text{ for } j \in \mathrm{pa}(x)\right) \prod_{i=1}^{n} P(z_i^{k_i} | z_j^{k_j} \text{ for } j \in \mathrm{pa}(i)),$$

where $\mathrm{qa}(i)$ and $\mathrm{pa}(i)$ are the set of indices of parents of z_i in the proposal and generative models respectively.

THE IMPORTANT DETAIL ABOUT Q_{MP}: note that $Q_{\mathrm{MP}}(z)$ *is a distribution over all samples* $z = (z^1, \ldots, z^K)$*, not just the samples being indexed by* $\mathbf{k}$*, i.e.* $z^{\mathbf{k}}$. We need this dependency for everything to work, but there are many ways we could define Q_{MP} other than the (simple) graphical-model-type definition above (and the permutation/bootstrapping/mixture simple-sample proposals $Q_{\mathrm{MP}}(z_i)$). (Note that this is also why in the global IS case at the start we wrote $z = (z^1, \ldots, z^K) \sim Q$ and defined our ratios $r_k(z)$ with k in the subscript and taking in the whole collection of samples z, rather than just defining it as $r(z^k)$.)

Simpler explanation of Q_{MP}

In the global case, sampling is easy: we sample z_1^1, use that to sample z_2^1, use that to sample z_3^1, and then repeat $K - 1$ more times.

In the massively parallel case, we have to induce some distribution over 'parent' and 'child' samples, so what we do instead is:

- Sample K copies of the first latent, $z_1^1, \ldots, z_1^K$
- Sample K copies of the second latent $z_2^1, \ldots, z_2^K$ **using a parent sample selected (uniformly) at random from $(z_1^1, \ldots, z_1^K)$.

- Sample K copies of the second latent $z_3^1, \ldots, z_3^K$ **using a parent sample selected (uniformly) at random from $(z_2^1, \ldots, z_2^K)$.
- &c.
  Thus when we evaluate $Q_{MP}(z, \mathbf{k})$, we have to marginalise over all possible parent-child combinations (thereby inferring over our K^N samples).

Using previous work Aitchison (2019), Heap et al. (2023), we can not only compute this estimator efficiently, but also show that it is indeed unbiased:

$$\mathbb{E}_{z \sim Q_{\mathrm{MP}}}[\mathcal{P}_{\mathrm{MP}}(z)] = P(x).$$

A Note on Computation

A natural question to ask: isn't summing over K^n combinations computationally awful?
In a naive approach it would be, but we can 1) put all the computations on GPUs in the form of tensor products (which GPUs can't get enough of); and 2) exploit the conditional independencies of the given model to greatly simplify things.

To illustrate 2) let's reconsider the example model from before. To avoid complicating things too much, let's just look at summing over the generative probability for all K^n combinations (and take my word for the fact that the same/similar tricks can be used when summing over importance weights $r_{\mathbf{k}}(z)$).

Lets index our $n = 3$ latents by $\mathbf{k} = (i, j, k) \in [K]^3$.

$$\begin{aligned}\frac{1}{K^n} \sum_{i,j,k \in [K]^3} P(x, z = (z_1^i, z_2^j, z_3^k)) &= \frac{1}{K^n} \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{k=1}^{K} P(z_1^i) P(z_2^j | z_1^i) P(z_3^k | z_2^j) P(x | z_3^k) \\ &= \frac{1}{K^n} \sum_{i=1}^{K} P(z_1^i) \sum_{j=1}^{K} P(z_2^j | z_1^i) \sum_{k=1}^{K} P(z_3^k | z_2^j) P(x | z_3^k)\end{aligned}$$

In the first equality, we've simply factorised the joint according to the likelihood and prior, whilst in the second, we've moved factors outside of sums of variables to which they are (conditionally) independent. This is going to be far more efficient, both in terms of computation, and in terms of memory (previously, we had a tensor of size $K \times K \times K$ being summed K^3 times, now we have tensors of size $K \times K$ each only being summed K times (***I think...***)).

How can we use this?

So far we've just used a fixed proposal Q, but what if we parameterise it by some $\theta \in \Theta$ and try to make $Q_\theta(z)$ approximate $P(z|x)$ in some sense? We'll now call Q_θ the *approximate posterior*.

(Here we're ignoring the massively parallel stuff, just know that Q_θ can be a global or massively parallel distribution (and massively parallel gets us better results).)

Variational Inference

One method for optimising θ is variational inference, in which we minimise the KL between Q_θ and $P(z|x)$:

$$\hat{\theta} = \arg\min_\theta D_{\mathrm{KL}}(Q_\theta(z)||P(z|x)) = \arg\min_\theta \int_{\mathcal{Z}} Q_\theta(z) \log \frac{Q_\theta(z)}{P(z|x)} dz$$

But this integral over $\mathcal{Z}$ is usually intractable (and dodgy because we're approximating $P(z|x)$ using e.g. importance sampling with $Q_\theta(z)$)! So we rearrange the KL to obtain the ELBO, which we *can* evaluate:

$$\begin{aligned} D_{\mathrm{KL}}(Q_\theta(z)||P(z|x)) = \int_{\mathcal{Z}} Q_\theta(z) \log \frac{Q_\theta(z)}{P(z|x)} dz &= \int_{\mathcal{Z}} Q_\theta(z) \log \frac{Q_\theta(z)P(x)}{P(x,z)} dz \\ &= \int_{\mathcal{Z}} Q_\theta(z) \left[\log \frac{Q_\theta(z)}{P(x,z)} + \log P(x)\right] dz \\ &= \mathbb{E}_{Q_\theta(z)}\left[\log \frac{Q_\theta(z)}{P(x,z)}\right] + \mathbb{E}_{Q_\theta(z)}\left[\log P(x)\right] \\ &= \mathbb{E}_{Q_\theta(z)}\left[\log \frac{Q_\theta(z)}{P(x,z)}\right] + \log P(x) \end{aligned}$$

Now $P(x)$ is fixed with respect to θ, meaning that our optimisation is equivalent to minimising $\mathbb{E}_{Q_\theta(z)}\left[\log \frac{Q_\theta(z)}{P(x,z)}\right]$, which is the same as maximising $\mathbb{E}_{Q_\theta(z)}\left[\log \frac{P(x,z)}{Q_\theta(z)}\right]$, a quantity we call the *Evidence Lower Bound* (or *ELBO*) since it forms a lower bound on the log model evidence:

$$\mathcal{L} = \mathbb{E}_{Q_\theta(z)}\left[\log \frac{P(x,z)}{Q_\theta(z)}\right] = \log P(x) - D_{\mathrm{KL}}(Q_\theta(z)||P(z|x)) \leq \log P(x).$$

(In which we've used the fact that KL divergences are always non-negative.)

It is common (and well-motivated) to use the tightness of the ELBO as a measure of the approximate posterior's quality.

The ELBO looks very similar to all the importance sampling equations we went through above. Indeed, we can motivate three VI algorithms immediately:

1. 'Single-sample' VI (VAE (Variational Autoencoders), Kingma & Welling (2014)):
    - Draw a sample $z' \sim Q_\theta$ and maximise

$$\mathcal{L} = \log \frac{P(x,z)}{Q_\theta(z)},$$

or better, draw K samples $z^1, \ldots, z^K \sim Q_\theta$ and maximise a Monte Carlo estimate

$$\mathcal{L}_{\text{VAE}} = \frac{1}{K} \sum_{k=1}^{K} \log \frac{P(x, z^k)}{Q_\theta(z^k)}.$$

(Then repeat on the updated Q_θ.)

2. Multi-sample VI (IWAE (Importance Weighted Autoencoders), Burda et al. (2016)).
    - Do the same as VAE, but switch the order of the sum and the log!

$$\mathcal{L}_{\text{IWAE}} = \log \frac{1}{K} \sum_{k=1}^{K} \frac{P(x, z^k)}{Q_\theta(z^k)} = \log \mathcal{P}_{\text{global}}.$$

Thanks to Jensen's inequality (non-Wikipedia reference: Jensen (1906)), since $\log$ is a convex function, we have that $\mathcal{L}_{\text{IWAE}}$ is a tighter bound than $\mathcal{L}_{\text{VAE}}$:

$$\mathcal{L}_{\text{VAE}}(\theta) \leq \mathcal{L}_{\text{IWAE}}(\theta) \leq \log P(x).$$

($\mathcal{L}_{\text{IWAE}}$ also has the pleasant property that each sample z^k contributes to the gradient $\nabla_\theta \mathcal{L}_{\text{IWAE}}(\theta)$ in proportion to its importance weight $r_k(z)$, whereas the samples contribute an equal weight to the gradient of $\mathcal{L}_{VAE}$. This greatly helps speed up gradient-based optimisation methods.)

3. Massively Parallel VI (TMC (Tensor Monte Carlo), Aitchison (2019)).
    - Do the same as IWAE, but use the massively parallel ELBO!

$$\mathcal{L}_{\text{IWAE}} = \log \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} \frac{P(x, z^{\mathbf{k}})}{Q_{\text{MP},\theta}(z, \mathbf{k})} = \log \mathcal{P}_{\text{MP}}.$$

This -- perhaps as expected -- turns out to be a tighter bound than IWAE (definitely empirically; I'm not sure whether there's a proof laying about somewhere or not...)

Following on from TMC, Heap et al. (2023) implemented a similar massively parallel inference algorithm (which doesn't rely on gradient-based optimisation) building on Reweighted Wake-Sleep (RWS) (Hinton et al. (1995)). Recent work extended this with an algorithm named QEM (Expectation-Maximisation for Approximate Posteriors) which bears similarities to MP-RWS and also to adaptive importance sampling (AIS) techniques (e.g. Adaptive Importance Sampling in Signal Processing (2015)), also avoiding the need for gradient-based optimisation.

In summary, we can (broadly speaking) rank the above algorithms as follows (getting better from left to right):

$$\text{VAE} \prec \text{IWAE} \prec \text{TMC/MP-VI} \prec \text{MP-RWS} \prec \text{QEM}$$

(Although the relationship between MP-RWS and QEM isn't quite that straightforward...)

QEM

Posterior Moment Estimates (Source-Term Trick)

It turns out (Bowyer et al. (2023)) that we can use the source-term trick (moment-generating function) to efficiently compute massively parallel posterior moment estimates.

$$m_{\mathrm{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\mathrm{MP}}(z)} m(z^{\mathbf{k}})$$

Computation of this estimator can be greatly accelerated by using the source-term trick, in which we differentiate a slightly modified marginal likelihood estimator, introducing an auxiliary variable $J \in \mathbb{R}$:

$$\mathcal{P}^{\exp}_{\mathrm{MP}}(z, J) = \frac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z) \exp(J m(z^{\mathbf{k}}))$$

(Note that $\mathcal{P}^{\exp}_{\mathrm{MP}}(z, J = 0) = \mathcal{P}_{\mathrm{MP}}(z)$.)

By differentiating this estimator with respect to J we can compute the moments of the posterior distribution (which can be done efficiently using an automatic differentiation framework e.g. PyTorch):

$$\left.\frac{\partial}{\partial J}\right|_{J=0} \log \mathcal{P}^{\exp}_{\mathrm{MP}}(z, J) = \frac{\left.\frac{\partial}{\partial J}\right|_{J=0} \mathcal{P}^{\exp}_{\mathrm{MP}}(z, J)}{\mathcal{P}_{\mathrm{MP}}(z)} = \frac{\frac{1}{K^n} \sum_{\mathbf{k} \in \mathcal{K}^n} r_{\mathbf{k}}(z) m(z^{\mathbf{k}})}{\mathcal{P}_{\mathrm{MP}}(z)} = m_{\mathrm{MP}}(z)$$

Adaptive Importance Sampling (via Exponential Moving Average)

Just do moment matching on your approx. posterior (with EMA for stability).