DATA20023 Bayesian Machine Learning

Exercise 5: Gradient-based variational approximation

---

1 Gradients for variational approximation

Consider again the same Gamma-Poisson model with two parameters u and v, studied previously in Exercises 2 and 3. Now we want to consider a **variational approximation** for this using gradient-based learning, using the approximation $q(u, v) = q(u|\mu_u, \sigma_u)q(v|\mu_v, \sigma_v)$ where both terms are normal distributions (because reparameterization is easier for them).

Instead of learning the optimal parameters, we will be simply looking at different ways of computing the gradient wrt $\boldsymbol{\lambda} = [\mu_u, \sigma_u, \mu_v, \sigma_v]$. If you wish you can also try optimizing the ELBO, but remember that our parameterization is very fragile – a gradient step might make σ negative.

(a) Implement Monte Carlo estimator for the evidence lower bound (ELBO)

$$\mathcal{L}(x, \boldsymbol{\lambda}) = \frac{1}{M} \sum_{m=1}^{M} [\log p(\mathbf{x}, u_m, v_m) - \log q(u_m|\mu_u, \sigma_u) - \log q(v_m|\mu_v, \sigma_v)]$$

by averaging the expression over M samples $[u_m, v_m]$ drawn from the approximation.

(b) Implement the *score function estimator* with elements (and analogous ones for μ_v and σ_v)

$$\frac{d\mathcal{L}}{d\mu_u} = \frac{1}{M} \sum_{m=1}^{M} w_m \frac{d \log q(u_m|\mu_u, \sigma_u)}{d\mu_u}, \qquad \frac{d\mathcal{L}}{d\sigma_u} = \frac{1}{M} \sum_{m=1}^{M} w_m \frac{d \log q(u_m|\mu_u, \sigma_u)}{d\sigma_u},$$

where $w_m = \log p(\mathbf{x}, u_m, v_m) - \log q(u_m|\mu_u, \sigma_u) - \log q(v_m|\mu_v, \sigma_v)$. You can either compute the derivative of the approximation yourself or use automatic differentiation in PyTorch. You can also try control variates if you wish – simply add or subtract a scalar number for all of the weights (hint: try something close to the mean of all w_m, to pull the weights around zero).

(c) Implement the *reparameterization gradient estimator*. Write both of the approximation terms as deterministic transformation $f(u, \mu, \sigma) = \mu + \sigma z$ of $z \sim \mathcal{N}(0, 1)$ and then form the derivatives using the chain rule. You will be needing the following terms:

- Derivative of $\log p(\mathbf{x}, u, v)$ wrt to u and v (which are provided in Model solutions of Exercise 3)
- Derivative of $\log q(u|\mu_u, \sigma_u)$ wrt to u (and naturally the same for v)
- Derivative of $f(z, \mu, \sigma)$ wrt to μ
- Derivative of $f(z, \mu, \sigma)$ wrt to σ

You can again use either analytic derivatives or use PyTorch for computing them, but you need to show how you construct the full gradient based on these terms to demonstrate you understood the concept properly.

(d) Provide (a) the ELBO and (b) the gradient estimates (computed with both estimators) for $\mu_u = 5, \sigma_u = 0.5, \mu_v = 1, \sigma_v = 0.2$. Use something like $M = 10,000$ so that you get accurate estimates. You should be getting fairly similar gradients with both estimators, but will not be having identical numbers.

(e) Study how the three estimators (one for ELBO and two for the gradients) behave for different M. For example, plot the mean and variance of the estimators as a function of M and summarise your findings. If you tried control variates, comment also on whether it helps in reducing the variance. We already know the score function estimator should have higher variance. How what M would be needed with score function estimator to reach the same variance reparameterization estimator has for $M = 10$?

2 Variational autoencoder

The notebook has a simplified implementation of VAE for MNIST digits.

(a) Familiarise yourself with the code and then **describe the model in writing** (preferably using mathematical notation and equations as well). You should describe all the critical choices that were made, regarding priors, likelihood, network architectures etc. Do you think the choices made here are good for this data? The code also has functionality for plotting some kind of reconstructions of the digits and a 2D plot of the samples. Explain what exactly we are plotting here.

(b) Run the code to see hot it works and then try to improve it, so that it would better describe the underlying data. You can work on both improving the model or the optimization process, but focus on making just 1-3 concrete changes rather than playing around by changing all possible parameters. For each of the modifications you tried, **write a brief description of (a) what you did, (b) what did you try to achieve with the change, and (c) discuss your findings using suitable plots.**

To find ideas how to improve the model or the optimization, you can start by reading some of the material linked below or you can search for other sources by simply googling for something like 'practical tips for improving VAE training'. Alternatively, you can start modifying the model itself to better match what you know about the data.

- The paper by Kingma and Welling available at `https://arxiv.org/pdf/1906.02691.pdf` talks about optimization challenges in Section 2.8.
- The paper by Higgins et al. available at `https://openreview.net/forum?id=Sy2fzU9gl` introduces a minor VAE variant that is easy to implement.
- The paper by Fu et al. available at `https://aclanthology.org/N19-1021.pdf` introduces a particular trick for improving optimization