Machine Learning and Pattern Recognition Tutorial 6

Instructor: Iain Murray

1. **Gibbs sampling and Metropolis, discrete variables:** We wish to construct a Markov chain to approximately sample high-dimensional binary vectors, $\mathbf{x} \in \{-1, +1\}^D$, from a distribution of the form:
$$P(\mathbf{x}) = \frac{1}{Z} f(\mathbf{x}), \tag{1}$$
where $f(\mathbf{x})$ is a positive function that we can easily evaluate for any given input.

   Describe how to implement Gibbs sampling for any such distribution. Assume you have access to a function `rand()` that returns samples from Uniform[0,1]. Your description should include how you use this function.

   We could also consider Metropolis sampling with a sequence of proposals that only alter one variable at a time. We could use proposals that deterministically attempt to change one of the variables to the opposite of its current setting:
$$Q_d(\mathbf{x}'; \mathbf{x}) \;=\; \begin{cases} 1 & x'_d = -x_d, \quad \mathbf{x}'_{\setminus d} = \mathbf{x}_{\setminus d} \\ 0 & \text{otherwise.} \end{cases}$$
   This proposal is then accepted or rejected using the usual Metropolis rule.

   Write down a modified version of your previous pseudo-code for this Metropolis rule. Does it differ from Gibbs sampling? If so, how? Might there be any reason to prefer one method over the other? As a special case, consider applying your algorithms to independent variables, each with $P(x_d = +1) = 0.5$ and $P(x_d = -1) = 0.5$.

2. **Gibbs sampling, continuous variables:** Why is it hard to write generic code to Gibbs sample from an arbitrary distribution of the form in Equation (1) if the variables are real-valued rather than discrete?

   We will now consider Gibbs sampling for a particular case: simulating the posterior over parameters of a univariate Gaussian given N independent observations $\mathcal{D} = \{x_n\}_{n=1}^N$. The probability of a single observation is:
$$p(x_n \,|\, \mu, \tau) \;=\; \mathcal{N}(x_n;\, \mu,\, 1/\tau) \;=\; \sqrt{\frac{\tau}{2\pi}}\, \exp\Big(-\frac{\tau}{2}(x_n - \mu)^2\Big)\,.$$
   Before we can infer the mean μ and precision[1] τ, we need to specify priors:
$$p(\mu) \;=\; \mathcal{N}(\mu;\, \mu_0,\, 1/\tau_0) \;=\; \sqrt{\frac{\tau_0}{2\pi}}\, \exp\Big(-\frac{\tau_0}{2}(\mu - \mu_0)^2\Big)\,,$$
$$p(\tau) \;=\; \text{Gamma}(\tau;\, \alpha, \beta) \;=\; \frac{\beta^\alpha}{\Gamma(\alpha)}\, \tau^{\alpha-1}\, \exp(-\beta\tau).$$
   Derive the conditional distributions required for Gibbs sampling from the joint posterior $p(\mu, \tau \,|\, \mathcal{D})$. You should describe them as standard distributions that could be sampled using library routines[2]. (The next paragraph gives hints if you want them.)

1. The precision, one over the variance, the 'natural parameter' of the Gaussian, is more convenient to work with here than the variance.
2. Warning: as you'll see on Wikipedia, there's more than one standard parameterization of the Gamma distribution. If you ever do implement something involving 'standard' probability distributions, check the documentation for the library routine to see whether its parameterization is the same as you expect!

Hints: 1) Both conditional distributions are proportional to the joint distribution $p(\mu, \tau, \mathcal{D})$ so write that down first. 2) Start with the resampling equation for τ, which you'll probably find to be less work than the one for μ. 3) Try to do the minimal amount of work required to identify each distribution. For example, a Gaussian is of the form:

$$\log \mathcal{N}(z;\, m,\, 1/t) \;=\; -\tfrac{t}{2}(z-m)^2 \,+\, \text{const.} \;=\; -\tfrac{t}{2}\, z^2 \,+\, (mt)\, z \,+\, \text{const. wrt } z.$$

If the log probability of a variable z is quadratic, you know it's Gaussian. You don't need to do a lot of work to rearrange the distribution into the standard form of a Gaussian pdf[3]. You simply need to identify the coefficients of z^2 and z. The coefficient of z^2 is minus half the precision. Then find the coefficient of z, divide by the precision and you get the mean. There's no need to track constants.

3. **Laplace approximation:** Exercise 27.1 in MacKay's textbook (p342) considers a Poisson likelihood and an improper (not normalizable) prior:

$$P(r \mid \lambda) = \exp(-\lambda)\frac{\lambda^r}{r!}, \qquad p(\lambda) \propto \frac{1}{\lambda}.$$

Find the Laplace approximation to the posterior over λ given an observed count r.

Now reparameterize the model in terms of $\ell = \log \lambda$. After performing the change of variables, the improper prior on $\log \lambda$ becomes uniform, $p(\ell)$ is constant. Find the Laplace approximation to the posterior over $\ell = \log \lambda$.

Which version of the Laplace approximation is better? It may help to plot the true and approximate posteriors of λ and ℓ for different values of the integer count r.

4. **Extra question on the Laplace approximation:** Exercise 27.2 in MacKay's textbook (p342) is slightly more involved. We probably won't go through the maths in the tutorial. However, this question may be useful for revision.

The question asks you to apply Laplace's method to approximate the integral

$$Z(u_1, u_2) \;=\; \int_{-\infty}^{\infty} f(a)^{u_1}(1-f(a))^{u_2}\, \mathrm{d}a,$$

where $f(a) = 1/(1+\exp(-a))$ is the logistic sigmoid. What we do first is identify a probability distribution:

$$p(a) \;=\; \frac{1}{Z(u_1, u_2)}\, f(a)^{u_1}(1-f(a))^{u_2}.$$

Take the log of this distribution, take first and second derivatives wrt. a and identify a Gaussian approximation. Then equate your normalized density Gaussian at the mode with the equation above to approximate Z.

This integral is actually tractable, so we can check the answer, and get some insight into applying Laplace's method. If we change the variable of integration to f, identifying $\mathrm{d}a = {}^{\mathrm{d}f}/_{f(1-f)}$, we get the standard beta integral we've seen before,

$$Z(u_1, u_2) \;=\; \int_0^1 f^{u_1-1}(1-f)^{u_2-1}\, \mathrm{d}f \;=\; \frac{\Gamma(u_1)\,\Gamma(u_2)}{\Gamma(u_1+u_2)}.$$

Why did MacKay apply the change of variables $f \to a$? That is, why might the Laplace approximation not work as well when applied directly to the variable f? Consider a range of settings of u_1 and u_2.

3. I was always told to explicitly 'complete the square', which seems more work than necessary.