MLPR Tutorial Sheet 4

1. **Conjugate priors:** (This question sets up some intuitions about the larger picture for Bayesian methods.)

   A *conjugate prior* for a likelihood function is a prior where the posterior is a distribution in the same family as the prior. For example, a Gaussian prior on the mean of a Gaussian distribution is conjugate to Gaussian observations of that mean.

   a) The *inverse-gamma distribution* is a distribution over positive numbers. It's often used to put a prior on the variance of a Gaussian distribution, because it's a conjugate prior.

      The inverse-gamma distribution has pdf (as cribbed from Wikipedia):

      $$p(z \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} z^{-\alpha-1} \exp\left(-\frac{\beta}{z}\right), \qquad \text{with } \alpha > 0, \;\; \beta > 0, \tag{1}$$

      where $\Gamma(\cdot)$ is a gamma function.[1]

      Assume we obtain N observations from a zero-mean Gaussian with unknown variance,

      $$x^{(n)} \sim \mathcal{N}(0, \sigma^2), \quad n = 1 \ldots N, \tag{2}$$

      and that we place an inverse-gamma prior with parameters α and β on the variance. Show that the posterior over the variance is inverse-gamma, and find its parameters.

      Hint: you can assume that the posterior distribution is a distribution; it normalizes to one. You don't need to keep track of normalization constants, or do any integration. Simply show that the posterior matches the functional form of the inverse-gamma, and then you know the normalization (if you need it) by comparison to the pdf given.

   b) If a conjugate prior exists, then the data can be replaced with sufficient statistics. Can you explain why?

2. **Gaussian processes with non-zero mean:**

   In the lectures we assumed that the prior over any vector of function values was zero mean: $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, K)$. We focussed on the covariance or kernel function $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$, which evaluates the K_{ij} elements of the covariance matrix (also called the 'Gram matrix').

   If we know in advance that the distribution of outputs should be centered around some other mean $\mathbf{m}$, we *could* put that into the model. Instead, we usually subtract the known mean $\mathbf{m}$ from the $\mathbf{y}$ data, and just use the zero mean model.

   Sometimes we don't really know the mean $\mathbf{m}$, but look at the data to estimate it. A fully Bayesian treatment puts a prior on $\mathbf{m}$ and, because it's an unknown, considers all possible values when making predictions. A flexible prior on the mean vector could be another Gaussian process(!). Our model for our noisy observations is now:

   $$\begin{aligned} \mathbf{m} &\sim \mathcal{N}(\mathbf{0}, K_m), \quad K_m \text{ from kernel function } k_m, \\ \mathbf{f} &\sim \mathcal{N}(\mathbf{m}, K_f), \quad K_f \text{ from kernel function } k_f, \\ \mathbf{y} &\sim \mathcal{N}(\mathbf{f}, \sigma_n^2 \mathbb{I}), \quad \text{noisy observations.} \end{aligned}$$

   Show that — despite our efforts — the function values $\mathbf{f}$ still come from a function drawn from a zero-mean Gaussian process (if we marginalize out $\mathbf{m}$). Identify the covariance function of the zero-mean process for f.

1. Numerical libraries often come with a `gammaln` or `lgamma` function to evaluate the log of the gamma function.

Identify the mean's kernel function k_m for two restricted types of mean: 1) An unknown constant $m_i = b$, with $b \sim \mathcal{N}(0, \sigma_b^2)$. 2) An unknown linear trend: $m_i = m(\mathbf{x}^{(i)}) = \mathbf{w}^\top \mathbf{x}^{(i)} + b$, with Gaussian priors $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbb{I})$, and $b \sim \mathcal{N}(0, \sigma_b^2)$.

Sketch three typical draws from a GP prior with kernel:

$$k(x^{(i)}, x^{(j)}) = 0.1^2 \exp\big(-(x^{(i)} - x^{(j)})^2/2\big) + 1.$$

Hints in footnote[2].

3. **Some computation with probabilities:** It's common to compute with log-probabilities to avoid numerical underflow. Quick example: notice that the probability of 2000 coin tosses, 2^{-2000}, underflows to zero in Matlab, NumPy, or any package using IEEE 64 bit floating point numbers.

   If we have two possible models, M_1 and M_2, for some features, we can define:

$$a_1 = \log P(\mathbf{x} \mid M_1) + \log P(M_1) \tag{3}$$
$$a_2 = \log P(\mathbf{x} \mid M_2) + \log P(M_2). \tag{4}$$

   Up to a constant, these 'activations' give the log-posterior probabilities that the model generated the features. Show that we can get the posterior probability of model M_1 neatly with the logistic function:

$$P(M_1 \mid \mathbf{x}) = \sigma(a_1 - a_2) = \frac{1}{1 + \exp(-(a_1 - a_2))}. \tag{5}$$

   Now given K models, with $a_k = \log[P(\mathbf{x} \mid M_k)\, P(M_k)]$, show:

$$\log P(M_k \mid \mathbf{x}) = a_k - \log \sum_{k'} \exp a_{k'}. \tag{6}$$

   The $\log \sum \exp$ function occurs frequently in the maths for probabilistic models (not just model comparison). Show that:

$$\log \sum_k \exp a_k = \max_k a_k + \log \sum_k \exp\Big(a_k - \max_{k'} a_{k'}\Big). \tag{7}$$

   Explain why the expression is often implemented this way. (Hint: consider what happens when all the a_k's are less than -1000).

4. **Bonus question (if you don't do it now, it would be useful to review it later): Preprocessing for Bayesian linear regression and Gaussian processes:**

   We have a dataset of inputs and outputs $\{\mathbf{x}^{(n)}, y^{(n)}\}_{n=1}^N$, describing N preparations of cells from some lab experiments. The output of interest, $y^{(n)}$, is the fraction of cells that are alive in preparation n. The first input feature of each preparation indicates whether the cells were created in lab `A`, `B`, or `C`. That is, $x_1^{(n)} \in \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$. The other features are real numbers describing experimental conditions such as temperature and concentrations of chemicals and nutrients.

   a) Describe how you might represent the first input feature and the output when learning a regression model to predict the fraction of alive cells in future preparations from these labs. Explain your reasoning.

   b) Compare using the lab identity as an input to your regression (as you've discussed above), with two baseline approaches: i) Ignore the lab feature, treat the data from

2. The covariances of two Gaussians add, so think about the two Gaussian processes that are being added to give this kernel. You can get the answer to this question by making a tiny tweak to the Gaussian process demo code provided with the class notes.

all labs as if they came from one lab; ii) Split the dataset into three parts one for lab `A`, one for `B`, and one for `C`. Then train three separate regression models.

Discuss both simple linear regression and Gaussian process regression. Is it possible for these models, when given the lab identity as in a), to learn to emulate either or both of the two baselines?

c) There's a debate in the lab about how to represent the other input features: log-temperature or temperature, and temperature in Fahrenheit, Celsius or Kelvin? Also whether to use log concentration or concentration as inputs to the regression. Discuss ways in which these issues could be resolved.

Harder: there is a debate between two different representations of the output. Describe how this debate could be resolved.