ROB313: Introduction to Learning from Data
University of Toronto Institute for Aerospace Studies

Assignment 4 (12 pts)

Due April 16, 2021, 23:59 EST

Q1) 8pts Perform Bayesian inference for a logistic regression model with a Bernoulli likelihood (as we had considered in assignment 3), and with a zero-centered, uncorrelated Gaussian prior on the weights, as follows:

$$\Pr(y|\mathbf{w},\mathbf{x}) = \big[\widehat{f}(\mathbf{x};\mathbf{w})\big]^{y}\big[1-\widehat{f}(\mathbf{x};\mathbf{w})\big]^{1-y},$$
$$\Pr(\mathbf{w}) = \prod_{i=0}^{D}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\Big(-\frac{w_i^2}{2\sigma^2}\Big) = \mathcal{N}\big(\mathbf{w}\big|\mathbf{0},\sigma^2\mathbf{I}\big),$$

where $\widehat{f}(\mathbf{x};\mathbf{w}) = \Pr(y{=}1|\mathbf{w},\mathbf{x})$ gives the class conditional probability of class 1 by mapping $\mathbb{R}^D \to [0,1]$, and $\mathbf{w} = \{w_0, w_1, \ldots, w_D\} \in \mathbb{R}^{D+1}$. Also, $\widehat{f}$ is a logistic sigmoid acting on a linear model as follows

$$\widehat{f}(\mathbf{x};\mathbf{w}) = \mathsf{sigmoid}\Big(w_0 + \sum_{i=1}^{D} w_i x_i\Big),$$

where $\mathsf{sigmoid}(z) = \frac{1}{1+\exp(-z)}$. Making the assumption that all training examples are *i.i.d.*, the log-likelihood, and log-prior, along with their gradient (∇) and hessian (∇^2) can be written as follows

$$\log\Pr(\mathbf{y}|\mathbf{w},\mathbf{X}) = \sum_{i=1}^{N} y^{(i)}\log\big(\widehat{f}(\mathbf{x}^{(i)};\mathbf{w})\big) + \big(1-y^{(i)}\big)\log\big(1-\widehat{f}(\mathbf{x}^{(i)};\mathbf{w})\big),$$
$$\nabla\log\Pr(\mathbf{y}|\mathbf{w},\mathbf{X}) = \sum_{i=1}^{N}\big[y^{(i)} - \widehat{f}(\mathbf{x}^{(i)};\mathbf{w})\big]\ \bar{\mathbf{x}}^{(i)},$$
$$\nabla^2\log\Pr(\mathbf{y}|\mathbf{w},\mathbf{X}) = \sum_{i=1}^{N}\widehat{f}(\mathbf{x}^{(i)};\mathbf{w})\ \big[\widehat{f}(\mathbf{x}^{(i)};\mathbf{w})-1\big]\ \bar{\mathbf{x}}^{(i)}\bar{\mathbf{x}}^{(i)T}$$
$$\log\Pr(\mathbf{w}) = -\frac{D+1}{2}\log(2\pi) - \frac{D+1}{2}\log(\sigma^2) - \sum_{i=0}^{D}\frac{w_i^2}{2\sigma^2},$$
$$\nabla\log\Pr(\mathbf{w}) = -\frac{\mathbf{w}}{\sigma^2}$$
$$\nabla^2\log\Pr(\mathbf{w}) = -\frac{1}{\sigma^2}\mathbf{I}$$

where $\bar{\mathbf{x}}^{(i)} = \big\{1, x_1^{(i)}, \ldots, x_D^{(i)}\big\}^T \in \mathbb{R}^{D+1}$. All studies will be done on the `iris` dataset, with the training and validation sets merged, and considering only the second response to determine whether the flower is an *iris versicolour*, or not[1]. You are encouraged to re-use code from previous assignments where possible.

[1] Use `x_train, x_test = np.vstack((x_train, x_valid)), x_test` and `y_train, y_test = np.vstack((y_train[:,(1,)], y_valid[:,(1,)])), y_test[:,(1,)]`

(a) **(4pts)** Consider the prior variances $\sigma^2 = 0.5$, $\sigma^2 = 1$, and $\sigma^2 = 2$. Which of these priors gives a model with a higher complexity? Explain why. Choose between these prior variances by approximating the log marginal likelihood using a Laplace approximation. Report your results.

(b) **(4pts)** Choose a proposal distribution and use importance sampling to estimate the most probable predictive posterior class on each element of the test set using a prior variance of $\sigma^2 = 1$. Report your test set accuracy results and justify your chosen proposal. Also, analyze and comment on the accuracy of your proposal distribution. It may help to visualize the values of the posterior evaluated at your samples to help justify your answer. You may find the `scipy.stats` module useful for sampling.

Q2) 4pts Construct a Bayesian linear model on the `mauna_loa` dataset. Consider the following model

$$\Pr(\mathbf{y}|\mathbf{w}, \mathbf{X}) = \mathcal{N}\big(\mathbf{y}\big|\mathbf{\Phi w},\ \sigma^2\mathbf{I}\big)$$
$$\Pr(\mathbf{w}) = \mathcal{N}\big(\mathbf{w}\big|\mathbf{0},\ \mathbf{I}\big),$$

where $\sigma^2 = 10^{-4}$ and $\mathbf{\Phi} \in \mathbb{R}^{N\times M}$ contains the M features at all N training examples defined by the following function (please copy this code and use it in your solution)

```
import numpy as np
def features(x):
    """
    evaluates phi(x)

    Inputs:
        x : (N, 1) input datapoints
    Outputs:
        phi : (N, M) features for each datapoint
    """
    year = 0.057 # equal to one year in input space
    phi = np.hstack(
        # add polynomial terms
        [np.power(x, np.arange(11))]
        # add periodic terms:
        + [np.sin(x*2*np.pi*factor/year) for factor in range(1,11)]
        + [np.cos(x*2*np.pi*factor/year) for factor in range(1,11)]
    )
    return phi
```

The basis functions were designed for the `mauna_loa` dataset by observing that the response appears to have a periodic annual sawtooth pattern on top of a smooth multi-year curve. To model the long-term multi-year curve, polynomial features up to order ten are included and to model the periodic pattern, sine and cosine features are included with a yearly period, or an integer fraction of a yearly period (to enable recovery of something like a Fourier transform of the observed sawtooth pattern).

Use both the training and validation sets to predict on the test set. Plot the predictive posterior relative to the test data: plot the predictive posterior mean as a line and plot the 99.7% confidence interval (the predictive posterior mean $\pm$ three standard

deviations) as a shaded region[2]. On the same plot, also show the test data and discuss the quality of the predictive posterior. How might you improve this model? Use the Cholesky decomposition for the predictive posterior computations.

Submission guidelines: Submit an **electronic copy** of your report in **pdf** format, and **documented** python scripts. You should include a file named "README" outlining how the scripts should be run. Upload a single tar or zip file containing all files to Quercus. You are expected to verify the integrity of your tar/zip file before uploading. Do not include (or modify) the supplied *.npz data files or the data_utils.py module in your submission. The report must contain

- Objectives of the assignment
- A brief description of the structure of your code, and strategies employed
- Relevant figures, tables, and discussion

Do not use scikit-learn for this assignment, the intention is that you implement the simple algorithms required from scratch. Also, for reproducibility, always set a seed for any random number generator used in your code. For example, you can set the seed in numpy using `numpy.random.seed`

[2]Use the function `matplotlib.pyplot.fill_between` to plot the shaded region. Use the keyword argument `alpha=0.3` to give some transparency to the shaded region.