# ISTA 421/521 – Homework 4

**Due: Monday, October 29, 5pm**

20 pts total for Undergrads, 25 pts total for Grads

STUDENT NAME

Undergraduate / Graduate

## Instructions

In this assignment, exercises 3 and 4 require you to write small python scripts; the details for those scripts, along with their .py name are described in the exercises. All of the exercises in this homework require written derivations, so you will also submit a .pdf of your written answers. (You can use LaTeX or any other system (including handwritten; plots, of course, must be program-generated) as long as the final version is in PDF.)

**The final submission will include (minimally) the two scripts you need to write for problems 3 and 4, and a PDF version of your written part of the assignment. You are required to create either a .zip or tarball (.tar.gz / .tgz) archive of all of the files for your submission and submit your archive to the d2l dropbox by the date/time deadline above.**

NOTE: Problem 1 is required for Graduate students only; Undergraduates may complete this problem for extra credit equal to the point value.

(FCMA refers to the course text: Rogers and Girolami (2016), *A First Course in Machine Learning*, second edition. For general notes on using LaTeX to typeset math, see:
`http://en.wikibooks.org/wiki/LaTeX/Mathematics`)

1. [5 points; **Required only for Graduates**] Adapted from **Exercise 3.12** of FCMA p.135:

   When performing a Bayesian analysis of the Olympics data, we assumed that $\sigma^2$ was known. If instead we assume that $\mathbf{w}$ is known and an inverse Gamma prior is placed on $\sigma^2$,

   $$p(\sigma^2|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\alpha-1}\exp\left\{-\frac{\beta}{\sigma^2}\right\},$$

   then the posterior over $\sigma^2$ will also be inverse Gamma. Derive the parameters for the posterior belief in the variance.

   **Solution.**

2. [6 points] Adapted from **Exercise 4.2** of FCMA p.163:

   In Chapter 3, we computed the posterior density over $r$, the probability of a coin giving heads, using a beta prior and a binomial likelihood. Recalling that the beta prior, with parameters $\alpha$ and $\beta$, is given by

   $$p(r|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}r^{\alpha-1}(1-r)^{\beta-1}$$

   and the binomial likelihood, assuming $y$ heads in $N$ throws, is given by

   $$p(y|r,N) = \binom{N}{y}r^y(1-r)^{N-y},$$

   **compute the Laplace approximation to the posterior**. (Note, you should be able to obtain a closed-form solution for the MAP value, $\hat{r}$, by getting the log posterior, differentiating (with respect to $r$), equating to zero and solving for $r$.)

   **Solution.**

3. [4 points] Adapted from **Exercise 4.3** of FCMA p.163:

   In the previous exercise you computed the Laplace approximation to the true beta posterior. In this problem, plot both the true beta posterior and the Laplace approximation for the following three parameter settings:

   1. $\alpha = 5$, $\beta = 5$, $N = 20$, and $y = 10$,
   2. $\alpha = 3$, $\beta = 15$, $N = 10$, and $y = 3$,
   3. $\alpha = 1$, $\beta = 30$, $N = 10$, and $y = 3$.

   Be sure to clearly indicate the values in your plot captions. Include how the two distributions (the true beta posterior and the Laplace approximation) compare in each case. Include the python script you use to generate these plots; the script should be named `plot_laplace_approx.py`. **Suggestion**: for plotting the beta and Gaussian (Normal) distributions, you can use `scipy.stats.beta` and `scipy.stats.normal` to create the beta and Gaussian random variables, and use the `pdf(x)` method for each to generate the curves. Note that for `scipy.stats.normal`, the mean is the location (`loc`) parameter, and the sigma is the `scale` parameter. Also, `scipy.stats.normal` expects the scale parameter to be the standard deviation (i.e., take the square root: `math.sqrt(x)`) of the variance you'll compute for the Laplace approximation.

   **Solution.**

4. [4 points] Adapted from **Exercise 4.4** of FCMA p.164:

   Given the expression for the area of a circle, $A = \pi r^2$, and *using only uniformly distributed random variates*, devise a sampling approach for estimating $\pi$. Describe your method in detail and provide your

script to do the estimation – this script should be called `pi_sample_estimate.py`. Report your estimate based on 1 million samples to 6 decimal places. (NOTE: You do *not* need to use Metropolis-Hastings to compute this.)

**Solution.**

5. [6 points] Adapted from **Exercise 4.6** of FCMA p.164:

   Assume that we observe $N$ vectors of attributes, $\mathbf{x}_1, ..., \mathbf{x}_N$, and associated integer counts $t_1, ..., t_N$. A Poisson likelihood would be suitable:

   $$p(t_n|\mathbf{x}_n, \mathbf{w}) = \frac{f(\mathbf{x}_n; \mathbf{w})^{t_n} \exp\{-f(\mathbf{x}_n; \mathbf{w})\}}{t_n!},$$

   where $f(\mathbf{x}_n; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}_n$. Assuming a zero-mean Gaussian prior on $\mathbf{w}$ with constant diagonal covariance of $\sigma^2$, derive the gradient and Hessian of the posterior. Using these, express the parameter update rules for (a) gradient *ascent* (because we're maximizing) update (in class we looked at Widrow-Hoff, which is typically expressed for *descent*), and (b) Newton-Raphson.

   The following facts will help in the derivation. First, keep in mind that although $\mathbf{w}$ and $\mathbf{x}_n$ are vectors (of the same dimension), their dot product, $\mathbf{w}^\top \mathbf{x}_n$, is a *scalar* value. This means you can take the partial derivative of $\log \mathbf{w}^\top \mathbf{x}_n$ with respect to $\mathbf{w}$. Also, remember that the Hessian is a matrix representing the second partial derivatives of the gradient with respect to itself (see Comment 2.6 of p.73), and the second derivative will involve the *transpose* of the partial derivative with respect to $\mathbf{w}$. So, e.g., as part of taking the second derivative, if you are taking the transpose derivative part of $\mathbf{w}^\top \mathbf{x}_n$, as follows:

   $$\frac{\partial(\mathbf{w}^\top \mathbf{x}_n)}{\partial \mathbf{w}} = \mathbf{x}_n \quad \text{and} \quad \frac{\partial(\mathbf{w}^\top \mathbf{x}_n)}{\partial \mathbf{w}^\top} = \mathbf{x}_n^\top$$

   **Solution.**