ISTA 421/521 – Homework 5

Due: Wednesday, November 17, 5pm

25 pts total for Undergrads and Grads

STUDENT NAME

Undergraduate / Graduate

Instructions

In this assignment, exercises 2 and 3 require you to write small python scripts; the details are provided in those 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.)

NOTE: In this assignment there are no separate "Graduate" exercises.

As in previous homework, pytest "unit tests" are provided to help guide your progress.

You may work with others in the course on the homework. However, if you do, you. **must** list he names of everyone you worked with, along with which problems you collaborated. Your final submissions of code and written answers **MUST ALL BE IN YOUR OWN CODE FORMULATION AND WORDS**; you cannot submit copies of the same work – doing so will be considered cheating.

(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. [8 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 α and β, 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.

2. [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 exercise, you will implement the calculation of the posterior Beta parameters and the computation of the Laplace approximation parameters in the function `plot_laplace_approx` in the provided file `laplace_approx.py`. Once implemented, running the script (set `RUN_LAPLACE_APPROX` to `True`) will 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 $y = 3$.

   This will produce three separate plots (each with the Beta and associated Laplace approximation). Include these in your written answer and be sure to clearly indicate in your plot captions the parameter values that were used to generate the curves in the plot. In your written answer, also include how the two distributions (the true Beta posterior and the Laplace approximation) compare in each case. NOTE: `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 compute for the Laplace approximation.

   Solution.

3. [5 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*, that is, using only uniform random number draws computed by python's `random.uniform`, devise a sampling approach for estimating π. (Just to state the obvious: you are not allowed to use any other direct numeric representation of π, such as `math.pi`, in your solution.) In your written answer, describe your method in detail. Implement your method in the function `estimate_pi` in `pi_sample_estimate.py`. This function takes a single argument, N, representing the number of samples to be used in your estimate. In your written answer, give your estimate based on 1 million samples to 6 decimal places. The provided unit test expects that your estimate will be within 0.005 of π (more specifically, the approximation of `math.pi`) when based on $1,000,000$ (one million) samples. Since this is a sampling method, it is possible for a good solution to this exercise to still get an unlikely sample that is outside of 0.005 of π, however this is expected to be rare; with a good solution, you should almost always pass the unit test. (NOTE: You do *not* need to use Metropolis-Hastings to compute this.)

Solution.

4. [8 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 σ^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 $\mathbf{w}$. Second, here's a little more detail on taking the vector second derivative to get the Hessian. 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 $\mathbf{w}$. That is, first you will take the partial with respect to $\mathbf{w}$ to get the gradient (used for the first part of the exercise, for deriving the gradient ascent update) then you'll take the derivative of the gradient with respect to $\mathbf{w}^\top$. Taking the derivative with respect to $\mathbf{w}^\top$ works pretty much like taking the derivative with respect to $\mathbf{w}$, except the result will itself be transposed. The following shows examples of taking the derivative of $\mathbf{w}^\top \mathbf{x}_n$ with respect to $\mathbf{w}$ and separately with respect to $\mathbf{w}^\top$:

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

   In this exercise, when you're computing the Hessian, you'll be taking the *second* partial derivative with respect to $\mathbf{w}$ of the product of the prior and likelihood (let's call that $g(\mathbf{w})$), first taking the partial with respect to $\mathbf{w}$ and then $\mathbf{w}^\top$:

$$\frac{\partial^2 g(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^\top}$$

Solution.