ISTA 421/521 – Homework 3

Due: Sunday, October 15, 8pm

24 pts total for Undergrads, 30 pts total for Grads

STUDENT NAME

Undergraduate / Graduate

Instructions

In this assignment you are required to modify/write 4 scripts in python. Details of what this will involve are specified in exercises 1, 2, 5 and 6, below.

Included in the homework 3 release are following files in `code`:

- `poisson.py` - This script will be used in Exercise 1.
- `random_generator.py` - This script implements a class, `RNG`, that will simulate a random number generator, but where the random numbers are read from file. This is an attempt to make it so that all random number draws will be uniform across platforms. This will be used in Exercise 2.
- `approx_expected_value.py` - This script includes a demonstration of how to approximate an expected value through sampling. You will modify this code and submit your solution for Exercise 2.
- `predictive_variance.py` - This script contains code relevant to exercises 5 and 6. You will need to fill in two functions (as described in Exercise 5) as well as `exercise_6`
- `gauss_surf.py` - This is provided for fun – it is not required for any exercise here. It generates a 2d multivariate Gaussian and plots it as both a contour and surface plot. This provides an example of how to code making use of numpy array-computation; in particular, look at how the "mesh" of points in 2-d surface is computed simultaneously on lines 84-95 (see comments).
- `w_variation_demo.py` - This script is also provided for fun and is not required for the assignment. (It also provides more example python code!) This implements the simulated experiment demonstrating the theoretical and empirical bias in the estimate, $\widehat{\sigma^2}$, of the model variance, σ^2, as a function of the sample size used for estimation.

All exercises except exercise 1 require that you provide some "written" answer (in some cases also figures), 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.

As with the previous homework, the final submitted PDF written answers must be named `hw3-answers.pdf`.

NOTE: Exercises 3 and 7 are required for Graduate students only; Undergraduates may complete them for extra credit equal to the point value.

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

NOTE: For the unit test for Exercise 5c,
`code/test_ex5c_cov_w.py::test_ex5c_plot_functions_sampling_from_covw`
it is expected that you will get the following warning (line number could vary):

```
hw3_solution/code/predictive_variance.py:344:
RuntimeWarning: covariance is not symmetric positive-semidefinite.
```

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.

(FCML 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. [3 points] Adapted from **Exercise 2.3** of FCML:

   Let Y be a random variable that can take any non-negative integer value. The likelihood of these outcomes is given by the Poisson pmf (probability mass function):

   $$P(y) = \frac{\lambda^y}{y!} e^{-\lambda} \tag{1}$$

   By using the fact that for a discrete random variable the pmf gives the probabilities of the individual events occurring and the probabilities are additive, fill in the two functions in `poisson.py` as follows:

   (a) In `calculate_posson_pmf_a` compute the probability that $Y \geq 2$ and $Y \leq 6$ for $\lambda = 3$, i.e., $P(2 \leq Y \leq 6)$, and assign that to the return value `probability`.

   (b) In `calculate_posson_pmf_b`, using the fact that one outcome has to happen, compute the `probability` that $Y < 2$ or $Y > 6$ (again, for $\lambda = 3$).

   You are only allowed to use the python `math` package; no other packages are allowed.

   There is no need for a written answer to this exercise, only your code submission.

2. [4 points] Adapted from **Exercise 2.4** of FCML:

   Let X be a random variable with uniform density, $p(x) = \mathcal{U}(a, b)$.

   Work out analytically $\mathbf{E}_{p(x)} \left\{35 + 3x - 3x^2 + 0.2x^3 + 0.01x^4\right\}$ for $a = -1$, $b = 9$ (show the steps).

   The script `approx_expected_value.py` includes a function that demonstrates how you can use uniform random samples to approximate an expectation, as described in Section 2.5.1 of FCML. The script estimates the expectation of the function y^2 when $Y \sim \mathcal{U}(0, 1)$ (that is, Y is uniformly distributed between 0 and 1). This script generates a plot of how the estimation improves as larger samples are considered, up to 10,000 samples. NOTE: the script uses `random_generator.py`, which emulates a uniform random number generator interface but uses a fixed sequence of random numbers (`data/rand_uniform_10000.txt`); this ensures that correct solutions should be exact across platforms.

   Fill in the function `exercise_2` in `approx_expected_value.py` to compute a sample-based approximation to the expectation of the function $35 + 3x - 3x^2 + 0.2x^3 + 0.01x^4$ when $X \sim \mathcal{U}(-1, 9)$ and observe how the approximation improves with the number of samples drawn. In your written answer, include the generated plot `ex2_fn_approx.png` showing the evolution of the approximation, relative to the true value, over $10,000$ samples. Include a description of the trend you see in the plot.

   Solution.

3. [2 points; **Required only for Graduates**] Adapted from **Exercise 2.6** of FCMA:

   Assume that $p(\mathbf{w})$ is the Gaussian pdf for a D-dimensional vector $\mathbf{w}$ given in

   $$p(\mathbf{w}) = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{w} - \mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{w} - \mu)\right\}. \tag{2}$$

Suppose we use a diagonal covariance matrix with different elements on the diagonal, i.e.,

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_D^2 \end{bmatrix}$$

Does this assume independence of the D elements of $\mathbf{w}$? If so, show how by expanding the vector notation of Eqn. 2 and re-arranging. You will need to be aware that the determinant of a matrix that only has entries on the diagonal is the product of the diagonal values and that the inverse of the same matrix is constructed by simply inverting each element on the diagonal. (Hint, a product of exponentials can be expressed as an exponential of a sum. Also, just a reminder that $\exp\{x\}$ is e^x.)

Solution.

4. [4 points] Adapted from **Exercise 2.9** of FCML:

   Assume that a dataset of N binary values, $x_1, ..., x_N$, was sampled from a Bernoulli distribution, and each sample x_i is independent of any other sample. Explain why this is *not* a Binomial distribution. Derive the maximum likelihood estimate for the Bernoulli parameter of this distribution.

   Solution.

5. [8 points] Adapted from **Exercise 2.12** of FCML:

   Familiarize yourself with the provided script `predictive_variance.py`. It is mostly implemented, but you will have to fill in the details for two functions:

   - `calculate_prediction_variance`, which calculates the *variance* for a prediction at x_{new} given the design matrix, $\mathbf{X}$, the estimated parameters, $\hat{\mathbf{w}}$, and target responses, $\mathbf{t}$.
   - `calculate_cov_w`, which calculates the estimated covariance of $\hat{\mathbf{w}}$ given the design matrix, $\mathbf{X}$, the estimated parameters, $\hat{\mathbf{w}}$, and target responses, $\mathbf{t}$.

   Once implemented, then you can run the script.

   When you run the script, it will generate a dataset based on a function (implemented in `true_function`) and then remove all values for which $2.5 \leq x \leq 4.5$. Three groups of plots will be generated:

   (a) First is a plot of the data (this will be generated by Part 5a of the script).

   (b) Next, the script will plot the error bar plots for predictions of values for model orders 1, 3, 5 and 9 (this will be generated by Part 5b).

   (c) Finally, in Part 5c, the script samples model parameters $\hat{\mathbf{w}}$ from the covariance $\text{cov}(\hat{\mathbf{w}})$ and plots the resulting functions (again, for model orders 1, 3, 5 and 9).

   In total, you will plot 9 figures. You must include the plots in your written submission and do the following: Include a caption for each figure that qualitatively describes what the figure shows; contrast the figures within group (b) with each other. Do the same for group (c). Also, clearly explain what effect removing the points $2.5 \leq x \leq 4.5$ has done in contrast to if they were left in.

   (NOTE: the script `predictive_variance` also contains elements for Exercise 6 (namely, the function `exercise_6`), but you can ignore those components while completing this exercise.)

   Solution.

6. [5 points]

The code for this exercise is also found in `predictive_variance.py`, in the function `exercise_6`. In this exercise, you will fill in the missing pieces as indicated. Once done, running `exercise_6` demonstrates how model bias impacts variance, similar to the demonstration in Lecture 9. Once implemented, the script will generate four plots: a separate plot for each of the model polynomial orders 1, 3, 5 and 9. In your written answer, first describe what the code is doing, in your own words. Then, include the four plots with descriptive captions, and describe what happens to the variance in the functions in the plots as the model order is changed.

Solution.

7. [4 points; **Required only for Graduates**] Adapted from **Exercise 2.13** of FCML:

   Derive the Fisher Information Matrix for the parameter of a Bernoulli distribution.

   Solution.