COMPSCI 590N: Assignment 5

Due: October 18, 2017 at 11:55pm

Included with the assignment is a script for testing your solution called `assignment5_tests.py`. This script will test the output from your code against a number of test cases and will indicate if there are errors. Once you have written your code in `assignment5.py`, you can run these tests by executing:

```
python assignment5_tests.py
```

Be sure that you can run `assignment5_tests.py` before submitting as this is how we will test your code for grading! The provided test cases are meant to help you debug your code, but you should not assume that they are exhaustive. If a problem asks you define a function or class, **you should use exactly the name specified** in the problem for this function or class. Please submit to Moodle a **zip** file containing your modified version of `assignment5.py` as well as a **PDF** containing all plots and question responses.

Problem 1: Inverse Normal CDF

The inverse CDF for the Normal distribution with mean μ and variance σ^2 is given by:

$$F^{-1}(u) = \mu + \sqrt{2\sigma^2}\,\mathrm{erf}^{-1}(2u - 1) \tag{1}$$

where $\mathrm{erf}^{-1}(\cdot)$ is the inverse of the **error function**, $\mathrm{erf}(u) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$. Please implement a function called `inverse_normal_cdf` that takes as arguments a numpy array, `u`, and two floats, `mu` and `sigma2`. `inverse_normal_cdf` should apply the inverse CDF $F^{-1}(\cdot)$ for a Normal distribution with mean `mu` and variance `sigma2` elementwise to `u` and return the result as a numpy array. You should use the `scipy` implemenation of the inverse error function, `scipy.special.errinv`.

Problem 2: Inverse CDF Sampling

Implement a function called `normal_sampler` that takes three arguments: an integer `n_samples` and two floats `mu` and `sigma2`. `normal_sampler` should use inverse CDF

sampling to draw `n_samples` samples from the Normal distribution with mean `mu` and variance `sigma2` and return a numpy array containing the results. You should use your implementation of `inverse_normal_cdf` from Problem 1.

Debugging samplers can be difficult as the results are inherently random. To demonstrate your sampler, you should use it to draw 1,000 samples from a Normal distribution with $\mu = 0$ and $\sigma^2 = 2$ and plot/report a histogram of the samples using either `matplotlib` or `seaborn`.

Problem 3: Monte Carlo Expected Value

Please implement a function called `mcev` that uses your Normal sampler to compute approximate expected values. `mcev` should take as arguments a function object `fun`, an integer `n_samples`, and two floats `mu` and `sigma2`. `mcev` should draw `n_samples` samples from a Normal distribution with mean `mu` and variance `sigma2` and use these samples to compute an approximate expected value of `fun`. You may assume that `fun` takes and returns a single float. For example, if `fun` implements x^2,

```
def fun(x):
    return x**2
```

then you could approximate $\mathbb{E}[X^2]$ under a standard Normal (mean 0 and variance 2) using 1,000 samples by calling,

```
>>> mcev(fun,1000,0.0,2.0)
1.999031006917
```