Homework 1 – Due February 25, 12 am CST

1. Let $X_1, X_2, \ldots, X_n$ be a sample from the standard uniform $U(0,1)$ distribution. Find the distribution of $(\sum_{i=1}^{n} X_i) \text{mod} 1$. [*10 points*]

2. Let F be a cumulative distribution function (c.d.f.). Define $F^{[-1]}(u) = min\{x \mid F(x) \geq u\}$. Show that if $U \sim U(0,1)$, then $F^{[-1]}(U) \sim F$. [*5 points*]

3. There are many methods of Sampling from the Standard Normal distribution.

   (a) Prove that the Box-Muller algorithm provides two independent standard normal random deviates. [*5 points*]

   (b) Prove that the Polar algorithm for simulating from the standard normal distribution provides two independent standard normal random deviates. [*5 points*]

4. *Sampling from the tail of a standard normal.* Perform a simulation experiment in R (or any other software) to map the rejection rate with increasing d. Compare the result with the theoretical naive rejection rate. (For this experiment, no code is required to be turned in, but you are required to describe the experiment that you set up, and also detail the results, using appropriate figures (and tables) if needed.) [*10 points*]

5. *Compressed Column Storage.* Consider the $n \times p$ sparse matrix $\boldsymbol{W} = ((w_{ij})$ with m non-zero entries and stored in the following *Compressed Column Storage (CCS)* format:

   - Consider the triple given by $\boldsymbol{A} = (\boldsymbol{a}, \boldsymbol{r}, \boldsymbol{c})$. where $\boldsymbol{a} = (a_1, a_2, \ldots, a_m)$ is a m-dimensional vector containing the non-zero values in $\boldsymbol{W}$ (stored column-wise), $\boldsymbol{r} = (r_1, r_2, \ldots, r_m)$ is the m-dimensional vector with r_i storing the row index of the corresponding a_i, and $\boldsymbol{c} = (c_1, c_2, \ldots, c_p)$ is a $(p+1)$-dimensional vector with c_i indicating the element of $\boldsymbol{a}$ that starts the column of $\boldsymbol{A}$. By convention, $c_{p+1} = m + 1$. Thus, $a_i \equiv w_{r_i,k}$, where $c_k \leq i < c_{k+1}$. Matrices may alternatively be stored in the Compressed Row Storage format (CRS) defined correspondingly.

   (a) Consider the product of $\boldsymbol{A}$ with a p-variate vector $\boldsymbol{x}$. Let $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$. Provide an algorithm for calculating the j element of $\boldsymbol{y}$. That is, write an algorithm which will calculate y_j given $\boldsymbol{W}$ in CCS format $\boldsymbol{A} = (\boldsymbol{a}, \boldsymbol{r}, \boldsymbol{c})$ *and without expanding it back to* $\boldsymbol{W}$. No programming is required, but only the algorithm in pseudo-code should be provided. [*10 points*]

   (b) Suppose that we now have a sparse symmetric $p \times p$ matrix $\boldsymbol{V}$. Answer the following questions:

      i. Develop a similar CCS-type strategy for storing the matrix $\boldsymbol{V}$ in a *packed CCS* format. [*5 points*]

      ii. Provide, in pseudo-code, an algorithm for finding the jth element y_j of $\boldsymbol{y} = \boldsymbol{V}\boldsymbol{x}$. [*5 points*]

6. Let $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_{n_U}$ be a sample from Population 1 and $\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_{n_V}$ be a sample from Population 2. Consider the following samples hypothesized (but not known for sure) to be from different populations a follows: $\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_{n_W}$ from Population 1, $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{n_X}$ to be from Population 2, $Y_1, Y_2, \ldots, Y_{n_Y}$ from Population 3, $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_{n_Z}$ from Population 4. The total within sum of squares (WSS) in this scenario is given by

$$\begin{aligned}\sum_{i=1}^{n_U}\|\boldsymbol{U}_i - \hat{\boldsymbol{\mu}}_1\|^2 + \sum_{i=1}^{n_V}\|\boldsymbol{V}_i - \hat{\boldsymbol{\mu}}_2\|^2 + \sum_{i=1}^{n_W}\|\boldsymbol{W}_i - \hat{\boldsymbol{\mu}}_1\|^2 + \sum_{i=1}^{n_X}\|\boldsymbol{X}_i - \hat{\boldsymbol{\mu}}_2\|^2 \\ + \sum_{i=1}^{n_Y}\|\boldsymbol{Y}_i - \hat{\boldsymbol{\mu}}_3\|^2 + \sum_{i=1}^{n_Z}\|\boldsymbol{Z}_i - \hat{\boldsymbol{\mu}}_4\|^2,\end{aligned} \tag{1}$$

where $\hat{\boldsymbol{\mu}}_1 = (n_U\bar{\boldsymbol{U}} + n_W\boldsymbol{W})/(n_U + n_W)$, $\hat{\boldsymbol{\mu}}_2 = (n_V\bar{\boldsymbol{V}} + n_X\bar{\boldsymbol{X}})/(n_V + n_X)$, $\hat{\boldsymbol{\mu}}_3 = \bar{\boldsymbol{Y}}$ and $\hat{\boldsymbol{\mu}}_4 = \bar{\boldsymbol{Z}}$. The above total WSS can be rewritten to be a sum of WSS from the samples $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_{n_U}$ and $\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_{n_V}$ plus some terms. (This is important in the context of the following questions.) We are interested in finding how the total WSS changes when we reassign an (unsure) observation from one population to the other. To do so, we will investigate the three possibilities (there is actually a fourth, but that is the converse of Case (b) below).

   (a) Suppose we reassign $\boldsymbol{W}_{n_W}$ from Population 1 to Population 2. How does the total WSS in (1) change? (*Note that such a reassignment affects both* $\hat{\boldsymbol{\mu}}_1$ *and* $\hat{\boldsymbol{\mu}}_2$). [*10 points*]

   (b) Suppose that from the original setup leading to (1). we reassign $\boldsymbol{W}_{n_W}$ from Population 1 to Population 3. How does the total WSS in (1) change? (*Note that such a reassignment affects both* $\hat{\boldsymbol{\mu}}_1$ *and* $\hat{\boldsymbol{\mu}}_3$). [*10 points*]

   (c) Suppose that from the original setup behind (1). we reassign $\boldsymbol{Y}_{n_Y}$ from Population 3 to Population 4. How does the total WSS in (1) change? (*Note that such a reassignment affects both* $\hat{\boldsymbol{\mu}}_3$ *and* $\hat{\boldsymbol{\mu}}_4$. *Note that an extension of this is the basis for the Hartigan-Wong (1979) implementation of the* k*-means algorithm.*) [*10 points*]

   The reductions in all three parts above provides a stripped-down context of the basis for an efficient implementation of the k-means semi-supervised clustering algorithm.

7. For each program, please write out how you compiled, executed the program and result. You may include this code in as a comment.

   (a) Write an annotated C program which converts temperature, from Fahrenheit to the Celsius scale and vice-versa. [*7 points*]

   (b) Write an annotated C program which takes in two integers $i = 320$ and $j = 256$ and reports their product. Store the product as a `short` and as an `int` and print the result. What differences do you see? Please put in your observations and reasoning as a comment in the C program. [*8 points*]