Homework 1 – ECE 601: Machine Learning for Engineers
Due Date: Tuesday Feb 9, 11:59 pm

This homework is intended to help you review your linear algebra and learn (or refresh your understanding of) how to implement linear algebraic operations in Python using `numpy` (to which we refer in the code below as `np`) and `pandas` (to which we refer in the code below as `pd`).

For the `numpy` each of the problems below, write a method (e.g., `problem1`) that returns the answer for the corresponding problem. Put all your methods in one file called `homework1_UMassUSERNAME.py` (e.g., `homework1_aghasemi.py`). See the starter file `homework1_template.py`. In all problems, you may assume that the dimensions of the matrices and/or vectors are compatible for the requested mathematical operations. **Note**: Throughout the assignment, please use `np.array`, not `np.matrix`.

Homework Problems

1. Given matrices A and B, compute and return an expression for $A + B$. [1 pt]

   Answer (freebie!): While it is completely valid to use `np.add(A, B)`, this is unnecessarily verbose; you really should make use of the "syntactic sugar" provided by Python's/`numpy`'s operator overloading and just write: $A + B$. Similarly, you should use the more compact (and arguably more elegant) notation for the rest of the questions as well.

2. Given matrices A, B, and C, compute and return $AB - C$ (i.e., right-multiply matrix A by matrix B, and then subtract C). Use `dot` or `np.dot`. [2 pts]

3. Given matrices A, B, and C, return $A \odot B + C^T$, where $\odot$ represents the element-wise (Hadamard) product and T represents matrix transpose. In `numpy`, the element-wise product is obtained simply with $*$. [2 pts]

4. Given matrix A, return a matrix with the same dimensions as A but that contains all zeros. Use `np.zeros`. [2 pts]

5. Given square matrix A and column vector x, use `np.linalg.solve` to compute $A^{-1}x$. Do not explicitly calculate the matrix inverse itself (e.g., `np.linalg.inv`, A^{**-1}) because this is numerically unstable (and yes, it can sometimes make a big difference!). [2 pts]

6. Given matrix A and integer i, return the sum of all the entries in the ith row, i.e., $\sum_j A_{ij}$. Do not use a loop, which in Python can be very slow. Instead use the `np.sum` function. [3 pts]

7. Given matrix A and scalars c, d, compute the arithmetic mean over all entries of A that are between c and d (inclusive). In other words, if $S = \{(i, j) : c \leq A_{ij} \leq d\}$, then compute $\frac{1}{|S|} \sum_{(i,j)\in S} A_{ij}$. Use `np.nonzero` along with `np.mean`. [3 pts]

8. Given a n-dimensional column vector x, an integer k, and positive scalars m, s, return an $n \times k$ matrix, each of whose columns is a sample from multidimensional Gaussian distribution $\mathcal{N}(x + mz, sI)$, where z is an n-dimensional column vector containing all ones and I is the identity matrix. Use either `np.random.multivariate_normal` or `np.random.randn`. [4 pts]

9. Given the path to a CSV file as `filepath` (which can be located on your hard drive or Google Drive), read the file, print its dimensions, the first column, and the first row. Use `pd.read_csv`, and `df.iloc` assuming that the name of the read file is df. [3 pts]