**Computer challenges for the neural networks.**

1. Load the file **training\_and\_testing\_values.mat** available in the [Week 11 repository](https://github.com/Mark-Kramer/BU-MA665-MA666/tree/master/Week-11%20Neural%20networks) into Python. You will find that this .mat file contains a dictionary with the following important variables:  
     
   Training Data  
   *x\_training* = x-coordinate to train the network,  
   *y\_training* = y-coordinate to train the network  
   *correct\_answer* = a vector of 0’s and 1’s.  
     
   These three variables define the training data. For each (x,y)-coordinate, the *correct\_answer* indicates whether the point is above (1) or below (0) the line. There are 1000 example (x,y)-cooridates, each with a correct classification.  
     
   Testing Data  
   *x\_testing* = x-coordinate to test the network  
   *y\_testing* = y-coordinate to test the network  
     
   The challenge is to determine whether each (x,y)-coordinate in the testing data is above or below the line. To do so, train a perceptron using the training data. Then, apply your trained perceptron to the testing data.
2. Load the file **training\_and\_testing\_values\_3.mat** available in the [Week 11 repository](https://github.com/Mark-Kramer/BU-MA665-MA666/tree/master/Week-11%20Neural%20networks) into Python. You will find that this .mat file contains a dictionary with the following important variables:  
     
   Training Data  
   *training* is an array with dimensions (4,1000). Each row includes an (x,y,z)-coordinate (columns 0-2), and a classification of this coordinate as either above (1) or below (0) a plane (column 3). There are 1000 example points each with a correct classification.  
     
   Testing Data  
   *testing* = is an array with dimensions (3,1000). Each row includes an (x,y,z)-coordinate (columns 0-2). The classification for these points, as either above or below the plane, is unknown.  
     
   The challenge is to correctly classify each point in the testing data as above or below the plane. To do so, train a perceptron using the training data. Then, apply your trained perceptron to the testing data.