# Homework 2: Part 2

Thuan Tran & Chayse Summers
Data Magnets

## Analysis

Note: for this method, we used the 1,2,3 ordering for Setosa,Versicolor and Verginica
Here is the list of steps for completing this assignment:

1. Load the data set (since I need to refer to the data in it). I can also refer to the data outside of the function, but just to be certain, I load it again. After loading the dataset, separate it into 3 flower types. Combine all three flower matrixes to get 150 observations total.
2. Next, calculate the Covariance Matrix of the new variable that has 150 observations by using the function cov that will create a 4x4 matrix. In addition, since this observation belongs to case 2, I used all of 150 observations together as a single covariance matrix for each class.
3. Then, take the inverse of the new covariance matrix by using the function inv().
4. Depending on ***i*** (the parameter), I used an *if* statement to set an appropriate meanValue. This will create a 1x4 matrix where each column is the average of each features in each of the 50 observations.
5. Note, the following was determined by working backward from the equation, all the way to calculate $w_i$. Since at first, I was having trouble following the formula where the inverse Covariance matrix multiply the meanValue. A 4x4 multiply a 1x4. So I worked backward to determine the appropriate path.
6. Looking at the last equation of g(x), in the book $g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + w_{i0}$ *(linear discriminant)* that create a scalar elements, since $w_{i0}$ is already a scalar (the distance). What is left is the term $w^t$ * $x$ where it will also produce a scalar. X is an 1x4 matrix so to get the result to a scalar, we need something like 1x4 X 4x1. Hence switch the place of x and w.
7. After observing this information, I need to get $w^t$ in the form 4x1. It makes sense since each column of x will be attached to each row of w, hence creating a scalar.
8. This is why I shifted the formula to calculate w as inverseCovariance * transpose(meanValue), where meanValue belong to class i. This is equal to a 4x4 X 4x1 matrix, hence creating a 4x1 matrix.
9. Now, onto the $w_{io}$ (distance from the origin). For this part I follow the original equation, but I let meanValue * inverseCovarianceMatrx * transpose(meanValue). We have A 1x4 X 4x4 X 4x1 is equal to a 1x1. Hence we have the result of a scalar.
10. Finally, return the value of foo after plugging in the equation.

The reason I did not follow the formula exactly as in the book is because, I think that there was a miscommunication about how the book actually represents the variables, versus how Matlab is representing them. The book might represent them in column form, whereas Matlab represents them in row form. So a slight amount of manipulation was required to produce the desired results.