**UNSUPERVISED LEARNING**

Francis Mendoza

ASUID: 1213055998

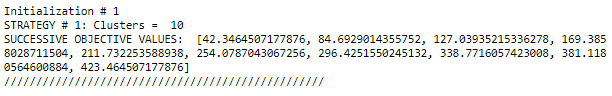
CSE575- Statistical Machine Learning

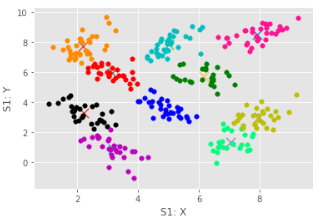
fmendoz7@asu.edu

1. **INTRODUCTION**
   1. I extracted a 2-D dataset as a .mat file and parsed the relevant data as a list of lists in my program. In this project I used two variations of K-Means Clustering, each initialized twice, to classify the points within this dataset. The entire file was first developed on the Jupyter Notebook environment and then converted into a .py file using the following command
      1. jupyter nbconvert --to script Mendoza-CSE575\_Project2.ipynb
2. **STRATEGY #1: Random Initialization**
   * 1. Strategy #1 involved randomly configuring the centroids for each individual run of the algorithm in the program main method, until we have reached k desired clusters. The program iterated several times over each individual run until convergence for k specified clusters was reached, until we ran K Means Clustering again, but for k+1 clusters instead.
     2. As per my discourse with TA Yuzhen Ding during her Zoom office hours, 3/27, the valid way to randomly initialize points was to select the first k points for k desired clusters for that run.
     3. In the graphs, the x values represented k number of clusters at a particular run, while the y values represented the SUM of the results from the objective function until k clusters were reached
        1. For example, if k = 3, the y-value reflected the sum of the objective function values for all k’s until k = 3
        2. The objective function is

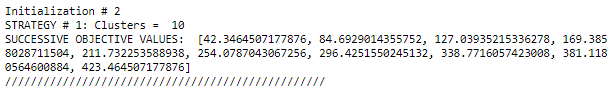


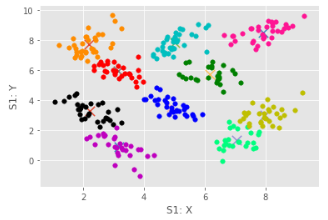
* + 1. The results for Initialization #1, Strategy #1, Yuzhen’s Method were:



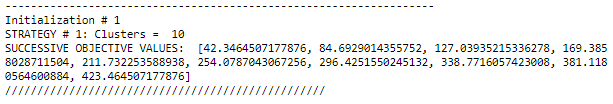
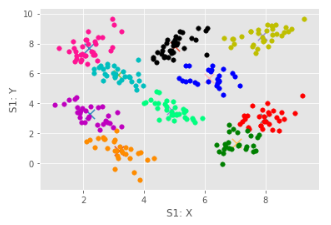
 

* + 1. The results for Initialization #2, Strategy #1, Yuzhen’s Method, were

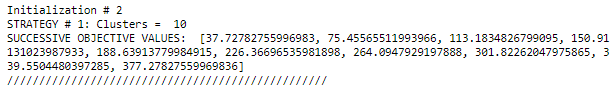
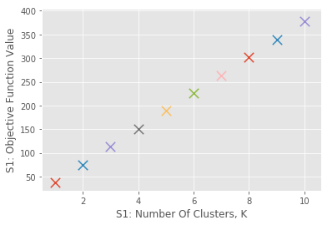
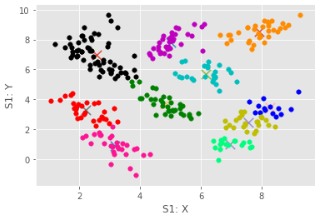


* + 1. Because we consistently picked the k first coordinates from the dataset as our valid way to randomly initialize (approved by Yuzhen Ding during her TA office hours on 3/27), there was consistency for the initializations
    2. However, K Means Clustering is very sensitive to the initial centroids. Suspecting a possible counter, I developed a secondary random initialization code block that instead relies on the Python “random” library, selecting random coordinates from the dataset using random.choice(dataset\_name). The results for this alternative method are:
    3. The results for Initialization #1, Strategy #1, Randomness Library method

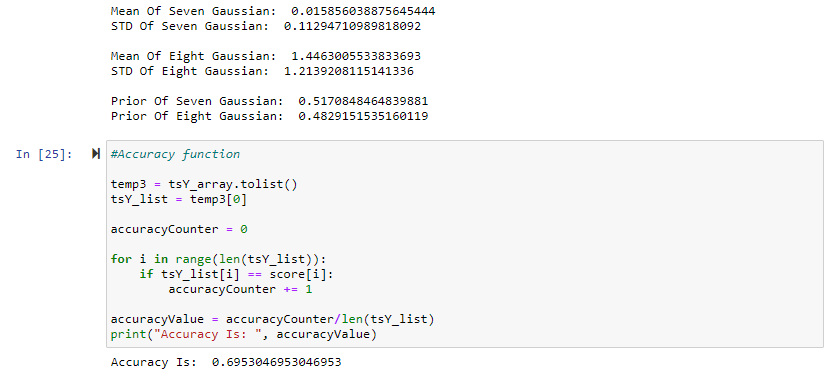
* + 1. The results for Initialization #2, Strategy #1, Randomness Library method:

1. **STRATEGY #2: Maximum Average Centroids**
   1. Although professor, in the announcements, mentioned an implementation using multivariate Gaussian distribution utilizing a mean vector and covariance, I utilized the still viable approach of disjointed 1D Gaussians that have the mean of means, mean of standard deviation, standard deviation of means and standard deviation of standard deviation. I have verified that this is a valid approach by going to Mr. Kevin Ding’s office hours on Tuesday and Wednesday (2/18-2/19) at 10:00-11:00 hrs MST.
      1. The formula I used for the Seven Naïve Bayes was as follows
      2. The formula I used for the Eight Naïve Bayes was as follows
      3. To determine the parameters (which were derived from the features), I took mean of the subarrays in the first coordinate spot and the standard deviation of the subarrays in the second coordinate spot to have a general input parameter vector consisting of the mean and standard deviation features. This returned a list of lists from the training set with 12116 rows and two columns.
      4. I then split them according to whether the respective samples would lead to a classification of “7” or “8”. Finally, I would then either take the mean or standard deviation of the entire column to get these values (for both “7” and “8” class)
         1. Mean of Means
         2. Mean of Standard Deviation
         3. Standard Deviation of Mean
         4. Standard Deviation of Standard Deviations
      5. I then took the higher value from each respective Naïve Bayes formula in order to determine the class
   2. The results from the initial feature extraction (calculating the means of both the means and the standard deviations) follows as such:

**TRAINING SET**

* 1. As a result of comparing the values returned from Bayesian formulas (by determining which is higher) for the class being a “7” or an “8”, I found that more samples leaned towards “7” than “8”. This is also reflected in the priors.
  2. The distribution is used to classify a test sample by getting n number of Gaussian distributions (depending on how many features you are measuring, in this case two: one for the mean feature and one for the standard deviation feature)
  3. Due to the results of the Naïve Bayes algorithm, I was able to find that the accuracy value is roughly 69%



1. **RESULTS**
   1. Although there are slightly more samples in the testing and training data for “7” than “8”, using Naïve Bayes yields an accuracy of roughly ~69%.
   2. Although I was unable to finish the logistic regression code, if it was completed in time, it essentially would’ve returned a decision boundary (with conditionals to determine under what class a value from the testing set would fall under), with gradient descent used to minimize loss while gradient ascent is used to adjust the weights for each epoch.