**CLASSIFICATION USING NEURAL NETWORKS AND DEEP LEARNING**

Francis Mendoza

ASUID: 1213055998

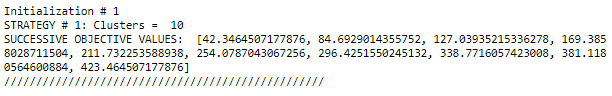
CSE575- Statistical Machine Learning

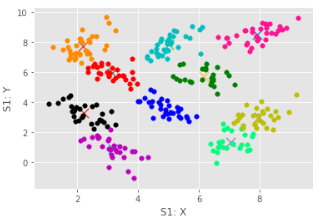
fmendoz7@asu.edu

1. **INTRODUCTION**
   1. I used the same MNIST dataset from project 1 to train a Convolutional Neural Network (CNN) using Tensorflow, Python, and Keras to classify digits. I first ran a modified version of the boilerplate code provided (keeping only the lr parameter for Adadelta, as the original copy did not run) and reported both the loss and the accuracy. In the second part of the project, I redid the experiment with the original boilerplate by changing the kernel size, and plot the learning errors as well as reported error and accuracy. I then redid the experiment a third time by changing the number of feature maps in the first and second convolutional layers, while reporting the same output of testing error, accuracy on the test set, and a plot of the learning error.
      1. jupyter nbconvert --to script Mendoza-CSE575\_Project3.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

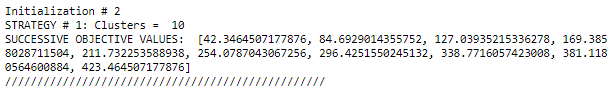


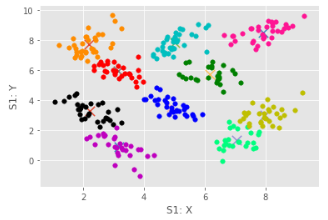
* + 1. The results for Initialization #1, Strategy #1, Yuzhen’s Method were:



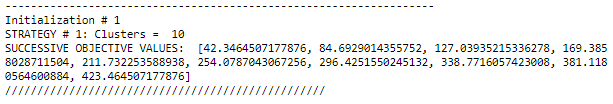
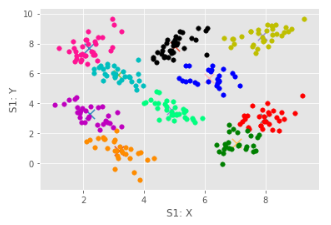
 

* + 1. The results for Initialization #2, Strategy #1, Yuzhen’s Method, were

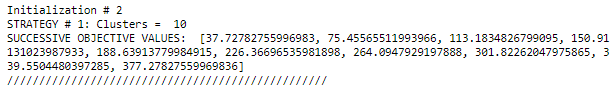
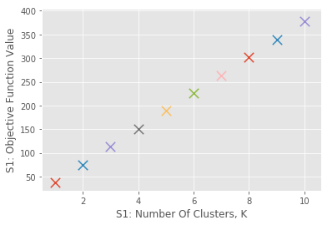
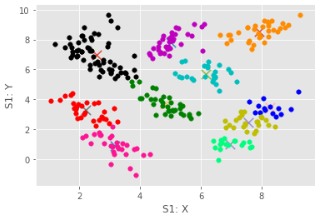


* + 1. Because we consistently picked the k first coordinates from the dataset as our valid way to randomly initialize (approved by Yuzhen Ding during her TA office hours on 3/27), there was consistency for the initializations
    2. However, K Means Clustering is very sensitive to the initial centroids. 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

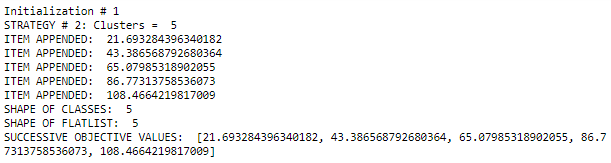
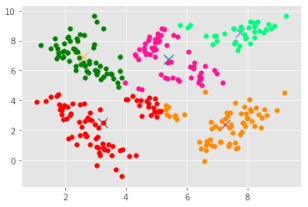
* + 1. The results for Initialization #2, Strategy #1, Randomness Library method:

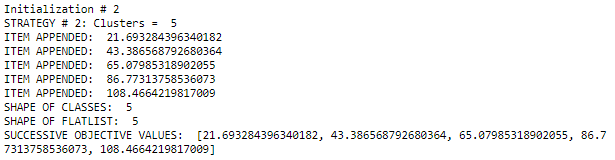
1. **STRATEGY #2: Maximum Average Centroids**
   * 1. Strategy #2 involved having the first centroid being randomly assigned, with subsequent centroids being the point with the maximum average distance in comparison to all other points. The program iterated several times over each individual run until 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 point within the dataset
     3. The graphs used the same format as strategy #1. The objective function is

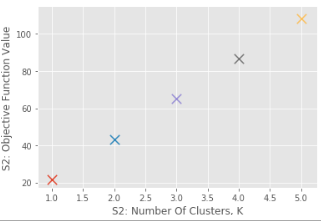
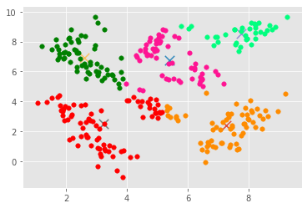


* + 1. The results for Initialization #1, Strategy #2, Yuzhen’s Method:

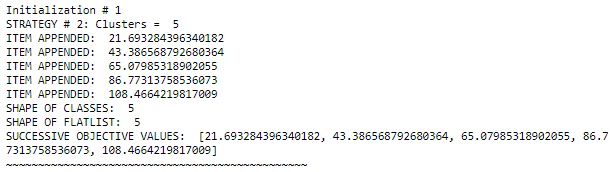
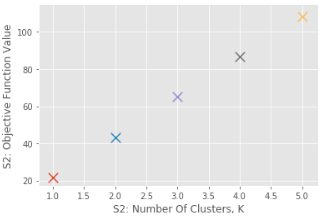
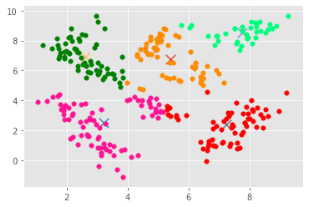
  

* + 1. The results for Initialization #2, Strategy #2, Yuzhen’s Method:

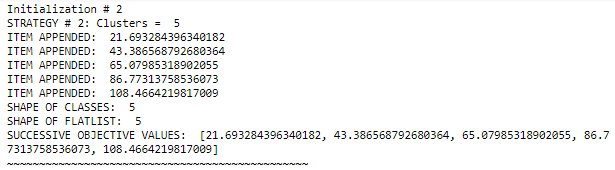


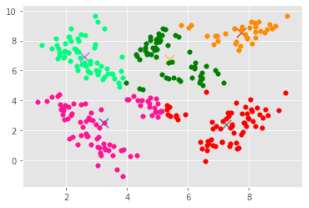
 

* + 1. The results for Initialization #1, Strategy #2, Randomness Library method:

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





1. **RESULTS**
   1. Yuzhen’s method of picking k centroids for randomly initializing (either all centroids for Strategy #1 or the first centroid for Strategy #2) did not yield any varying results
   2. Using the randomness library to initialize centroids did yield different results for multiple runs (initializations)
   3. The objective function graph for both strategies #1 and #2, with both Yuzhen’s method and the randomness library for initialization, yielded a consistent linear growth. However, for strategies #1, it yielded different values when it utilized the randomness library for initialization. This behavior has not been observed until k = 5 in strategy #2, however
   4. Interestingly, for Strategy #2, K Means Clustering operates fine until the 6th cluster. I have emailed the TA’s at roughly 15:00 hrs MST and am awaiting a response, but in case I haven’t, although the graphs are not listed for the remaining 5 clusters, the code demonstrates the core concept.
   5. The core code demonstrates clustering until k = 10 for Strategy #1 and Strategy #2. If you would like to see the progression of clustering for each increasing run for k, please run the code and comment out either Yuzhen’s method or the randomness library initialization method for your centroids
   6. Although I was unable to fully debug the Strategy #2 code to work for 10 clusters (only operates until k = 6), the core code fully demonstrates the respective operations for both methods, displaying both the objective function graph and the scatterplot, with the objective function having accurate values, as evidenced by this email correspondence with TA Yuzhen Ding

