Statistical Machine Learning Portfolio

Aryan Kumar
kumar.aryan@gmail.com/akuma420@asu.edu

***Abstract*—The statistical machine learning course involved the completion of three projects that focused on how to code the concepts that were taught during the lectures. These concepts ranged from naive bayes classifier, k-means and neural networks. Along with that the projects also asked to make charts that represented the accuracy or performance of that code.**

I. Introduction

During my time in statistical machine learning, I completed three projects individually that were required by the course. The three projects built on the conceptual knowledge in the lectures and taught me ways to program a number of machine learning concepts using python and libraries like numpy. These three projects were: density estimation and classification, two strategies of the K-means algorithm and classification using neural networks and deep learning.

II. Description of Each Project

A. Density Estimation and Classification

This project focused on deepening an understanding of the naive bayes classifier. The project gave two datasets and I had to code solutions for four different tasks. The first task was to extract features from the original train set. Specifically I had to extract the average brightness of each image and the standard deviation of the brightness of each image. To calculate the mean and standard deviation values, I used the mean() and std() functions in the scipy library.

The next task was to calculate all the parameters needed for the two class naive bayes classifier. The parameters were: the mean of feature1 with digit0, the variance of feature1 for digit0, the mean of feature2 with digit0, the variance of feature2 for digit0, the mean of feature1 with digit1, the variance of feature1 for digit1, the mean of feature2 for digit1 and the variance of feature2 for digit1. To calculate the mean and standard deviation values, I used the mean() and std() functions in the scipy library.

The third task was to implement classifiers and then predict all the unknown labels and for this the test set data was used. I did this by looping through the samples in the test features and running the predict function that I made on each sample. The predict function

The final task was to calculate the accuracy of my code for the two digits of both features. I divided the sum of the predicted labels with the length of the predicted labels. My two accuracy calculations were 0.917 and 0.9233 respectively.

The two features that I was supposed to extract for the project was the brightness of each image and standard deviation of the brightness of each image. These features can be seen as functional dependent and a study by Fish at T.J. Watson Research Center found that this classifier is pretty effective when used [3] for independent and functionally dependent features. My findings also concurred with his since my accuracy calculations for both datasets were over 90%.

B. K-Means

For this project I coded my variation of the K-means algorithm based on the steps outlined in the lectures. The project itself explored two different strategies to perform K-means. For one part use the provided initial centroids as the starting point for the algorithm. For the other I was only given one initial centroid and had to randomly generated the other. These were the two different strategies for the K-means algorithm that were covered during the lectures.

The code for the K-means algorithm is the same for both parts of the project. My code for the algorithm took three arguments: a dataset, centroids and the number of cluster. While in general for K-means the number of clusters is unknown as it is a unsupervised algorithm [4], for the project we were given the number of clusters. First the algorithm started by making a numpy array called DivideCluster of the same size as the dataset and initializing it with zeros with the zeros function in the numpy library. Because the global k-means algorithm is known to be iterative [2], I then looped through the rows in the dataset and inside that loop looped through the number of clusters and found the minimum distance between one of the centroids and all the points in the dataset along with the index of that point and I set the value of the DivideCluster array at that same index to the minimum distance. Then at the end returned both the centroids and the DivideCluster.

While a common problem with K-means is that random initialization of points can lead to convergence that isn't accounted for [1], this wasn't an issue in this project because the points or at least one point was provided instead of randomly generating everything.

C. *Classification Using Neural Networks and Deep Learning*

This project explored neural networks, the code needed to create one and how to evaluate the performance of how the network trains by calculating the loss and accuracy. The provided code goes through each step starting with importing the libraries then making definitions for the convolutional layer, the max pooling layer, the fully-connected layer, the activation and loss functions and the neural network. Then the data gets loaded in and the network is initialized to do the training and testing. This final step is where my evaluate function comes in to get the accuracy and loss values for the training and testing operations. In the training and test process step, the code loops through each epoch and runs the evaluate function after it gets the values for the sub train images, sub train labels, sub test images and sub test labels.

The evaluate function takes in three parameters: a network, images and labels. It then loops through the number of rows after finding it using the shape function of numpy arrays. It uses the same row index for the images and labels. Then the function uses the cross entropy function from the loss function definition and adds it all up in the loop to get the total loss. For the accuracy, it uses the argmax function to compare the argmax of the output and for the label, if they're equal accuracy is added by one and the total accuracy is calculated by divided that final accuracy value by the number of rows of the images array.

III. EQUATIONS

The formula to get mean of a data set in project 1:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The formula to get variance of a data set in project 1

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

Bayes rule formula [5] related to project 1

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

The objective function for k-means in project 2:

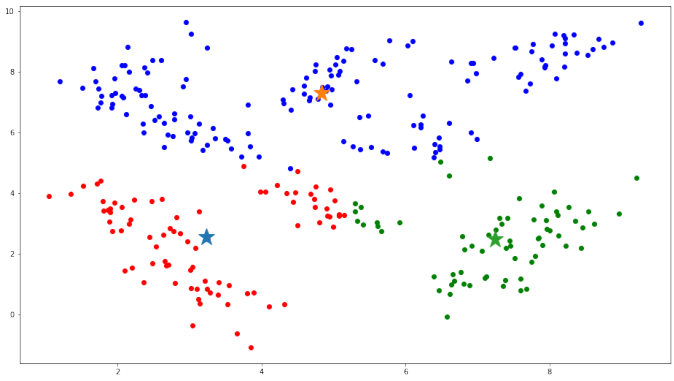
$$\sum_{i=1}^{k} \sum_{x \in D_i} \left| x - \mu_i \right|^2$$

IV. FIGURES AND RESULTS

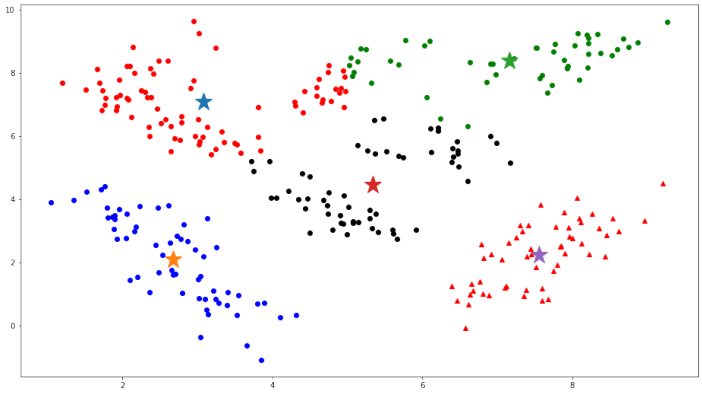
The following are the figures generated for the projects where it was applicable along with an explanation for what those graphs meant for the results of the algorithms that each project explored. Lab 1 didn't have any charts just python code for the calculations so the following figures are just from labs 2 and 3.

A. *Figures*

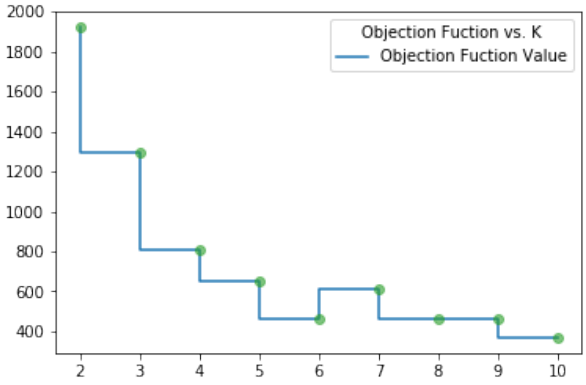
The first four figures are from lab 2 where I coded two different methods of K-means. Figures 1 and 2 go show a scatter plot that represent cluster and the stars represent the centers for the clusters that are found by the K-means algorithm using one strategy. This is the strategy randomly generates all four points as the initial starting centers for the algorithm and slowly refines it over each iteration, as mentioned earlier it follows the idea of the global k-means algorithm [2]. That refinement is shown in figure 3 where the objective function is graphed over the K. As the algorithm goes on the objective function, the error, gets closer and closer to zero after starting off high.



1. Figure 1

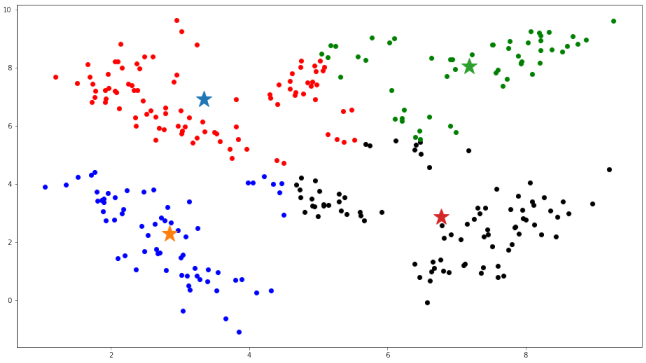


2. Figure 2

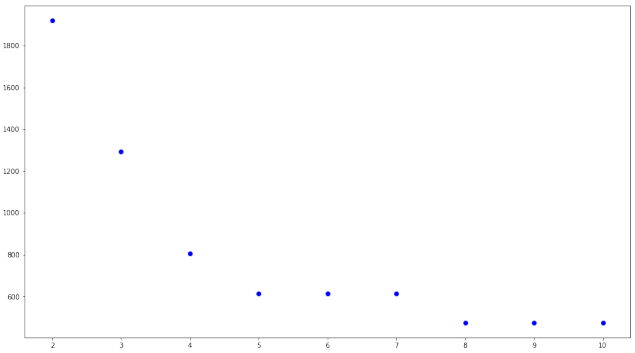


3. Figure 3

Figures 4 and 5 are still showing the results from the same lab as the last three figures but are related to the second part of the lab where a different strategy is used for K-means. This strategy gives one fixed point and then randomly generates the others instead of all of the points being randomly generated like the first method. Similar to the previous scatter plots, this one also uses stars to represent the resulting centers for the scatter plots. Figure 5 shows the performance of this method similar to figure 3 in the last method. Just like the last method the error rate gets lower and lower as more iterations of the algorithm are done. Figure 3 and figure 5 really prove the the general idea that the nature of the global k-means algorithm is to be iterative [2] and is a good visual representation of the loop used in my code to perform that iteration.



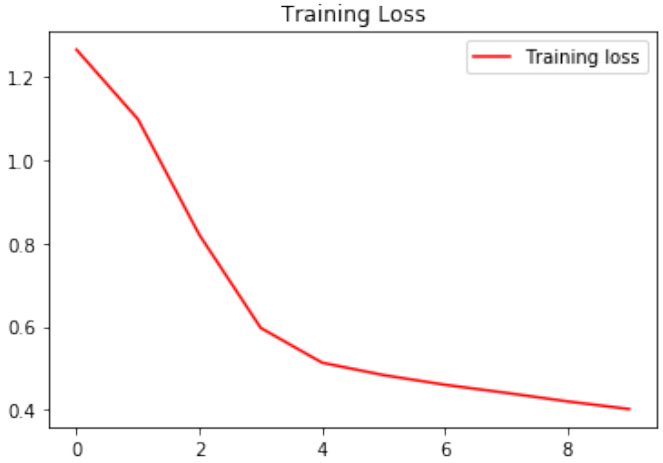
4. Figure 4



5. Figure 5

The next set of graphs show the output of lab 3. In this lab the task was to finish the evaluate function which would give the accuracy and loss of the classification algorithm used in the neural network provided by the remainder of the lab code. The generated for the accuracy and loss calculations highlighted a common link between these machine learning algorithms. They seem to get better as they go through more iterations seemingly learning and improving as they get more attempts.

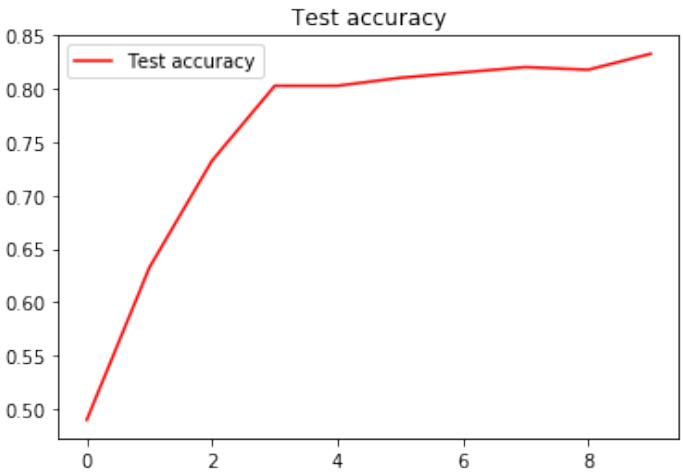
6. Figure 6



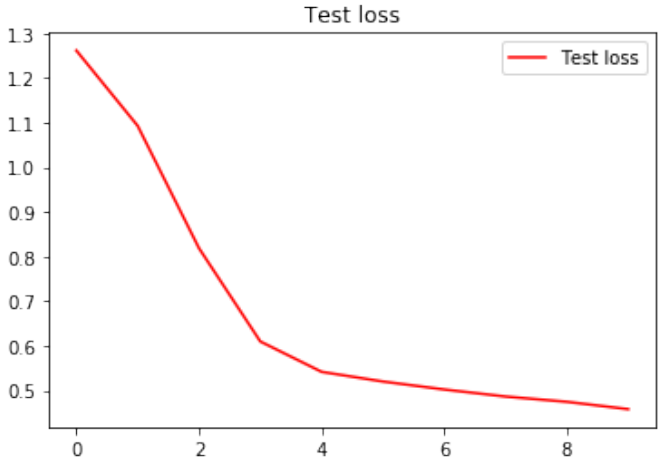
7. Figure 7

As shown by figure 6, the accuracy of the training data steadily goes up and is still trending upwards without plateauing so it can be reasonably assumed that if I had let the algorithm run about 15-20 times then the accuracy would've kept climbing until it hit 100% or over 99% accuracy. This trend continues with figure 7 which measures the error rate of the same training

dataset. The error steadily goes down and continues to show a downward tread as the error comes closer and closer to zero, almost perfectly mirroring figure 6.



8. Figure 8



9. Figure 9

Figures 8 and 9 show that there's a similar trend in the test dataset as well as the training. An interesting observation is that the training dataset produced a smoother curve than the test dataset at least for the accuracy calculations but despite that it still got to a similar accuracy as the training dataset after the same out of runs. This leads me to the conclusion that regardless of what data is in the dataset the algorithm produced similar enough results and will eventually lead itself to the most accurate, least loss solution if its allowed to run for enough time.

ACKNOWLEDGMENT

Thanks to professor Taher and the teaching assistants for doing live sessions to help further my understanding of the lectures which was crucial in helping me actually code out the concepts for each of the projects. Also thanks to professor Li for making the pre-recorded lectures for this class to even be possible in the first place.

REFERENCES

1. Ahmed, Mohiuddin, Raihan Seraj, and Syed Mohammed Shamsul Islam. 2020. "The *k-means* Algorithm: A Comprehensive Survey and Performance Evaluation" *Electronics* 9, no. 8: 1295. https://doi.org/10.3390/electronics9081295.
2. Likas, A., Vlassis, N. and Verbeek, J.J., 2003. The global k-means clustering algorithm. Pattern recognition, 36(2), pp. 451-461
3. Rish, I., 2001, August. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, No. 22, pp. 41-46).
4. Sinaga, Kristina P., and Miin-Shen Yang. "Unsupervised K-means clustering algorithm." *IEEE access* 8 (2020): 80716-80727.
5. Webb, Geoffrey I., Eamonn Keogh, and Risto Miikkulainen. "Naïve Bayes." *Encyclopedia of machine learning* 15 (2010): 713-714.