CSE-575: Statistical Machine Learning

Sneha Kumari
Arizona State University
Online
skumari6@asu.edu

1. Project Introduction

There are three projects in the course CSE 575. All are related to work on MNIST [1] dataset. In all parts of projects, we will work to get the classification accuracy with the help of three different probabilistic machine learning models.

Project 1: Density Estimation and Classification

The MNIST dataset is a set of handwritten digits (0-9), and in the first project, our job is to build a computer program that takes as input of all training images from digit 0 and digit 1. Compute the Probability density for both digits of training data, implement the Naïve [2] Bayes Classifier and use it to produce a predicted label for each image of testing sample. The Naïve Bayes classifier is a simple classifier that is often used as a baseline for comparison with more complex classifiers.

Project 2: Unsupervised Learning (K-Means [3])

For this project, we need to implement K-Mean approach to find the classification accuracy of MNIST dataset without using any specific Python libraries like, scikit-learn. K-means tries to partition data points into the set of k clusters where each data point is assigned to its closest cluster. This method is defined by the objective function which tries to minimize the sum of all squared distances within a cluster, for all clusters. It is an unsupervised learning algorithm.

Project 3: Classification Using Neural Networks[4] and Deep Learning

In this project, we need to get the classification of MNIST dataset, where 60,000 images are used to train the network and 10,000 images to evaluate how accurately the network learned to classify images. Here we will use the approach of Neural Network model. A neural network is a type of machine learning which models itself after the human brain. This creates an artificial neural network that via an algorithm allows the computer to learn by incorporating new data. A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can input an image, assign importance to various aspects in the image and be able to differentiate one image from the other.

2. Solution Details

Project 1: Density Estimation and Classification
For the first Project, I worked on Naïve Bayes classifier. For this I divided the given MNIST dataset into two parts: Training dataset and Testing Dataset. Where 60,000 images were used as training dataset and 10,000 as testing dataset.
For this the first step was to extract features for both training set and testing set in Python. The computation was performed with the help of predefined mean and var functions of Python "numpy" library.
Next step was to estimate/compute the parameters of the relevant distributions. Then to perform MLE Density Estimation first parameter from feature 1 (average brightness) was calculated as mean of feature 1 for each image. And second parameter as variance of feature 1 for each image of Train0 data. From feature 2 (average variance) first parameter was calculated as mean of feature2 for each image. And second parameter as variance of feature2 for each image of Train0 data.
To estimate or compute the probability density for both Test data. This computation was done in a user defined function. Where this formula for normal distribution was used.

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

In the next step I implemented the Naïve Bayes classifier and use it to produce a predicted label for each testing sample. With the given prior probabilities, 0.5 for both digit 0 and digit 1.
To implement the Naive Bayes Classifier for both Test data sample with both the Train data. This was computed with the given formula.

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{i=1}^{d} p(x_i \mid y)$$

In the final step, I computed the classification accuracy. For this task first Test0 data sample for each digit and Test1 data sample was calculated individually. This

task was performed with the help of predicted labeled list from the Naive Bayes classifier and the Test0 and Test1 label list from related label files.
Accuracy (In Percentage) = No. of correct predictions/ Total no. of prediction

Both individual and overall accuracy in percentage are as follows.

Classification accuracy for TestData0 sample is: 91.224%
Classification accuracy for TeatData1 sample is: 92.247%
Overall classification accuracy: 91.773%

Project 2: Unsupervised Learning (K-Means)

For this project, I worked on K-Mean algorithm. Like project1 this algorithm was implemented with some steps.
In the first step, I got sample data in python and then convert those files in the Python "Data Frame".
It contains a set of 2-D points. Two different strategies were performed for choosing the initial cluster centers.
Strategy1: Initial centers were picked randomly from the given samples.
Strategy2: First center was picked randomly; for the i-th center (i>1), a sample (among all possible samples) was chosen such that the average distance of this chosen one to all previous (i-1) centers is maximal.
Strategy1:
This flow chart was implemented to get the centroids with strategy1.

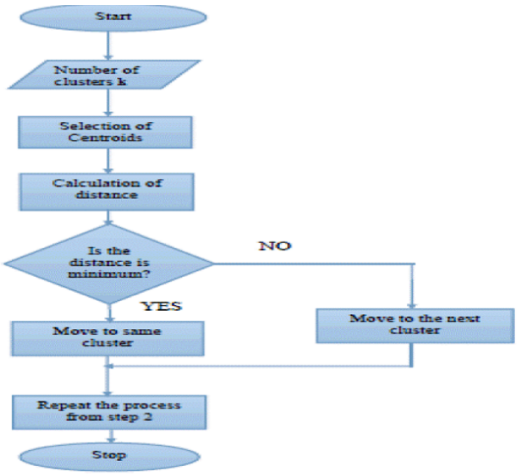
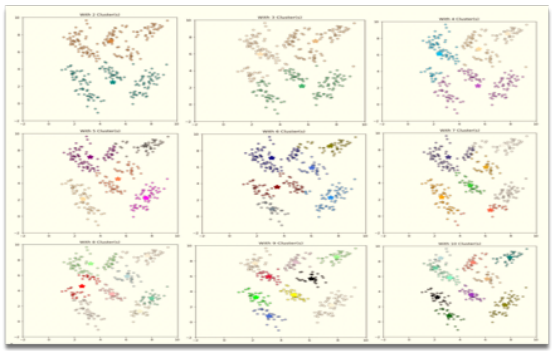


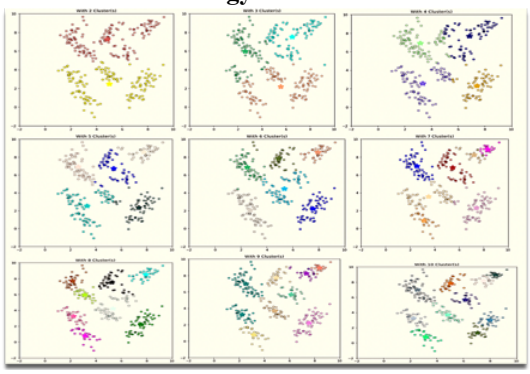
FIG. 1. **Flowchart for K-means clustering.**

Strategy2:
All centroids are initialized with the concept of strategy 2. K-Means algorithm is implemented with two sets of initialized centers. First, one centroid was initialized then other were chosen maximally apart from earlier one.

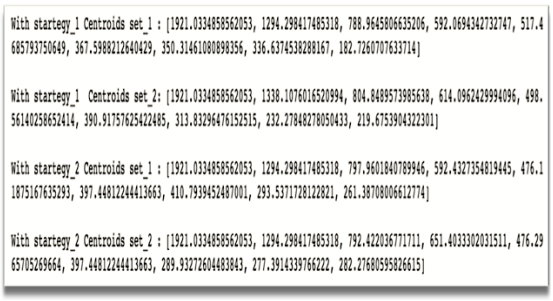
Results with Strategy1:



Results with Strategy2:

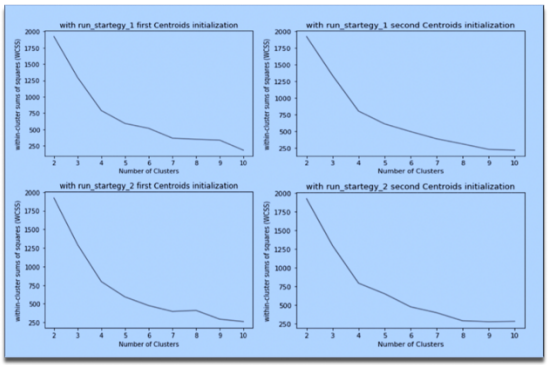


Final centroids with both strategies:



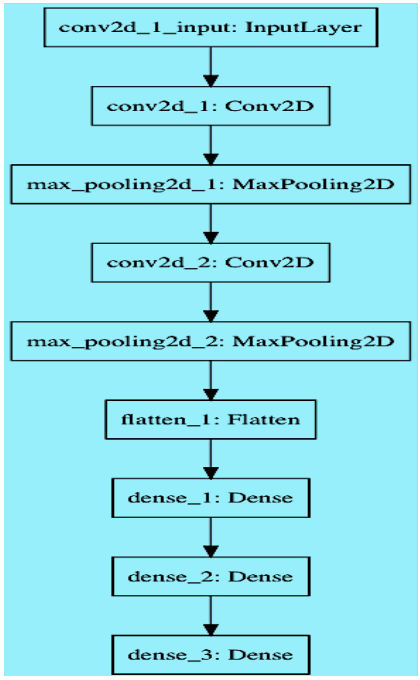
To compute the objective function with values of k (k = 2, 3, ..., 10). With strategy1 and strategy2 and plots for the objective function values under these conditions. A measure of how well the centroids represent the members of their clusters is *sum of* the squared distance of each vector from its centroid summed over all vectors.

Plotted graph with both strategies from two sets of centroids are:



Project 3: Classification Using Neural Networks and Deep Learning

This is the project to classification of MNIST dataset, where 60,000 images are used to train the network and 10,000 images to evaluate how accurately the network learned to classify images. The images are 28x28 NumPy arrays, with pixel values ranging from 0 to 255. Classification of images are done with the help of Neural Network. Some predefined Python language libraries were used to perform the tasks.
With help of library (keras.datasets import mnist). The data set is divided into two parts, training and testing data. Some initialization is performed like:
batch_size= 128, num_classes=10, epochs=12, img rows, img cols= 28,28.
Got the shape of train and test data samples. The input size is the size of the image (28x28). 60000 train samples, 10000 test samples
To convert class vectors into binary class matrices. Then perform some steps for baseline. For baseline, experiment with the convolutional neural network.
To perform this, experiment this flow chart is used:



Final table with all experiment results:

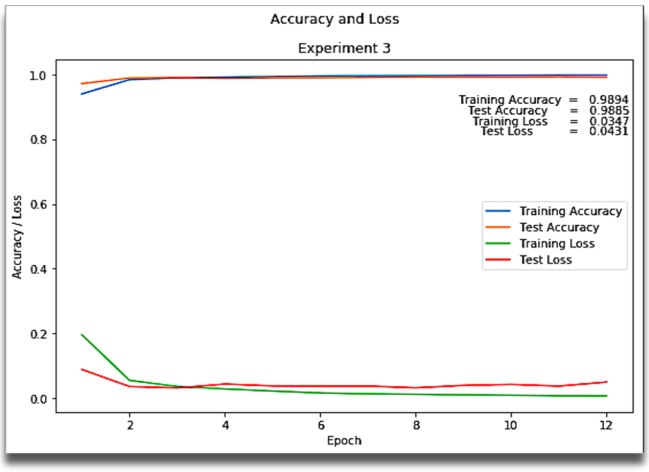
Execution	Training Accuracy	Training Loss	Test Accuracy	Test Loss
Baseline	0.9887	0.0360	0.9870	0.0445
Experiment #1	0.9900	0.0324	0.9895	0.0367
Experiment #2	0.9895	0.0331	0.9898	0.0357
Experiment #3	0.9894	0.0347	0.9885	0.0431

Train the network with the training set and then test it with testing set. Plot the training error and the testing error as a function of the learning epochs.

(1) The input size is the size of the image (28x28).
(2) The first hidden layer is a convolutional layer, with 6 feature maps. The convolution kernels are of 3x3 in size. Use stride 1 for convolution.
(3) The convolutional layer is followed by a max pooling layer. The pooling is 2x2 with stride 1.
(4) After max pooling, the layer is connected to the next convolutional layer, with 16 feature maps. The convolution kernels are of 3x3 in size. Use stride 1 for convolution.
(5) The second convolutional layer is followed by a max pooling layer. The pooling is 2x2 with stride 1.
(6) After max pooling, the layer is fully connected to the next hidden layer with 120 nodes and "relu" as the activation function.
(7) The fully connected layer is followed by another fully connected layer with 84 nodes and relu as the activation function, then connected to a "SoftMax layer" with 10 output nodes (corresponding to the 10 classes).

```
Training Accuracy      = 0.9893736111111112
Test     Accuracy      = 0.9885333333333334
Training Loss          = 0.034727734817206334
Test Loss              = 0.043144933235565563
```

```
Title BaseLine
Configuration Parameters
Batch Size          128
Number of Class     10
Epochs              12
Kernel_x            3
Kernel_y            3
Conv2D1             6
Conv2D2             16
```



At that level the performance is close to human-equivalent, and is arguably better, since quite a few of the MNIST images are difficult even for humans to recognize with confidence. Accuracy obtained with these parameters are very close to perfection. With two kernel values and two sets of feature maps size.

3. Results

In the first project the accuracy I got with the help of Naïve Bayes classifier was pretty good. It was 91.773%. As this model of machine learning is used as the base for other models, the results are quite interesting.
In the second project where I used the K-Means algorithm of Machine learning models. Based on the results it can be said that the result might not be globally optimal: We can't assure that these strategies will lead to the best global solution. Selecting different random centroids at the beginning affects the final results. The third project, which was done with Neural Network and Deep Learning with the help of some specific Python libraries. The results were comparatively better than the other models I used. The accuracy I got for the overall dataset was 98.94%. This project showed that accuracy can be improved with changing feature maps, number of Kernels and epochs.

4. My Contribution

In the course all three projects were individual. I was the solo contributor for all three projects. All the solutions and results obtained by me with the help of course materials and some other recourses which I found online.
During working on projects all models are applied with Python language. For all three specific models of machine learning like, Naïve Bayes Classifier, K-Means or Neural network predefined libraries for those models were not used. The steps were performed manually from scratch to get the accuracy with those models. I performed some additional tasks, plotted the normal distribution graph to visualize the features characteristics. I compared all the results of mine with Python predefined libraries to those specific libraries. I got pretty similar results.

5. Lesson Learned

During all three projects of the course, I learnt how to work effectively with MNIST dataset. Some important Python libraries helped me to get the results quick and accurate. It was quite exiting to work with the dataset and visualize the features with the help of graphs I plotted during the projects. To work on machine learning models from scratch, we must know the basics of linear algebra, statistics and probability. I spend quite some time to refresh those skills. Some specific libraries I worked with like, NumPy, Pandas, Keras, Skilit-Learn which I found exceptionally supportive during projects. I will continue to use these learning as a reference in the future.

References

[1] https://en.wikipedia.org/wiki/MNIST_database
[2] https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
[3] https://realpython.com/k-means-clustering-python/
[4] https://pythonprogramming.net/neural-networks-machine-learning-tutorial/