Statistical Machine Learning

Trayson Keli'i
Arizona State University

Abstract

This paper aims to demonstrate the practical application of concepts covered in CSE 575 through a series of mini-projects. Unlike a single extensive project spanning the course, these mini-projects provided a diverse range of hands-on experiences with various statistical machine learning algorithms. This structure facilitated a comprehensive understanding and application of different methodologies. The course, conducted in Python, included coding assignments focused on the implementation of Naive Bayes, K-Means clustering, and Convolutional Neural Networks (CNN). This approach allowed for a thorough exploration and practical engagement with these fundamental statistical machine learning techniques.

1 Introduction

For each machine learning task, the Modified National Institute of Standards and Technology (MNIST) handwritten digit database was utilized. This dataset comprises 60,000 training examples and 10,000 test examples[3]. The structure of this paper is as follows: an introduction to the problem and concept, a description of the proposed solution, presentation of results, an outline of my contributions, and a discussion of the lessons learned.

1.1 Naive Bayes

For the Naive Bayes classifier, the task involved training a model to distinguish between images of the digits one and zero. This classifier belongs to the family of Bayes probabilistic classifiers and is termed "naive" due to the assumption of independence among the features of the dataset[1]. Bayes' Theorem, represented as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

leverages this independence assumption. By applying Bayes' Theorem, the product of the probabilities for each feature of a given sample can be used to classify whether an image represents a one or a zero.

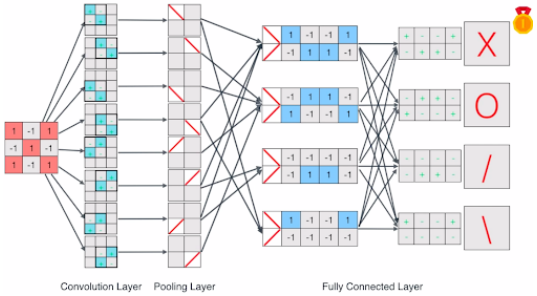

Figure 1: Visual representation of CNN architecture

1.2 K-Means

K-Means is employed as an unsupervised learning approach, although it is also recognized for its applications in semi-supervised learning[2]. It was assigned a task analogous to that of the Naive Bayes problem, involving the separation of images of ones and zeros. However, instead of classifying the images, the K-Means algorithm was used to cluster them. The objective function of K-Means is given by:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2 \tag{2}$$

This equation represents the Sum Squared Error (SSE), where k denotes the number of clusters and n denotes the number of members in a given cluster. The goal is to minimize J, the objective function, thereby grouping each sample into a cluster with similar features.

1.3 CNN

The problem CNN will try to solve is to classify data randomly selected from four groups of the MNIST data-set. CNN is a supervised learner with multiple layers. It is important to note that the majority of this lab was to understand the layers. There are three main layers that make up the CNN: the Convolution Layer, the Pooling Layer, and the fully connected layer.

As demonstrated in Figure 1, we learn of the different functions for each layer at a high level. The convolution layer extracts features from the image through element-wise multiplication and summation, that output is then fed to the pooling

layer which takes those features to a more compact representation of the image. The output from the pooling layer is then sent to the fully connected layer, which is basically a multilayer perceptron and the final output of that fully connected layer is the classification of the given image.

2 Description of Solution

This section describes the technical details of the assignments and how each method was implemented to accomplished the problems introduced in the prior section.

2.1 Naive Bayes Solution

The features that were selected to represent an input image was the average brightness and the standard deviation of brightness for each image. The average brightness was calculated through the following formula:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{3}$$

while the standard deviation was calculated with the equation below:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}} \tag{4}$$

Both cases were very trivial especially through leveraging the numpy python library functions numpy.average() for μ and numpy.std() for σ[5]. With the features extracted for each image in the data set we can revisit equation 1. We will let A be a given input feature and B be the label for the given input. We now know that we no longer need to find P(B) because that is implied through P(A|B). We can further simplify this equation by substituting P(A) with 0.5 because the assignment said that the distribution of ones and zeros are both 0.5. Since both are .5, they make no mathematical difference to the equation leaving us left to calculate P(B|A). It was also given that the data was sampled from a normal distribution. This meant that we are able to use the Gaussian Normal Distribution (GND) function to help find P(B|A). Below is the equation.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{5}$$

This equation requires that both μ (mean) and σ^2 (variance) for the given feature in order to find the distribution of x. With that in mind, we would need to find the mean and variance of feature μ and the mean and variance of feature σ. Once those were found we could plug all the given values into the GND function to determine what probability a given input was given a certain label. Since we found these parameter the GND would be calculated four times per input. Twice for features μ and σ given that the label was one and twice more for those same features given that the label was zero. Since we are using Naive Bayes and assumed independence among features, the two outputs from the same label (both μ and σ given label one for instance) could be multiplied to provide an overall probability of the image. The prediction would be made of the larger of the products.

2.2 K-Means Solution

K-Means is a very intuitive solution. We were assigned to approach this problem with two different strategies. The first being randomly selected initial points and the second being that described as K-Means++[4] which basically states that the next initial centroid will be the point that is the furthest average distance of the currently selected centroids (first point initialized at random). The concept of K-Means is simple:

1. Given k initial centroids
2. Assign all points in the data-set to the closest centroid
3. Recalculate the new centroid values (average) after all points have been assigned
4. Repeat steps 2 and 3 until the centroid values no longer change

In the assignment the distance metric used was the Euclidean Distance (ED) which is defined by the equation below:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{6}$$

The next dilemma is determining error without labelled data to valide. We can solve this through seeing how "good" a cluster is. There are two measurements that contribute to a clusters "goodness" the first being the intra-cluster distance (distance of points in the same cluster) and the second being the inter-cluster distance (distance of points from other clusters). A good K-Means algorithm will have a small intra-cluster distance and a large inter-cluster distance [6]. If you remember the equation 2, we are summing up all the intra-cluster distances for each cluster, so if we minimize this objective function (the SSE) we are essentially creating the best possible solution for K-Means.

2.3 Deep Learning Solution

For the Deep Learning project, the majority of the code scaffolding was already given. The main point of the lab was understanding how we could evaluate an image input after the training had been completed. In order to perform the evaluate we needed to understand how each layer was interacting with the data. The first layer was the convolution layer:

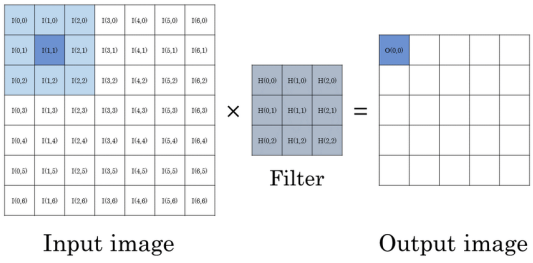

Figure 2: Example of convolution

In the convolution layer a kernel is used to element-wise multiply and then sum the values to output a new value as depicted in the image above. The project used a 5x5 kernel with a stride of two. The activation function used for the output of

this convolution is the Rectified Linear Unit (ReLU) which is depicted below:

$$a = max(0, z) \tag{7}$$

The next Layer that passes through the activation layer is the pooling layer. In the assignment we are using Maxpooling which can be depicted in the image below:

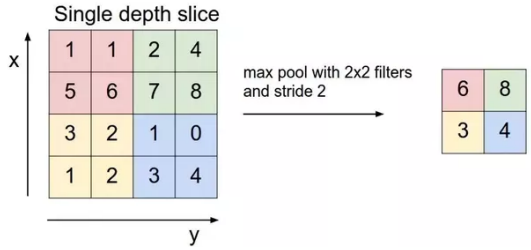

Figure 3: Example of maxpooling

As depicted, the number of the largest value for a given slice is compressed to a smaller representation of the image. These values get flattened (meaning taken from their matrix representation to a 1D array of values) and placed into the fully connected layer:

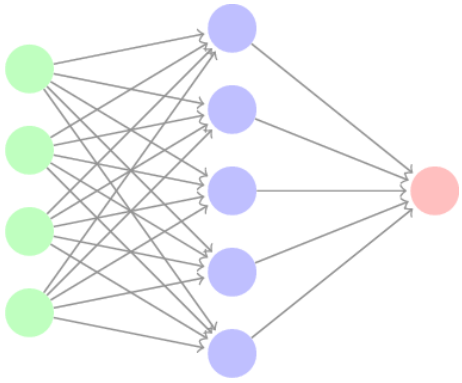

Figure 4: Visual representation of the fully connected layer

Those values are then activated with ReLU layer and passed to another fully connected layer. In the end a softmax layer is used to perform the final prediction:

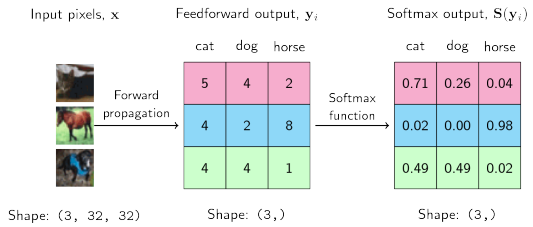

Figure 5: Visual representation of the softmax layer

This is the high level representation of the overall process. Now we need to update the weights (meaning updating the values from the fully connected layers all the way back to the kernel values). To calculate the error we will use the cross-entropy loss function:

$$CE = -\sum_{i}^{C} t_i log(f(s)_i) \tag{8}$$

These calculated errors are back-propagated through the layers and update the weights accordingly. The project came with all this code filled out for us. The main thing we needed to understand was the forward pass process. When running an image through the forward pass we are essentially using the network to make a prediction for the image. In order to calculate accuracy we would need to get the number of images classified correctly over total images. The same would also need to happen for the loss being returned as well.

3 Results

3.1 Naive Bayes Results

There was great success using the Naive Bayes classifier. It should be noted that Naive Bayes Classifier is a generative model (meaning that we estimated $P(B|A)$ and P(A) given $P(A|B)$ from equation 1). We think that the reason for such success comes from how trivial the assignment is. Looking at the parameters below also reveal observable data that coincide with the image properties (for instance, $(B_\mu|0)$ (Mean Brightness given Zero) should have a larger value than $(B_\mu|1)$ (Mean Brightness given One) because there would be more pixel values taking up space relative to the pixel values required to write the number one)

Param	Value
$(B_\mu\|1)$	44.1451875000
$(B_\sigma\|1)$	115.564696532
$(B_\mu\|0)$	87.3679032414
$(B_\sigma\|0)$	102.206746792
$(S_\mu\|1)$	19.3792637755
$(S_\sigma\|1)$	30.8765656831
$(S_\mu\|0)$	61.3955623814
$(S_\sigma\|0)$	80.8244432731

Figure 6: B is brightness and S is standard deviation

Accuracy	Value
1	0.9233480176211454
0	0.9173469387755102

Figure 7: The final accuracy for both Naive Bayes classifiers

3.2 K-Means results

Initially you can see the trade-off to get initial points that were further away did produce drastic changes in results in terms of SEE. It should also be noted that there is an inverse relationship between the SSE and the number of K's. It cannot

be ignored that with selected points of K=4 there was a significant decrease in SSE compared to K=3, what was inconclusive is whether the loss was due to the increase in K or the selection of points. Looking at the graphs we can clearly see that the SSE would be large for three because of the sparsity of points. It may be worth mentioning that perhaps a graph such as DBSCAN may have better fitted the data. K-Means with ED tend to perform better when the data is globular.

K	SSE
3	1338.13
5	613.99

Figure 8: SSE results for selecting K initial points at random

,

K	SSE
4	803.22
6	468.95

Figure 9: SSE results for selecting K initial points based on furthest average distance

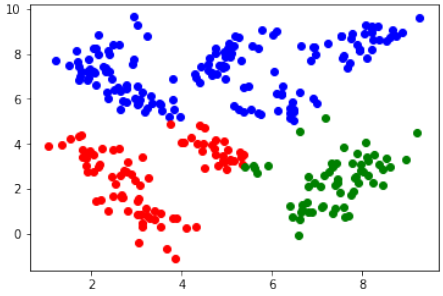

Figure 10: Scatter plot results for K = 3 with random initial point selection

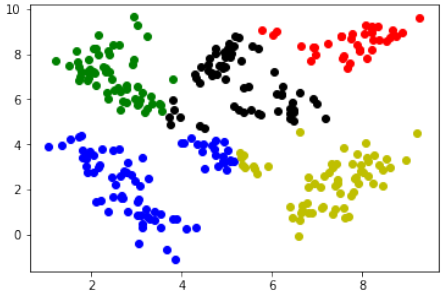

Figure 11: Scatter plot results for K = 5 with random initial point selection

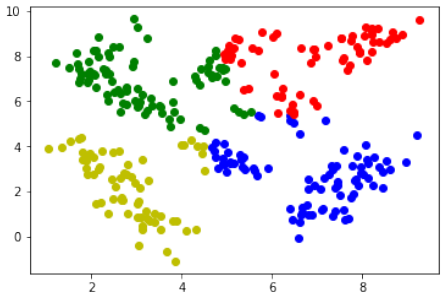

Figure 12: Scatter plot results for K = 4 with furthest average distance for initial point selection

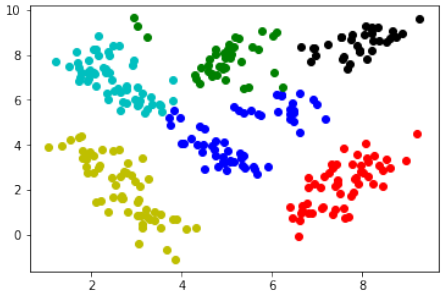

Figure 13: Scatter plot results for K = 6 with furthest average distance for initial point selection

3.3 CNN Solution

As we can observe the graphs for both train and test accuracy shows that learning of the images took place. It is also worth mentioning that there was a dip in accuracy on the last epoch of the training accuracy vs epoch graph. The loss for both train and test also followed similar patterns and they too had slight upward ticks of Loss gained. This may infer that there is a minima there within the data. There is an inverse relationship between the both accuracies and losses. As the accuracy increases, the loss decreases for each epoch. For further research, I'd argue that there are better ways to find the optimal solution for this CNN through a more rigorous stopping criteria that may be based on the loss or accuracy, rather than an arbitrary ten epoch stop. I do understand this was to facilitate grading.

4 Contribution

I performed all the experiments solo. This included the programming, the testing, the research, and the final write up.

5 Lessons Learned

I have come to appreciate the slice notation of python and the power and functionality of numpy. I have also learned not only the importance of, but how applicable Linear Algebra is to machine learning. There is a lot of overlap between the definitions, for instance the term norm $||x||$ and Euclidean distance are basically one in the same. I also have come to understand that in the world of machine learning there are overarching patterns or designs (Neural Nets vs Bayes vs SVM...)

CNN Results

Epoch	Train Acc	Train Loss	Test Acc	Test Loss
1	0.250	1.385	0.250	1.385
2	0.368	1.377	0.335	1.380
3	0.433	1.349	0.398	1.355
4	0.487	1.237	0.480	1.239
5	0.614	0.941	0.583	0.945
6	0.638	0.837	0.598	0.849
7	0.662	0.777	0.625	0.797
8	0.689	0.740	0.657	0.764
9	0.709	0.707	0.665	0.739
10	0.701	0.715	0.672	0.750

Figure 14: According to the CNN results we can see the inverse relationship between accuracy and loss for both the test and train classes as the epochs increase

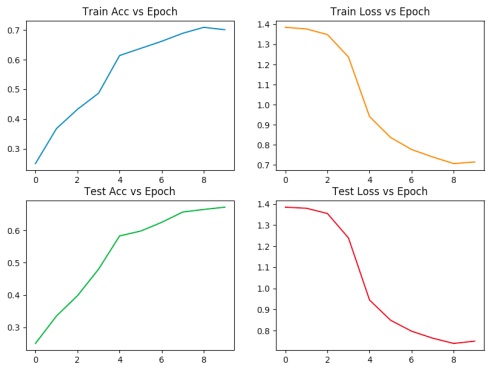

Figure 15: Graphs for each given metric in respect to time

and how true research is understanding different ways to apply said approaches to solve complex problems. This concept was very apparent in the CNN/Deep learning section of the class. Realizing that the idea of Gradient Descent (taught in the Artificial Neural Network section) was scaled through back-propagation of layers was mind blowing. I am curious to know how to determine certain layers regarding a given problem. It makes me wonder whether patterns are a lot of guess and check or is there learning in all the madness. I am grateful for the topics this class has taught me and I am excited to learn more and apply these concepts into my professional career.

References

[1] Naive Bayes Classifier (April 2020). [Wiki]. Retrieved from https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[2] K-Means Clustering (April 2020). [Wiki]. Retrieved from https://en.wikipedia.org/w/index.php?title=K-means_clustering&action=history

[3] Burges, C., Cortes, C., LeCun, Y. (n.d.). [Website]. Retrieved from http://yann.lecun.com/exdb/mnist/

[4] K-Means++ (February 2020). [Wiki]. Retrieved from https://en.wikipedia.org/wiki/K-means%2B%2B

[5] Numpy. (n.d.). [Website]. Retrieved from https://numpy.org/

[6] Cluster Analysis. (March 2020). [Wiki]. Retrieved from https://en.wikipedia.org/wiki/Cluster_analysis