ECE 532 Final Project Final Report

Owen Seymour
Section 002
December 11, 2020

Introduction

GitHub page: https://github.com/oseymour/ECE_532_Final_Project.git

This project sought to evaluate the performance of three machine learning algorithms in classifying images of handwritten digits from the MNIST dataset. The three algorithms evaluated were K-means, ridge classification, and a neural network. In designing each algorithm, various parameters were modified to produce the most accurate model before applying the algorithm to the withheld test set of images.

MNIST Dataset

The MNIST image dataset (published at http://yann.lecun.com/exdb/mnist/), is a set of 70,000 28 pixels x 28 pixels images of handwritten digits. For this project, the dataset was loaded via an interface that was part of the TensorFlow package for Python. The dataset was split into a training set consisting of 60,000 images and a test set consisting of 10,000 images. Each image was vectorized and organized into a row of a matrix. Thus, the dataset was considered to have 784 features, one for each pixel. Below is a figure showing the distribution of images projected into a two-dimensional space. The extreme density of images and digit classes will prove to make classification difficult with a couple of the classification algorithms.

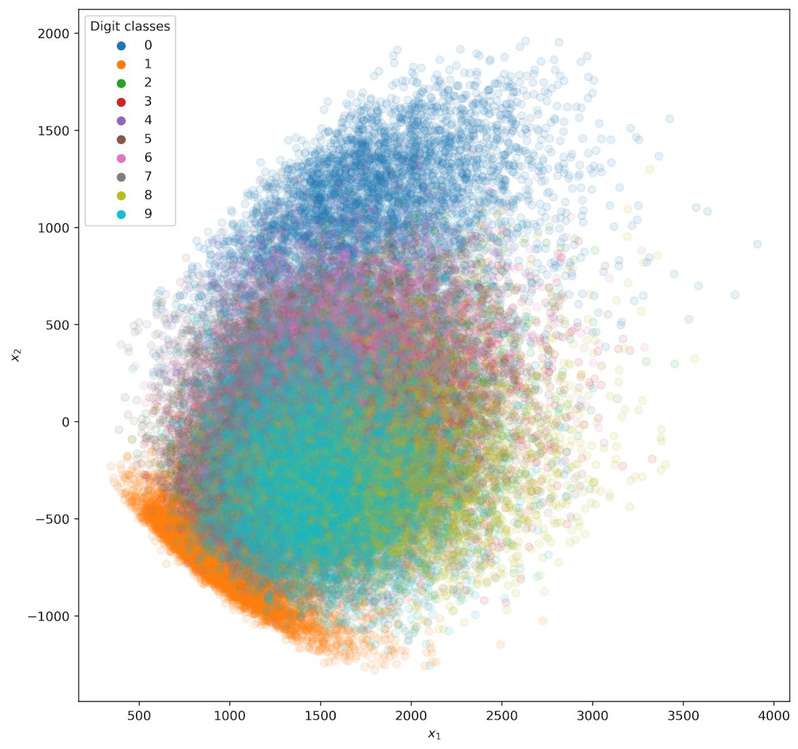

Figure 1: A 2-D representation of MNIST images

Algorithms Used

K-means

The first algorithm, K-means, is an unsupervised learning algorithm that clusters images in the training dataset into n = k clusters. To do this, cluster center locations (centroids) are initialized and then the following occurs iteratively:

1. Assign every image to the cluster of the nearest centroid.
2. After all images are assigned, update the centroids as the average position of all images in that centroid's cluster.
3. Repeat the steps above until the centroids have converged and have not changed position from one position to the next.

The parameters of this algorithm that were modified to achieve better performance included k (the number of clusters) and the method of initiating centroids locations. Ideally, cross-validation would be used to optimize both of these parameters, but due to restrictions, a full cross-validation process was not performed for this algorithm. Once the training process was completed, the algorithm returned the centroids and they were then viewed as images. These images needed to be labeled as the digits they most closely resembled prior to being able to use them for classification. This labeling of centroids was a manual process and thus prevented a large-scale cross-validation procedure from occurring for this algorithm, due to time constraints. Additionally, the algorithm is very slow to train and this was another factor that prevented a true cross-validation procedure for the K-means algorithm. Training and classification functions for this algorithm were developed by the author.

Ridge classification

This flavor of classification uses a squared error loss function and regularizes with the squared two-norm of the weights vector, weighted by a regularization parameter lambda, λ.

Eq (1) $min \| y - x^T w \|_2^2 + \lambda \|w\|_2^2$

A weights vector is found to minimize Eq (1) and this weights vector defines the classifier's decision boundary. Since ridge regression was being used for classification in this project, a decision boundary was found for each of the digits. The first step of classifying a new image was to find the inner product between the image and the weight vector.

Eq (2) $\widehat{y} = x^T w$

This inner product was found with the weights vector for every decision boundary. The image then got classified as the digit whose weight vector produces the most positive inner product.

For this algorithm, the sklearn.linear_model.RidgeClassifier class was used, and again two parameters were modified to optimize algorithm performance. The first parameter modified was the number of singular values kept in reducing the training data with the singular value decomposition, and the second was the regularization parameter, lambda. For this algorithm, full cross-validation was performed. The 60,000 training images were divided into five subsets and one of those subsets was withheld for testing while the other four were used for training. Retention of one, 100, 200, 300, 400, 500, 600, 700, and 784 (no reduction) singular values were all tested, and lambda values of tenths from 0.1 to 1.0 were also tested.

Neural network

Neural networks can be thought of as collections of classifiers, taking an inner product between some weights and input vector as input and then outputting a single value. These classifiers are nodes. These nodes can then be arranged into layers, and multiple layers can be stacked in sequential order. Each node can apply a different function to its inner product input, however, called an activation function. Some example activation functions are shown below, where x is a vector of inputs and w is a vector of weights for that node.

Eq (3) Rectified Linear (ReLU) activation: $max\{0,\ x^T w\}$

Eq (4) Sigmoid activation: $\frac{1}{1+e^{-x^T w}}$

Eq (5) Softmax activation: $\frac{e^{x^T w}}{\sum_{j=1}^{K} e^{x^T w_j}}$ where K is the number of classes.

For this project, three neural networks were developed. Model 1 had a single 10-node softmax layer. Model 2 had a 784-node ReLU layer followed by a 10-node softmax layer. Model 3 had a 784-node sigmoid layer followed by a 10-node softmax layer. The parameters modified during cross-validation included which model was used and the number of training epochs. This was done across five subsets of the training images, as with the ridge classification algorithm, and epochs from one to ten were tried.

Results

K-means

Again, no true cross-validation was performed for this algorithm due to the limitations discussed during the introduction to the K-means algorithm, but four configurations of this algorithm were evaluated for accuracy on the training images. Training images were used to train and classify these configurations. The four configurations were 1) ten clusters, initialized randomly, 2) twenty clusters, initialized randomly, 3) ten clusters, chosen initially, 4) twenty clusters, chosen initially. For the ten chosen initial clusters, one of each digit was chosen. For the twenty chosen initial clusters the counts of each digit were 1 zero, 1 one, 1 two, 1 three, 2 fours, 3 fives, 3 sixes, 3 sevens, 4 eights, and 1 nine.

The fourth configuration, twenty chosen initial clusters, was used for final testing of the classifier. Despite having a slightly lower accuracy than twenty random clusters, the fourth configuration had every digit represented by at least one centroid. Only results from classifying the test images will be provided for this classifier since no cross-validation was done for this algorithm. Test results are below in Table 1.

Table 1: Test results for a K-means classifier with twenty centroids and chosen initial centroids.

Recall	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine	Overall Accuracy
Test	0.65	0.99	0.56	0.80	0.28	0.28	0.76	0.81	0.74	0.50	0.64

These results were more or less identical to the training results, and so there was no concern that overfitting occurred. As noted above, some digits are much harder to classify due to their similarity to other digits, such as four and five.

Ridge Classification

Two cross-validations were done for this algorithm, one for the number of singular values to keep when reducing the table, and one for the regularization parameter. Cross-validation found that accuracy was unaffected by the number of singular values retained, after about 100 singular values. The regularization parameter had no impact on cross-validation accuracy. From these cross-validation results, it was chosen that for final train/test results, the model would keep 100 singular values and have a lambda value of 1.

Table 2: Training and test classifier recall for the ridge classifier with data reduced to 100 singular values and a regularization parameter of 1.

Recall	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine	Overall Accuracy
Train	0.95	0.97	0.79	0.83	0.88	0.68	0.93	0.88	0.76	0.81	0.85
Test	0.95	0.97	0.80	0.87	0.90	0.68	0.93	0.88	0.78	0.82	0.86

Again, train and test results were very similar, so there was no concern of overfitting. Performance compared to K-means, however, was improved, both in terms of recall of individual digits and overall accuracy. Again, there were several digits that the algorithm struggled to classify accurately, most noticeably five.

Neural network

For the neural network, cross-validation was performed over the three models and for the number of training epochs. Models 2 and 3 performed better than model 1, regardless of the number of training epochs. Isolating for just the number of training epochs, models 2 and 3 were most accurate when trained over five, six, seven, or ten epochs. For the final classifier, model 2 was used with five training epochs. Results are below in Table 3.

Table 3: Train/test results for the neural network classifier. Results below are the classifier's recall for each digit.

Recall	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine	Overall Accuracy
Train	0.96	0.96	0.95	0.89	0.92	0.93	0.99	0.94	0.93	0.94	0.94
Test	0.96	0.96	0.94	0.87	0.91	0.92	0.97	0.91	0.92	0.92	0.93

This was by far the most accurate classifier, and did not struggle with digit recall like the first two classifiers.

Strengths and Limitations

K-means

One strength of the K-means algorithm is that it is unsupervised and so does not require labeled training samples. This is useful for applications where the classifications may be unknown, or to help find new classes that had not been considered before. A limitation of the K-means algorithm is that its solution surface is not convex. This means that the solution could converge to a local minima rather than the global minima, and may not be the most accurate solution.

Ridge Classification

A strength of ridge classification is the regularization by the two-norm of the weights. This helps to prevent overfitting during training. The squared error loss could also be a limitation to ridge classification in other applications, however. If data points are very spread out, those large distances will affect the decision boundary disproportionately, and could lead to misformed decision boundaries.

Neural network

The biggest strength of neural networks is their flexibility. Besides being flexible enough to be used for almost any project, they can also provide multiple outputs at one time, as was the case with the neural networks used in this project. A limitation is their complexity. Models 2 and 3 in this project both over 600,000 weights that needed training, and they are very simple neural networks. Many networks have millions of weights to train.

Conclusion

The three algorithms used in this project were all optimized with regards to a couple of parameters each, before being used to classify a set of withheld MNIST test images. The K-means algorithm struggled due to the similarities in features between many of the digits. The ridge classifier performed much better, but also struggled with the proximity of the digits. The neural network performed the best, by far, when organized as a 784-node ReLU layer followed by a 10-node softmax layer.