Assignment 4: Classifying with K-Means and De-noising with an Autoencoder

Amila Ferron
aferron@pdx.edu

1 K-Means Clustering

K-Means clustering was performed on 1000 images from the MNIST data set, with 100 examples of each digit.

Initial clustering showed defined clusters, but labels were not aligned with the actual numbers. Likely label/class matches were identified and the adjusted confusion matrix was plotted as shown in Figure 1. This shows 9's are often misclassified as 4's or 7's, and 5's are misclassified as 3's, 1's are misclassified as 5's, and 0's are misclassified as 9's. These types of misclassifications are believable and indicate that the rearrangement of the columns was likely in a valid arrangement.

The accuracy of the clustering after adjusting the columns was 0.5550. In comparison, the results using a Neural Network for written digit recognition achieved an accuracy of 0.8847. This used a Neural Network with a learning rate of 0.01, batch size of 10, ReLU activation, and pollution. K-means clustering appears to be far less accurate than Neural Network classification, because it is an unsupervised learning method and the number of examples was relatively small. This exercise shows the limitations of unsupervised learning, but also the possibility because some level of accuracy was able to be achieved, despite the limitations of the methods.

2 K-Means Clustering Using Feature Vectors

Two methods were used with K-Means clustering using embeddings from an Autoencoder.

2.1 K-Means with Encoded Images

An Autoencoder was used to create embeddings that could replace MNIST images for classifying with K-Means clustering. The Autoencoder was trained on 1000 images of each of the 10 digits, then each of the images was fed into the encoder to produce 8x8 pixel embeddings. These were then fed into K-Means clustering and classification results were plotted as in Section 1. Classification was done on embeddings from two Autoencoders – one trained for 5 epochs and another for 20 epochs. Results are in Figures 2 and 3.

The accuracy using this method was 0.5477 for the Autoencoder trained for 5 epochs and 0.5322 for the Autoencoder trained for 20 epochs. The better performance of the first model and the lower performance overall using the Autoencoder may be due to variation in clustering since K-means clustering itself can vary depending on the way the points are initialized. However, the slightly reduced performance with the Autoencoder may be related to the difference between the original and encoded images. It seems like creating an embedding of the image would improve performance when clustering because properties of the image that make it reproducible have been selected by the encoder, but it is also possible that the reduction in size makes the classification slightly more difficult because there is less information to work with. It is interesting to see that the results can be so similar when classifying a 28x28 original image and an 8x8 embedding despite the reduction in pixels from 784 to 64.

2.2 K-Means with PCA on Encoded Images

Classification with K-Means was done using dimensionally-reduced embeddings. PCA was applied after producing embeddings, and the reduced-dimension embeddings were passed to the K-Means classifier. Several trials were run with varying number of components and the optimal number was selected for this trial. The number of components was set to 32 and the Autoencoder was trained for 20 epochs before producing the embeddings. The accuracy using this method was 0.5708. This is an improvement from other combined mod-

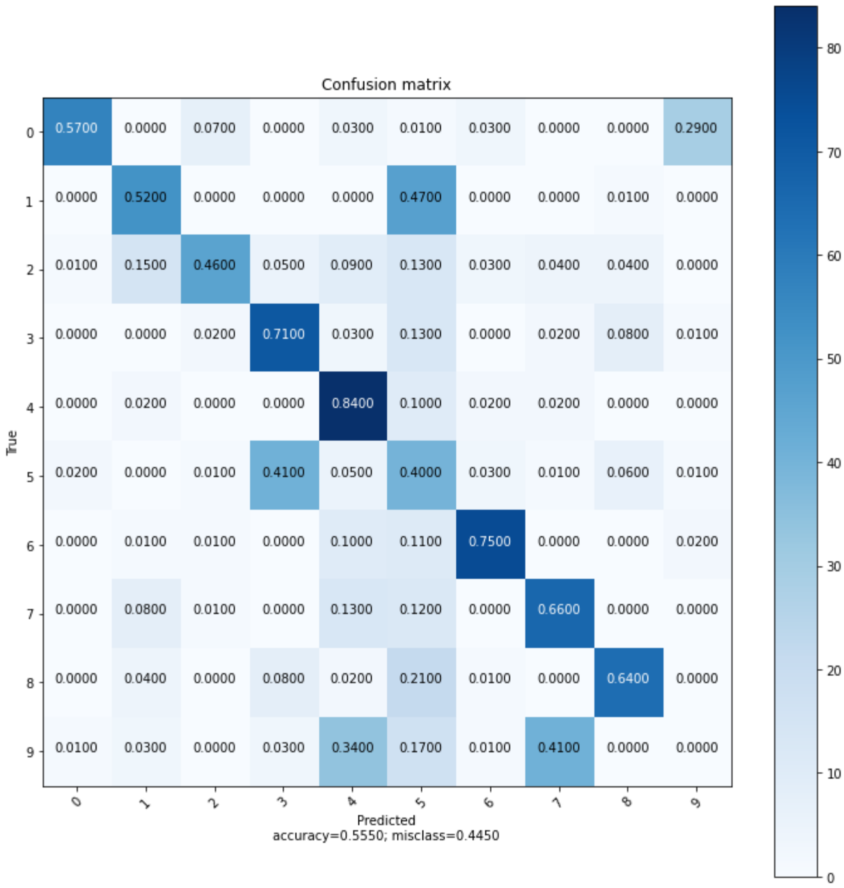

Figure 1: Adjusted clustering of MNIST images has accuracy 0.5550.

els because the embeddings and the PCA remove extraneous information that isn't useful to identify differences between digits. Once the image representations are pared down, K-Means is able to more successfully separate the images into classes representing each of the digits. Results are shown in Figure 4.

3 De-Noising with an Autoencoder

An Autoencoder was used to remove noise from images. A training set was created by adding noise at random locations with random shades to MNIST images at a set percentage of pixels. The Autoencoder was trained by feeding in the training images and comparing the output images with the images before noise was added. Five different rates of adding noise were used, ranging from 20 - 70%. In this process, the Autoencoder creates an embedding at its narrowest point of the most relevant information from the noisy image in order to recreate the original image. The results are shown in Table 1. The de-noised images do resemble the original images and are legible as digits, up to and including about the noise addition rate of 50%. At 60%, it becomes unclear which digit is represented by the image.

The Colab notebook for this project follows tutorials linked in the notebook and is **here**.

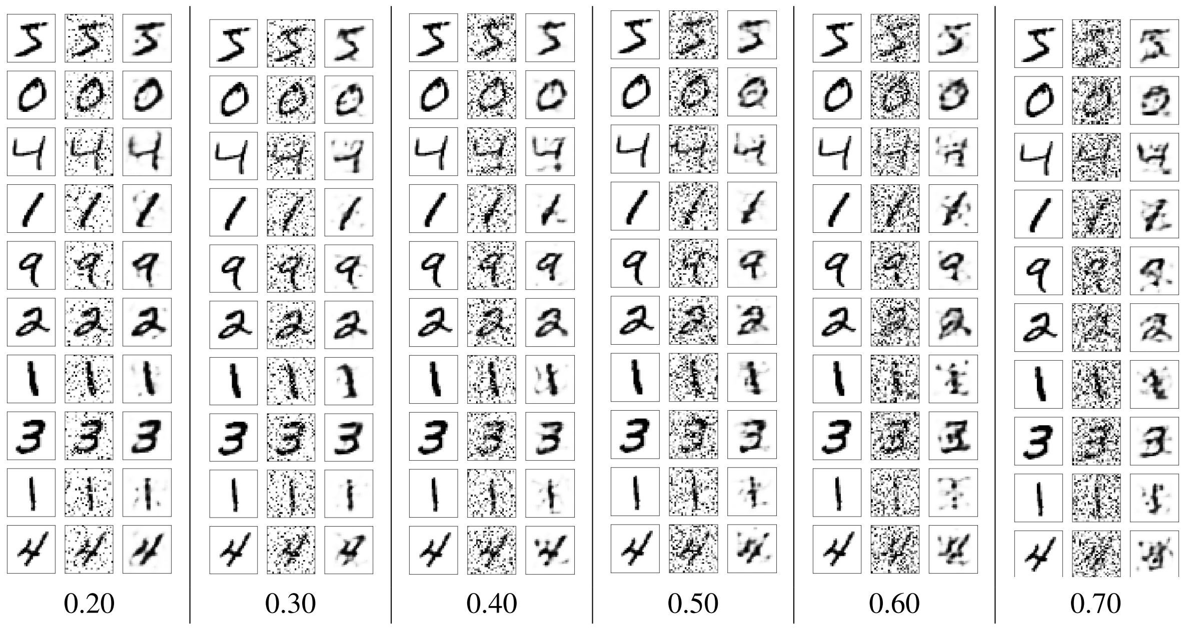

Table 1: Images before noise is added, after noise is added, and after repairing the noisy image with the Autoencoder. The fraction of pixels that were altered in adding noise is below each set of images.

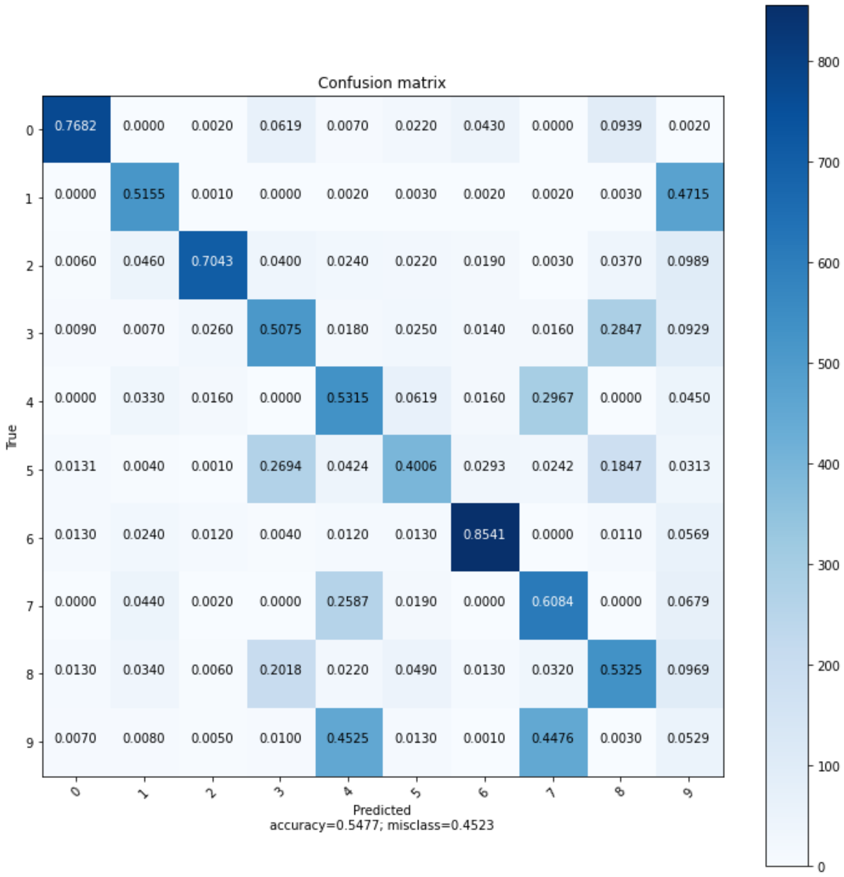

Figure 2: Results using K-Means clustering with embeddings from an Autoencoder trained for 5 epochs.

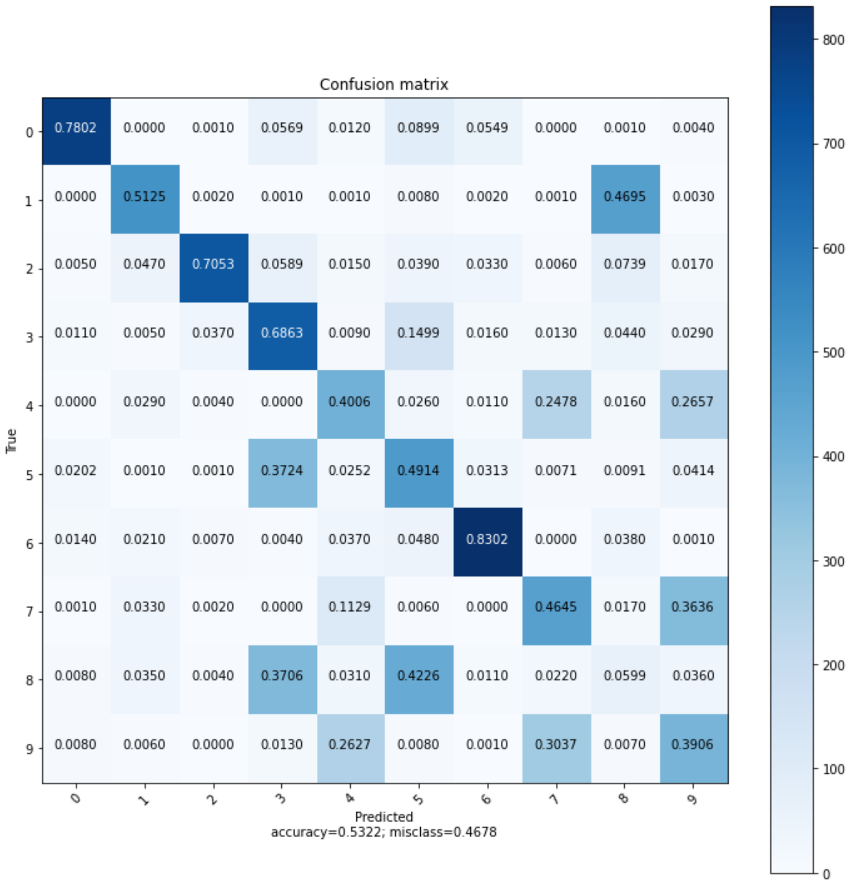

Figure 3: Results using K-Means clustering with embeddings from an Autoencoder trained for 20 epochs.

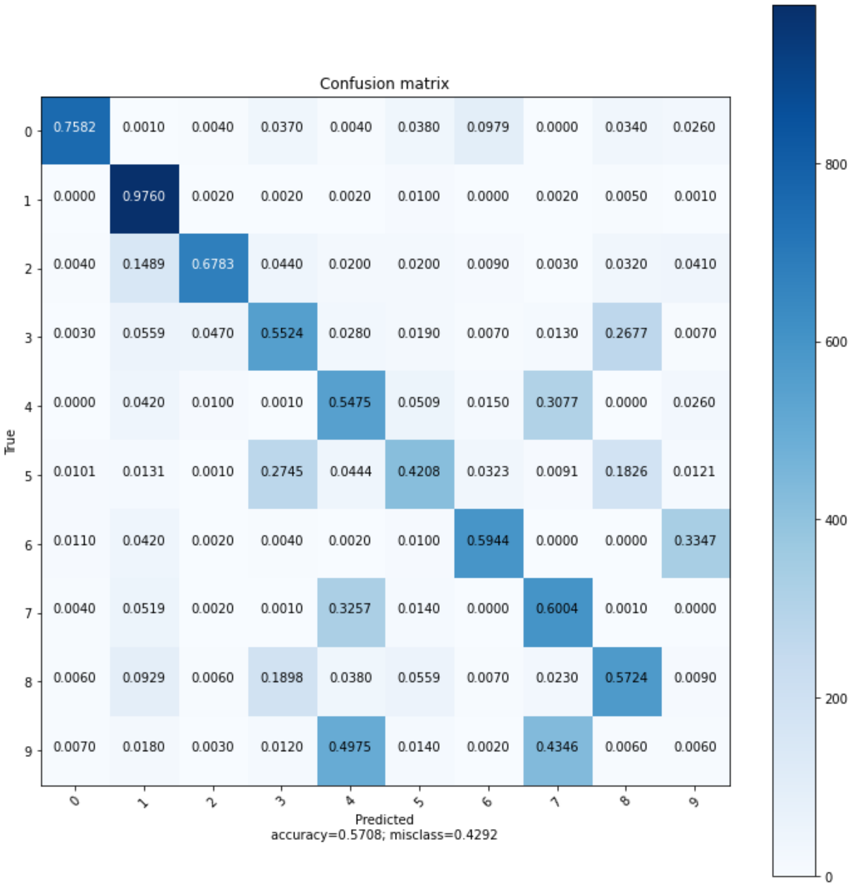

Figure 4: Results using PCA and K-Means clustering with embeddings from an Autoencoder trained for 20 epochs.