Dimensionality Reduction: Removing Noise with SVD

By Sam Watson

Overview

At first glance, it seems that the more data and features we have, the more information we can obtain from the dataset. However, the complexity of datasets increases exponentially as more features are added. The problems that occur when working with data in the higher dimensions is called “The Curse of Dimensionality.” Dimensionality reduction is a set of techniques used to combat this.

The process of reducing or compressing a dataset into as fewest coordinates as possible is known as dimensionality reduction. There are several reasons why a more compact representation is more favorable. It can reduce the overall size of the dataset, it can make the data more comprehensible, and it can help with visual representations.

Singular Value Decomposition

Singular value decomposition is the process of factoring a matrix (M) into 3 separate matrices, conventionally named U, Sigma, and V. If M matrix is of size (m x n), then U will be (m x m), Sigma will be (m x n), and V will be (n x n). U and V are orthogonal matrices. Sigma is a diagonal matrix, where the diagonal entries are in descending order. These values are denoted as singular values, and show the magnitude of vectors in U and V. Through matrix multiplication, matrix M can be projected back into the same dimensional space with only a selected number of principal components.

Dataset:

MNIST: dataset of hand drawn digits

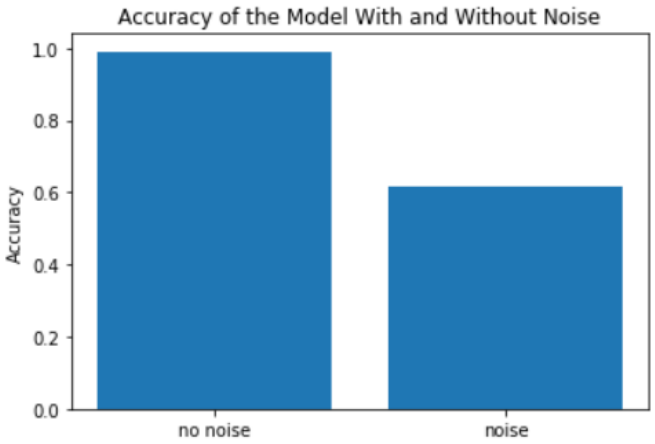
- Target: 0 through 9
- 70,000 samples
- 28 x 28 images (784 features)

Baseline Model:

The baseline model used was a convolutional neural network, sourced from “Deep Learning with Python” by Francois Chollet.

Summary:

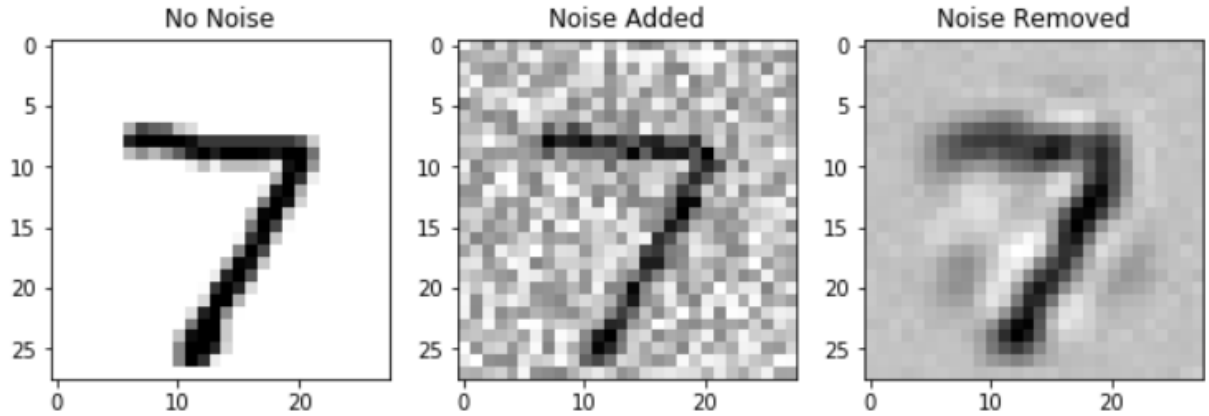
First, I trained a convolutional neural net on a subset of the MNIST data (60,000 samples). The model was tested on the remaining 10,000 samples, and retrieved an accuracy of 98%. Nice! This baseline model is extremely effective on this dataset.



But what happens when the data images are blurry, or hard to read? Random noise was added to the test data, and then tested on the same model. After adding the random noise, the accuracy of the model dropped to 60%.

This is where dimensionality reduction comes in to play. Maybe most of the noise added to the image is not represented in the first 50, or 100 principal components. If this were so, than singular value decomposition (SVD) can be used to split the matrix into three separate matrices (U, Sigma, and V), and then projected back into the same dimension with only a selected number of principal components.

Below are three images of the same test image of a hand-written digit "7." One is the image without noise, one with the noise added, and finally one with the noise removed using SVD. The noise was removed by projecting the dataset back into the same dimension and only keeping the first 40 principal components.



The test dataset was flattened out, turning each of the 10,000 28 x 28 images into 10,000 vectors of length 784 (28 x 28=784). Using SVD, the test dataset matrix of size [10,000 x 784] was split into three matrices of size [10,000 x 10,000], [10,000 x 784], and [784 x 784]. Next, the matrix was projected back into the same dimension [10,000 x 784], but only keeping a specified number of principal components. If I were to project the dataset back with only the first 40 principal components, for example, then the sizes of the matrices would look like [10,000 x 40] (U), [40 x 40] (S), and [40 x 784] (V).

Findings:

SVD was the right dimensionality reduction technique for this dataset and problem. However, it was only able to do so much to improve the accuracy of the model on the blurry images. No matter how many principal components you keep in the projection, the model never gets back to the 99% from the non-noisy test images. It does, however, get back to about 85% when keeping the first 40 principal components. This appears to be the "sweet spot" of just the right number of principal components to

remove the noise while keeping the overall essence of the image. Keeping less than 40 principal components will take away too much of the hand-written digit's qualities. The only instances when the images perform worse on the model than the noisy images is after keeping anything less than the first 5 principal components. Any principal component number after 40, the model's accuracy gradually declines to 60% again. After all, there are only 784 principal components in the dataset, and keeping a number close to that won't remove enough of the noise from the dataset. The below graph summarizes these findings, showing the accuracy of the model for the projection of the dataset with the selected number of principal components.