SVD-based Principal Component Analysis and Image Decomposition

Jonathan Ang, Marley Abowitz, Alexander Liu, Grayson Newell

1 BACKGROUND

1.1 Introduction

Images used for machine learning (ML) modeling often use more data than necessary, causing ML applications to be more expensive. This project will optimize ML modeling using singular value decomposition (SVD) to perform principal component analysis (PCA) and image decomposition. We will then apply our implementation to the MNIST dataset to test the effect of reducing the dimensions of the data.

1.2 Singular Value Decomposition

SVD is a central matrix decomposition method. The SVD of matrix A represents a linear mapping $\phi : V \rightarrow W$. The SVD quantifies the geometrical change between the two vectors.

1.3 Principle Component Analysis & Linear Dimensionality Reduction

High-dimensional data, such as images, is difficult to analyze, interpret, and visualize and it is expensive in terms of storage. To solve these problems we use linear dimensionality reduction to exploit the properties of high-dimensional data that lie in low-dimensional subspaces to compact it without losing information. PCA is an algorithm for linear dimensionality reduction. PCA finds the projections $\tilde{x}_n$ of data points x_n, maximizing their similarity to the original data.

PCA works by transforming the original data into a new coordinate system such that the greatest variance lies along the first axis (first principal component), the second greatest variance along the second axis (second principal component), and so on. These principal components are orthogonal to each other, meaning they capture the maximum amount of variance in the data while being uncorrelated. The compressed data is reconstructed into $\tilde{x}_n$ which is a lower-dimension representation of x that lives in the original data space. This is achieved by finding the eigenvectors (principal components) of the covariance matrix of the data. The eigenvectors with the largest corresponding eigenvalues capture the directions of maximum variance in the data. Mathematically, PCA aims to maximize the variance of the projected data onto each principal component. This is equivalent to minimizing the reconstruction error when the data is projected onto a lower-dimensional subspace spanned by the principal components. The importance of principal components in data representation and feature extraction lies in dimensionality reduction, noise reduction, feature extraction, and visualization.

1.4 MNIST Dataset

The MNIST dataset is a set of images of digits 0-9, containing 70,000 images of handwritten digits. Each entry is a greyscale image (0-255) of the size 28 x 28, containing 784 pixels. Each image is thus in a vector $x \in \mathbb{R}^{784}$. The challenge posed by this dataset is it's high-dimensional qualities.

2 METHODOLOGY

2.1 Matrix of Data

The first step is representing the MNIST dataset as a matrix A with 60,000 column vectors. The 60,000 column vectors corresponds to the first 60,000 images we will use as the training set. Each column will be a 784-dimension representation of every pixel in the 28x28 image. PCA will be performed on matrix A. This process is then implemented using a data analysis library called Scikit-Learn.

Matrix A is then normalized into A_0, where each column has a mean of zero and a standard deviation of 1. This is done for each column v by first calculating the mean $\bar{x}_v$ and standard deviation σ_v of all values in v. Then, the mean is subtracted from every pixel value, and the difference is divided by the standard deviation as shown below. In Equation 1, w represents the column vector in A_0 corresponding to v in A.

$$w_i = \frac{v_i - \bar{x}_v}{\sigma_v} \tag{1}$$

2.2 Covariance Matrix

The next step of PCA is to convert the normalized dataset matrix A_0, into its covariance matrix Σ. A covariance matrix provides a numerical representation of the relationship between rows in an original given matrix. Every entry in A_0 can be related to entries in A using the formula in Equation 2.

$$cov_{x,y} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{N - 1} \tag{2}$$

In Equation 2, x and y are each features, or rows in A_0. x_i and y_i are entries from these rows, $\bar{x}$ and $\bar{y}$ are the row means, and N is the number of entries in x and y, 60,000 in the case of the MNIST dataset.

Equation 2 demonstrates how covariance between two features or rows is calculated, but can easily be applied to an entire matrix as shown in Equation 3.

$$\Sigma = \frac{A_0^T A_0}{N} \tag{3}$$

In Equation 3, A_0 is the zero-centered representation of A, obtained by subtracting the dataset mean from each entry. This ensures that A_0 has a mean of zero, which is essential for PCA to eliminate offset in the data. Furthermore, N is again the number of image vectors (columns) in A.

2.3 Matrix Decomposition

Then, the eigendecomposition of Σ is found by first solving the characteristic equation shown in Equation 4, for its eigenvalues λ.

$$|\Sigma - \lambda I| = 0 \tag{4}$$

Once this has been solved for a series of λ_i, these values are substituted into Equation 5, which can then be solved for corresponding eigenvectors u_i.

$$|\Sigma - \lambda_i I| u_i = 0 \tag{5}$$

Now that all pairs of eigenvalues and eigenvectors of Σ have been found, the spectral theorem dictates that $\Sigma = U \Lambda U^T$, where U is a matrix composed of all eigenvectors as columns and Λ is a diagonal matrix of the eigenvalues. Note that each pair should be listed in descending order based on its eigenvalue. This relation can then be expanded to show each column in Σ as a product of column vectors, as demonstrated in Equation 6. (Where u_i, λ_i, and v_i are columns from U, Λ, and U^T respectively.)

$$A = u_1 \lambda_1 v_1 + u_2 \lambda_2 v_2 + \ldots + u_n \lambda_n v_n \tag{6}$$

2.4 Extracting Principal Components

Assuming that the eigenvalues $(\lambda_1, \lambda_2, ..., \lambda_n)$ are in descending order, $u_k \lambda_k v_k$ is the k-th Principal Component of A. As each of these is orthogonal, the first m Principal Components form an m-dimensional basis upon which the data can be represented. Note that this "PCA Space" will have a lower rank than the original dataset, consistent with the goal of dimensionality reduction.

To demonstrate this, the left graph of Figure 1 shows a dataset of blue points and their first two principal components as green arrows. The first principal component is the green arrow pointing towards the bottom right. The next principal component is orthogonal to the first principal component and points to the top right instead. It is clear that each arrow can be formed into an axes upon which the data varies the most. The data can then be projected onto the axis of the first principal component, resulting in the right graph of Figure 1.

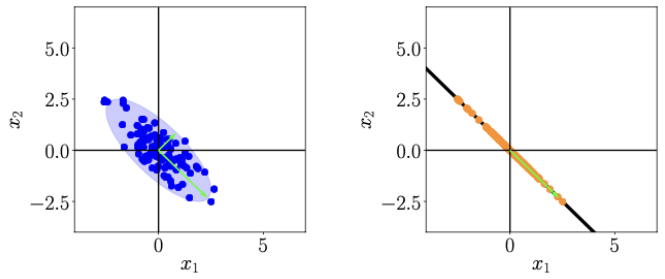

Figure 1: Left: Dataset with first two principal components shown in green; Right: Dataset projected onto PCA space; page 337.

This suggests similarity to linear regression, which is accurate given that both are techniques of representing trends in data.

2.5 PCA Spaces

These PCA Spaces are essentially rank-m approximations of the original data, retaining its most important characteristics in less space. Thus, as m increases, PCA becomes more accurate but less concise. Figure 2 shows several images from the MNIST dataset and their reconstructions at varying values of m.

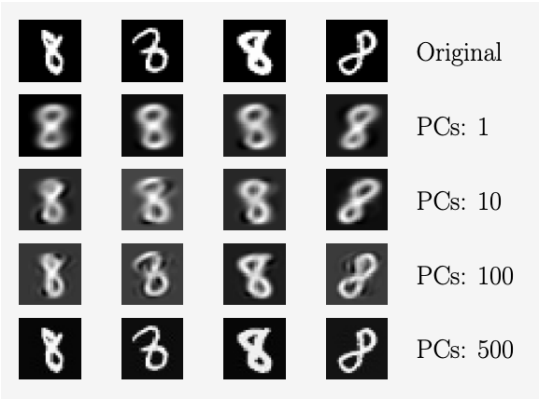

Figure 2: Image detail retained by more principal components; pg 338.

This brings into question what value of m maximizes the accuracy with which the PCA space represents the data while minimizing its dimensions.

3 IMPLEMENTATION

We first import the MNIST data set and split it into our training set (the first 60000 images) and testing set(last 10000 images). As shown in Listing 1, PCA can be easily implemented in code using scikit-learn where we can specify the number of dimensions we want using the ncomponents parameter. In this example we use 100 components. It is also important to note that scikit-learn first automatically centers the dataset around the origin.

Listing 1: Using Scikit-learn to import MNIST dataset and reduce it dimensions to 100 using PCA

```
from sklearn.decomposition import PCA
from sklearn.datasets import fetch_openml

mnist=fetch_openml('mnist_784',as_frame=False,parser="auto")
X_train, y_train=mnist.data[:60_000],mnist.target[:60_000]
X_test, y_test = mnist.data[60_000:],mnist.target[60_000:]

pca = PCA(n_components = 100)
X_reduced = pca.fit_transform(X_train)
X_test_reduced = pca.transform(X_test)
```

In Figure 3, the Principal Components is then printed by the line of code at the top. This outputs an array of 100 elements, with each element representing a principal component. The principal components are also represented as a 784-dimensional array of the principal component's coefficients.

```
print(pca.components_)

[[ 8.51472758e-19 -3.80214801e-19  1.35747413e-18 ... -0.00000000e+00
  -0.00000000e+00 -0.00000000e+00]
 [-9.82699237e-18 -5.36491700e-18  3.18199534e-17 ... -0.00000000e+00
  -0.00000000e+00 -0.00000000e+00]
 [-1.56287635e-17  1.09800375e-18  1.90748255e-17 ... -0.00000000e+00
  -0.00000000e+00 -0.00000000e+00]
 ...
 [-1.38125196e-17  2.02823577e-17 -1.05930279e-17 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 1.29702139e-17  1.19551854e-17  1.69319574e-17 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 4.10688216e-17  8.60925129e-18 -6.48985907e-18 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]]
```

Figure 3: Printed result of array of 100 Principal Components

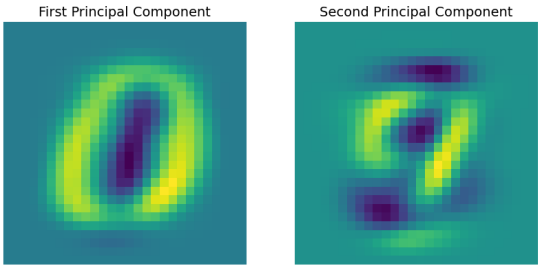

Figure 4: The first 2 principal components when converted to an image

We use the first two principal component and reconstruct an image of it, resulting in the images shown in Figure 4. In the first PC, the lighter colours represents a higher coefficient, suggesting that the ring around the center consists of the pixels who's values correlate strongly together. On the other hand, the pixels outside the ring are a neutral color representing mostly 0 values. This is as expected because these pixels are almost always white (0 coefficient and 0 when normalized). This highlights the power of PCA, it is able to differentiate the pixels which don't matter and the pixels that are strongly correlated. ML

models do not need to process the data of all pixels, but only need to look at these trends.

4 APPLICATION TO ML MODELLING

After creating using PCA to reduce the dimensions of our dataset, we then train an ML Model, Stochastic Gradient Descent (SGD) Classifer. We test the effect of reducing dimensions on the accuracy of the model and the time it takes to train it on the first 60000 images.

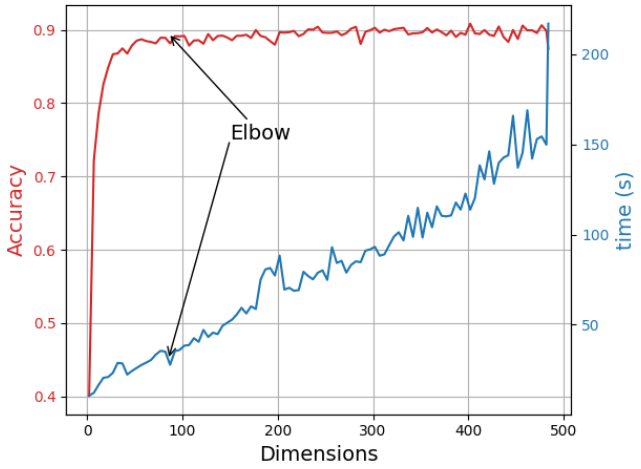

Figure 5: Graph plotting number of dimensions of dataset against accuracy level and time taken to train the SGD model

5 EVALUATION

As shown in Figure 5, we found that 85 dimensions is the elbow curve for accuracy and reduction, meaning we can reduce to 85 dimensions without overly compromising accuracy. We can also see that PCA enabled the model to achieve the highest accuracy of over 0.90 at around 400 dimensions. When compared to the results at 484 dimensions, where PCA was not used at all, the model took a much longer time to train at over 3 minutes whilst also achieving a far lower accuracy at 0.874. This suggests that PCA is useful for not only improving training times but also reducing over-fitting by the SGD model.

With no PCA, the apparent elbow at around 85 dimensions demonstrates that there will be no further substantial improvement in performance with every added dimension going forward. However, with PCA, there is a clear positive linear relationship between dimensions and accuracy, at least within the scope of our numbers, which eventually allowed it to surpass the performance of the original model.

6 CONCLUSION

We find that PCA is a useful tool for ML models. The benefits are twofold. Firstly, PCA reduces the data size of data sets so that less computational power is require to train ML models on it. This is especially important for larger and more complex computer vision problems. Secondly, PCA provides insights into the underlying structure of the data, enabling researchers to uncover latent patterns and relationships that may not be apparent in the original dataset.

Due to the scope of this project, we have only used 1 ML model, SGD Classifier. Furthers areas of research include figuring out how PCA affects other models such as Random Forest Classifier.

7 REFERENCES

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). Mathematics for Machine Learning. Cambridge University Press.

Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.