Homework 2 - Numerical Methods

Maria Pana, 315 CA

May 20, 2023

1 Project description

Description. The aim of the program is to recognize hand-written numbers from the MNIST dataset. To do so, the image is first compressed using **Singular Value Decomposition (SVD)** and **Principal Component Analysis (PCA)**. Irrelevant data is removed as a result, which implies storing less data while still generating accurate predictions, but also an increased program efficiency. Lastly, the program uses the **k-nearest neighbour** algorithm to make the final predicitions.

2 Compressing images with SVD

Images can be stored in matrices, where each pixel represents the color or the intensity of light in a specific location.

A matrix $A \in \mathbb{R}^{m \times n}$ is a grid of numbers consisting of m rows and n columns. Phenomenologically, any matrix describes some linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$ by specifying through its columns where the basis vectors should land relative to the initial basis.

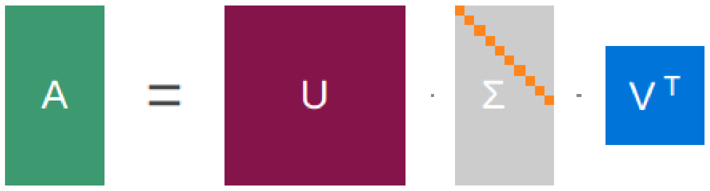
In other words, linear transformations can rotate, flip and stretch space. In fact, a matrix may represent multiple actions performed on the space.

2.1 How SVD works

In the case of SVD, it is considered that any matrix $A \in \mathbb{R}^{m \times n}$ can be written as the following product:

$$A = U \cdot \Sigma \cdot V^T \tag{1}$$

where $U \in \mathbb{R}^{m \times m}$ and $V^T \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{m \times n}$ is a rectangular, diagonal matrix (the elements on the diagonal are the singular values of A).



This equation is another form of saying that the output of the linear transformation represented by A can be reached by applying the other transformations consecutively.

To be precise:

V^T rotates space. Its row vectors are also known as *right singular vectors*.

Σ scales space (relative to A's singular values).

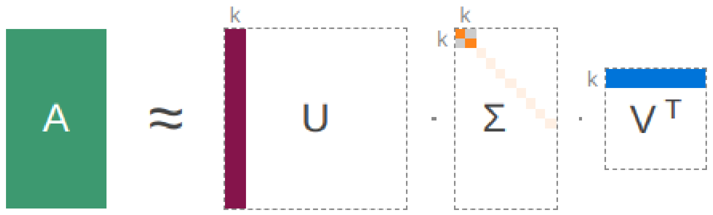
U rotates space once again. Its column vectors are also called *left singular vectors*.

The greatness of SVD lies in its ability to provide a compact representation of the original matrix. The singular values in Σ are arranged in descending order on the diagonal, allowing the program to identify and retain the most important components while discarding the less significant ones.

2.2 Using SVD in image compression

To compress an image, the program applies **SVD** to its correspondent matrix. The accuracy of the approximated image is determined by the number of singular values k used (the larger k, the better image quality, but also the more data needing to be stored).

Once U, Σ and V are computed, the program extracts the first k columns of U and V and the upper left kxk square of Σ (the k largest and most important singular values).



Consequently, the storage needed for the resulted approximation is

$$m \cdot k + k + k \cdot n = k \cdot (1 + m + n) \qquad (2)$$

To illustrate the power of image compression using SVD, let's consider the following grayscale 350x530 image of Mona Lisa:

The original matrix of the image is 350x530. Therefore, there would be $350 \cdot 530 = 185500$ values to store. However, for $k = 30$, after SVD, there would

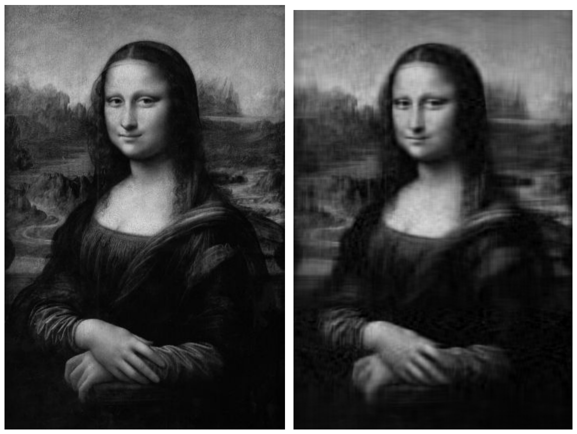


Figure 1: The original image of Mona Lisa (left). The approximated image after SVD (right)

be only 26430 values left to work with (roughly 14% of the original number of values).

In the program, this is done through the *task*1 function.

3 Compressing images with PCA

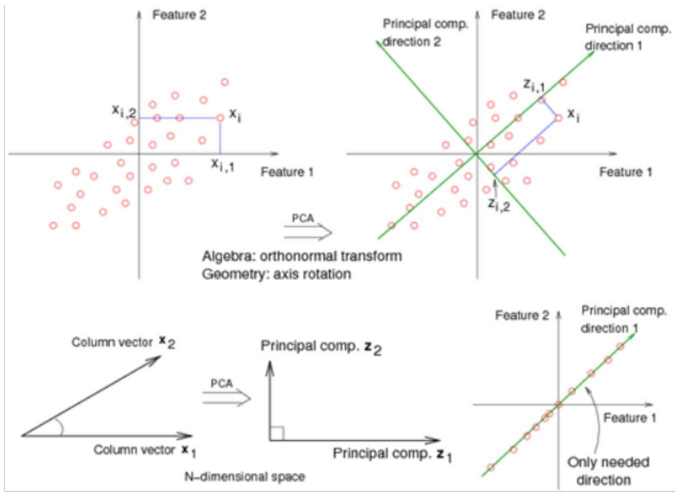
Principal Component Analysis (PCA) means flattening (reducing the dimensionality) of a dataset while preserving the maximum variation. It transforms the original set of correlated variables into a smaller number of uncorrelated variables called *principal components*. This is done by finding the directions of maximum variance in the data set and projecting the data onto these directions.

Since PCA directions are highly sensitive to the scale of the data, the program first goes through the standardization stage: transforming the data to comparable scales in order to avoid variables with large ranges dominating over the others. To avoid such scenario, the *task*2 function substracts the mean of each line in the image matrix from the line itself.

It is also important to note that highly correlated variables contain redundant information. These correlations are determined by computing the symmetric *covariance matrix*.

$$Z = \frac{A^T}{\sqrt{n-1}} \tag{3}$$

The sign of the covariances in the matrix indicates the relationship between two variables. If the sign is positive, then they increase or decrease together (correlated). If it is negative, one decreases while the other increases i.e. they are inversely correlated.



Feature 2
$x_{i,2}$
x_i
$x_{i,1}$
Feature 1
Principal comp. direction 2
Feature 2
Principal comp. direction 1
$z_{i,1}$
x_i
Feature 1
$z_{i,2}$
PCA
Algebra: orthonormal transform
Geometry: axis rotation
Column vector $\mathbf{x}_2$
Column vector $\mathbf{x}_1$
PCA
Principal comp. $\mathbf{z}_2$
Principal comp. $\mathbf{z}_1$
N-dimensional space
Feature 2
Principal comp. direction 1
Feature 1
Only needed direction

Figure 2: Visualizing PCA in a 2D example

The program then determines the principal components (in descending order) by applying SVD to the covariance matrix. It keeps only the first (most significant) k ones in matrix W (k-sized space of principal components) and then projects the standardized matrix onto the new space, creating matrix Y.

All that is left is to compute the final data set:

$$Y = W^T A \tag{4}$$

$$A_{new} = WY + \mu \tag{5}$$

where μ is the vector containing the means of each line in the original matrix.

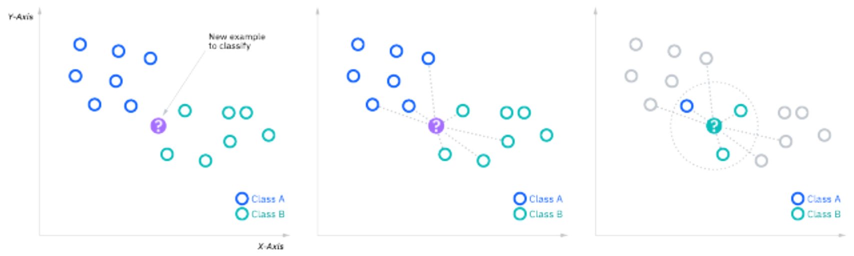
An alternative way to determine the principal components is implemented in the *task*3 function. In this scenario, the covariance matrix is:

$$Z = \frac{A * A^T}{n-1} \tag{6}$$

The function then computes the eigenvalues and eigenvectors of the covariance matrix, which will lead to computing matrix W the same as in the *task*2 function.

4 Classifications and predictions using KNN

To predict what number is in the image, the program uses the **K-Nearest Neighbours** algorithm. Its main objective is to use the k ”closest” images to the one given to make the classification. For the purpose of this project, we chose k to be 5.



However, the program must first transform matrix Y (computed previously) into a vector and compute the euclidian distances between this new vector and the tested vectors. Only the smallest 5 distances are kept. Calculating the median of the vector of labels for the 5 found images will generate the final prediction.

5 Bibliography

- https://www.youtube.com/watch?v=vSczTbgc8Rc
- http://timbaumann.info/svd-image-compression-demo/
- https://setosa.io/ev/principal-component-analysis/
- https://online.stat.psu.edu/stat508/book/export/html/657
- https://www.ibm.com/topics/knn