Classification of MNIST Digits with SVD Decomposition

In this homework the Singular Value Decomposition is used to classificate MNIST digits. This dataset, which is loaded in the form of a 256×1707 matrix X, contains the flattened version of 1707 16×16 grayscale handwritten digits between 0 and 9.

A binary classification problem was considered, where it was required to classificate if a given digit of dimension m x n represents the number 3 or the number 4 whose class is C1 and C2 respectively. Right after that, the obtained dataset is split into training and test sets. The training set is then split once more into X1 and X2, the matrices such that their columns are a flatten version of each digit in C1 and C2 respectively. Considering the SVD of the two matrices,

$$X1 = U_1 \Sigma_1 V_1^T$$
$$X2 = U_2 \Sigma_2 V_2^T$$

and given that

Theorem 1. *Let's consider W a subspace of $\mathbb{R}^n$ where dim $W = s$ and $\{w_1, \dots, w_s\}$ an orthogonal base of W. Given a generic $y \in \mathbb{R}^n$ we have that the projection $y^\perp$ of y onto W has the following form:*

$$y^\perp = \frac{y \cdot w_1}{w_1 \cdot w_1} w_1 + \cdots + \frac{y \cdot w_s}{w_s \cdot w_s} w_s. \tag{1}$$

Corollary 1.1. *If $X \in \mathbb{R}^{m \times n}$ is a given matrix with SVD decomposition $X = USV^T$, since the $p = rank(X)$ is the dimension of the space defined by the columns of X and the columns of U, $\{u_1, \dots, u_p\}$ are an orthonormal basis for that space, the projection of an n-dimensional vector y on this space can be easily computed as:*

$$y^\perp = U(U^T y) \tag{2}$$

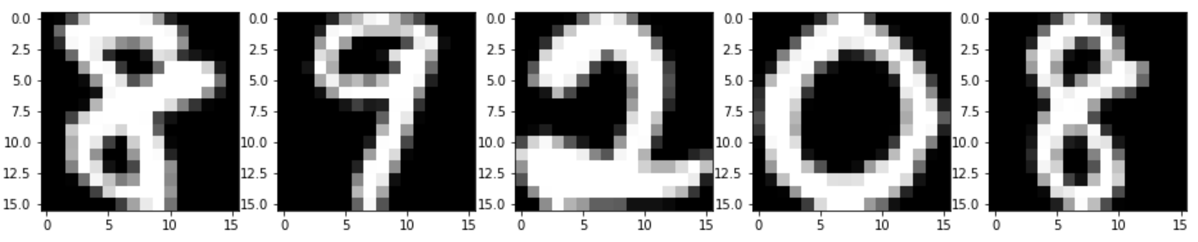
then if y is a new unknown digit, it can be classified by first flatten it to a vector and then projecting it to the spaces of X1 and X2:

$$y_1^\perp = U_1(U_1^T y)$$
$$y_2^\perp = U_2(U_2^T y)$$

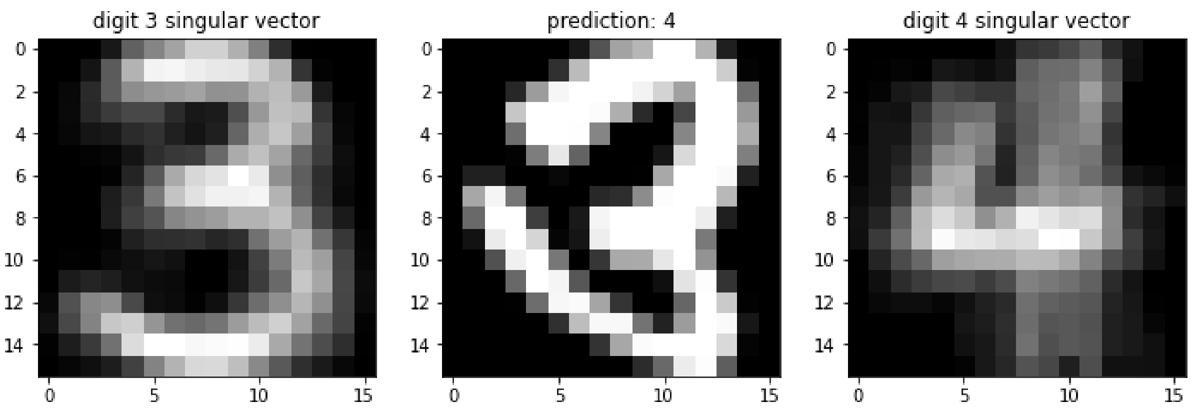
Then y will be classified as C1 if $||y - y_1^\perp||_2 < ||y - y_2^\perp||_2$, otherwise it will be classified as C2.

Results

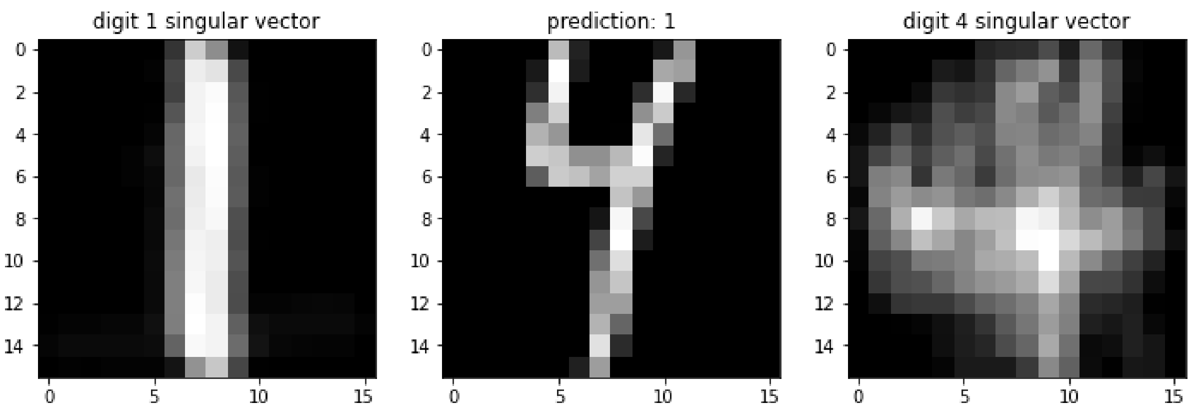
• Visualize a bunch of datapoints of X with the function plt.imshow

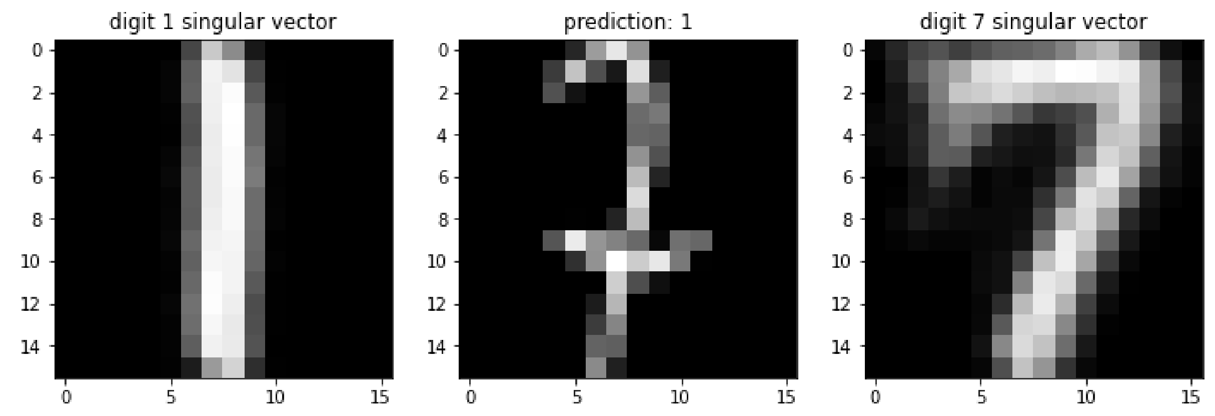


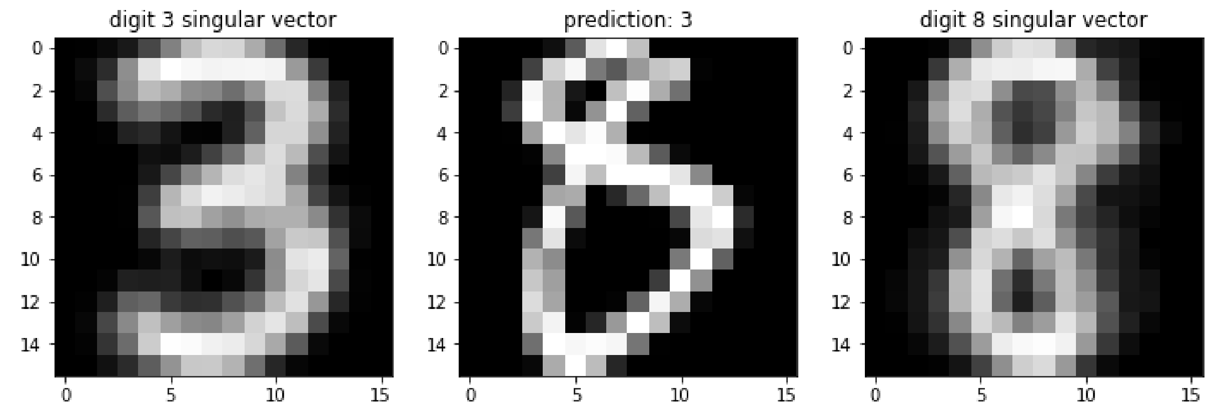
For digits 3 and 4 the algorithm is very accurate reaching almost 100% in most iterations. The only exceptions are digits badly written.



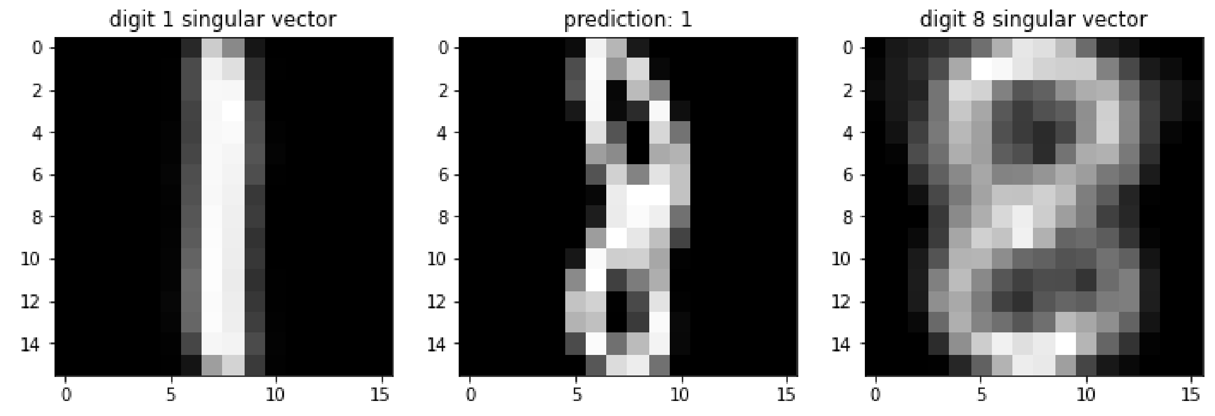
Whilst visually similar couples of digits, e.g. (1, 4) , (1, 7) , (3, 8) , could lead to higher mismatch percentage.



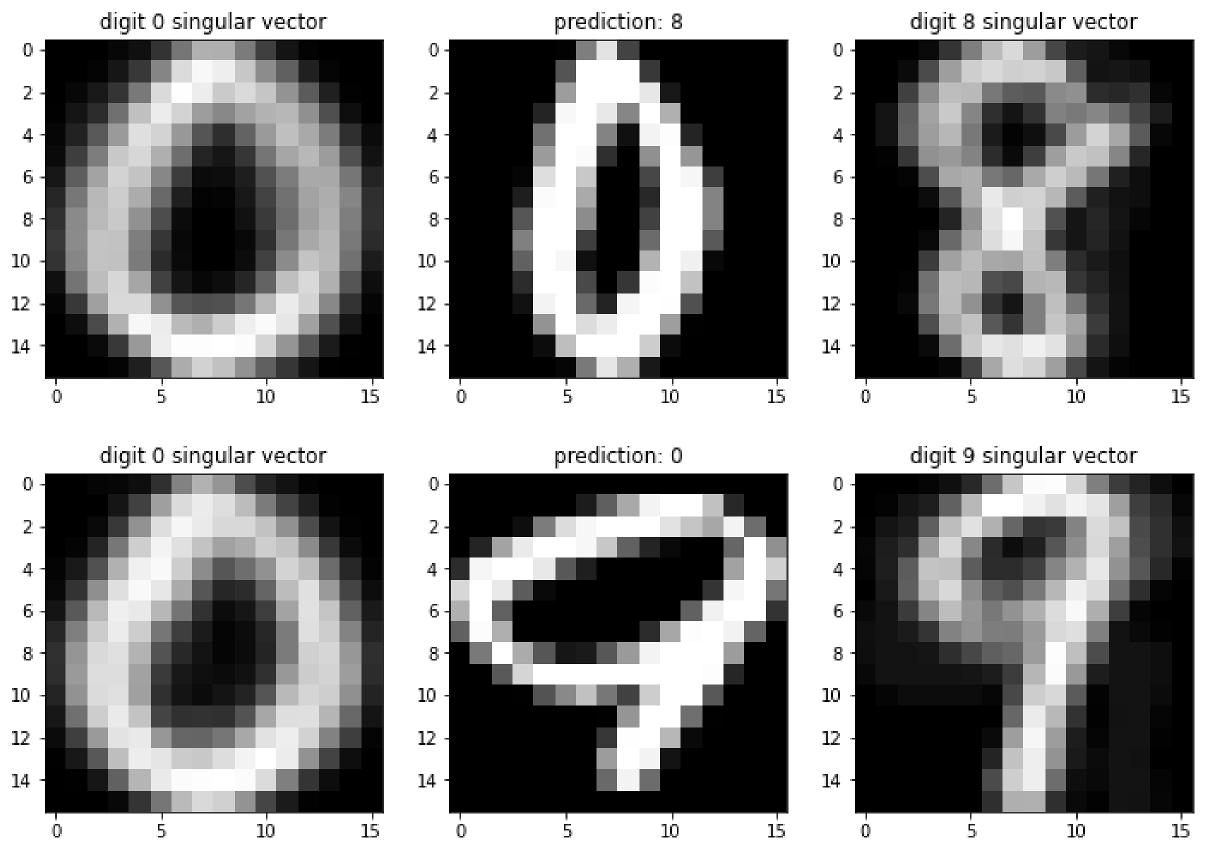




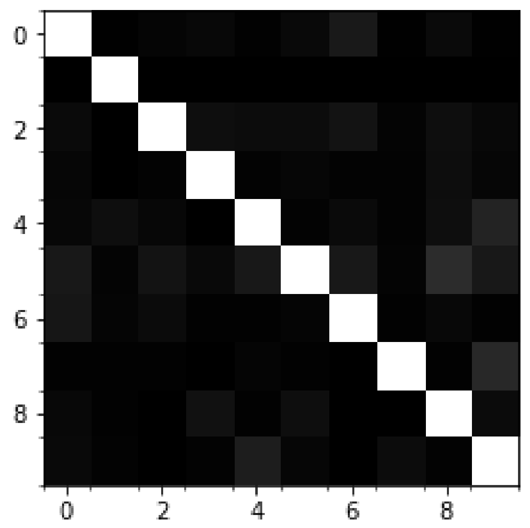
An exception is the couple (1, 8) , these digits are not visually similar but could still lead to mismatch because of the presence of many stretched 8s that almost degenerate in a straight line resembling the digit 1.



The digit 0 is very badly classified, probably given its round shape that could resemble most of the digits, i.e. 3,5,6,8,9.

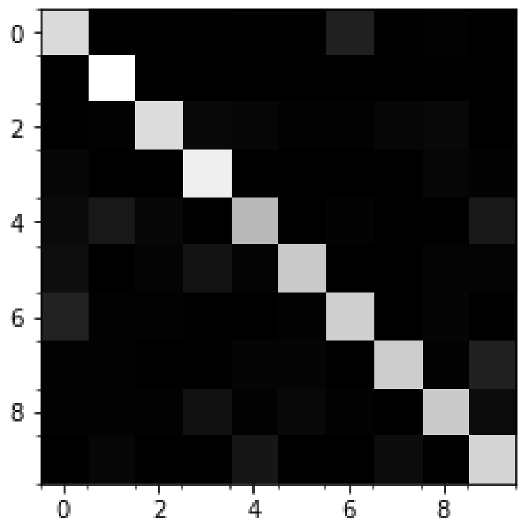


The main problem is that the dataset is unbalanced: the digits are not represented with the same number of samples. This influences the performance of the algorithm. Once the dataset is balanced, the results improve quite a lot. The heatmap comparing every couple of digits is shown below:



The heatmap changes at every iteration meaning that the main problem is given by the unbalanced dataset, not the visual similarity between digits.

Using more than 2 classes the procedure is the same but the results are generally worse than the 2 classes case as shown in the heatmap below:


0
2
4
6
8
0
2
4
6
8