Scientific Computing III Final project

Pattern recognition using singular value decomposition

June 10, 2019

Juuso Kauppala

UNIVERSITY OF HELSINKI
DEPARTMENT OF PHYSICS

PB 64 (Gustaf Hällströmin katu 2)
00014 Helsinki university

Contents

Abstract

The final project task #2 of the course Scientific Computing III was to implement a pattern recognizing program using singular value decomposition (SVD) of matrices. The used patterns for training and testing were 28x28 pixel images of handwritten digits from NIST database.

These patterns can be recognized using the **U**-matrix of the SVD, of which columns form a orthogonal basis for the column space of original matrix with vectorized images of digits as column vectors. The algorithm is to compute SVD for matrices formed by different digits' data sets and then finding a minimum for a certain residual.

Only some of the columns of the matrix **U** is used as an approximation and the cutoff value k is problem dependent and therefore a suitable value for this problem was investigated. Visualizations of some of the written digits and their so called "eigen-patterns" are also given and the recognition algorithm accuracy was tested using a bit larger data set consisting of images of digits. The program recognized approximately 97 % of the test samples correctly with given k-value.

1 Introduction

The final project task #2 of the course Scientific Computing III was to implement a pattern recognition program using singular value decomposition (SVD). The patterns used in this project were handwritten digits (0-9). This kind of pattern recognition can be a challenging task, since there can be quite a lot of variety in the written digits and also some of the digits have fairly similar patterns. Also the task becomes computationally more demanding with increasing image resolution.

Pattern recognition algorithms for automatic conversion of handwritten text or digits have applications for e.g. automating post sorting by address or postal code, licence plate readers and digitizing printed documents.

2 Methods

The used method for pattern recognition in this project was performing singular value decomposition to matrices consisting of vectorized images of digits as columns and then using the resulting U-matrix to recognize the patterns. If $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n)$ is the matrix of n images as column vectors of length m, then the SVD would be

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}, \tag{1}$$

where $\mathbf{U}$ is an $m \times m$ matrix and its columns form an orthogonal basis for the column space of $\mathbf{A}$. Using this matrix, a given vectorized image $\mathbf{z}$ of a digit can be recognized by minimizing the residual

$$r = \left|\left|\left(\mathbf{1} - \mathbf{U}_k\mathbf{U}_k^{\mathrm{T}}\right)\mathbf{z}\right|\right|_2, \tag{2}$$

since the matrix $\mathbf{U}$, which gives the smallest residual is most likely from the set of digits same as $\mathbf{z}$. A more thorough explanation is given in the course's project assignment document.

In this project the SVD:s are computed for all digits from the NIST datasets and the residuals from every U-matrix for two self made test images are compared to test performance of the program with different values for k. Lastly the method is also tested for a larger test data set for each digit.

3 Implementation

The program is implemented mostly with functions and methods from standard python libraries. The program uses images of handwritten digits (28x28 pixels) from the NIST database in training and testing the pattern recognition algorithm. The program is implemented to leave part of the data sets as testing data (200 last digits from each digit set). Descriptions of all of the functions used and the main program are given below. Example images of the handwritten digits in grayscale is shown in figure 1.

Figure 1: Examples of handwritten digits from the NIST data set. The first 5 images of the test data set.

3.1 Functions

In this section, detailed descriptions of the implemented functions and their parameters are given.

Functions:

- `get_path`

  This function adds the path to the input data directory to the given filename and needs only the filename as input parameter. Returns the filename with the path appended to it.

- `get_svd_of_digit`

  This function takes two integers as input parameters, a single digit and a the number of images to be left for testing data, and returns the SVD (U,Σ,V - matrices) for the given digit using the NIST data sets. For the computation

of the singular value decomposition a method `svd` from `scipy.linalg` python library is used. The method `svd` is implemented to use LAPACK library's `gesvd` or `gesdd` -drivers and the default is the `gesdd`, which uses the so called divide-and-conquer algorithm.

- `compute_residuals`

  A function for computing the residuals for a given digit using every digit's SVD (U-matrix). Takes a tuple of all U-matrices and an image of a digit as input parameters and returns a numpy vector consisting of the residuals.

- `recognize_digit`

  This function takes a tuple of all U-matrices (for each digit) and an image of a digit as input parameter and returns an integer number (digit) which has the residual minimizing U-matrix.

- `test_algorithm`

  A function for testing the algorithm's accuracy. The function takes a tuple of all the U-matrices, integer number of the test data set size and any number of digits (integers) for which it tests the algorithm. The function prints which digit it is testing and also how many of the test samples were recognized correctly and what was the test set size. More details of the implementation are given in the next section.

- `show_digits`

  This function takes an integer `n`, a digit data set and a filename as input parameters. The function saves `n` first grayscale images of the digits from the data set into a file named as the input filename for visualization of the data.

- `show_eigen_patterns`

  Similar function as the `show_digits`, but instead saves coloured images of the eigen patterns from the given U-matrix.

- `load_test_data`

  This function takes two integers as input parameters, a digit and the test data

set size. The function then reads the digits NIST data from the input data sets and returns an array consisting of images of digits as column vectors.

3.2 The main program

The main program computes the SVD:s for each digit using the given NIST data sets and leaves 200 last images from each set as testing data sets. Then the program reads two self made images for test digits and computes the residuals for both digits and using U-matrices of each digit. Three different values for k is used and the results are plotted as residual vs digit figures.

From testing with the 2 self made test images, the clearest minimum for the residual of the correct digit was obtained when $k = 10$ and therefore this value was used for testing the algorithms accuracy. The main program then uses the `test_algorithm` function to test the accuracy for each digit (0-9). This function loads the test data using `load_test_data` function and then uses the `recognize_digit` function for each digit in the test set. The `recognize_digit` function then again uses the `compute_residuals` function for calculating the residuals of the given digit and then the `recognize_digit` function returns a single integer number (digit) which has the smallest residual. The `test_algorithm` function then calculates how many digits were recognized correctly from the test samples and what was the original test data set size.

Lastly the program saves images of the digits from the test data set and some eigen patterns from the U-matrices of two chosen digits.

3.3 Running the program

The program can be simply run by command `python3 main.py` in the *src* directory. Outputted plots and figures are saved in the *run* directory but the figures might look different than in this document depending on the version of matplotlib. The results presented in this document are done using matplotlib version 3.0.2. Also the program searches the input images of digits from NIST database in directory *run/nist_digits* and will result in an error if the input files can not be found.

4 Results

The training data sets consisted of several thousands of images of handwritten digits of resolution 28x28. The first 5 "eigen-patterns" from the U-matrix of the SVD of the chosen two digits, which were 2 and 9, is shown in figure 2. The digits are easily recognizable by visual inspection from these eigen-patterns.

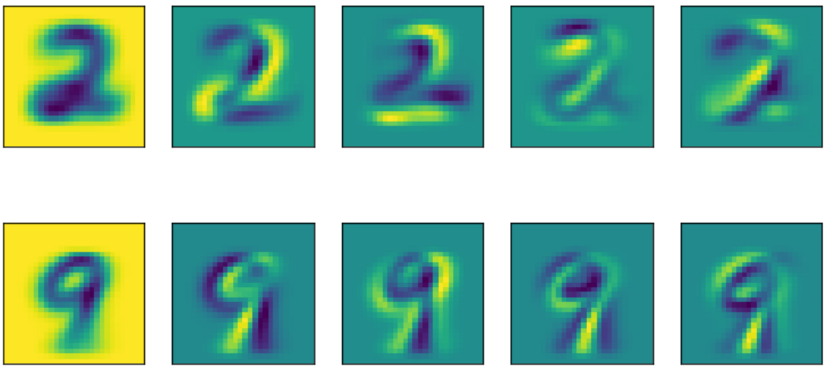

Figure 2: First 5 eigen-patterns of digits 2 and 9 from the computed U-matrices.

The residuals of the two self made images of digits were compared using three different k-values: $k = 10, 20, 30$ and the results are plotted below. The images were drawn using GIMP and prepared by simply cutting and making them into right size and they can be found in the /run directory.

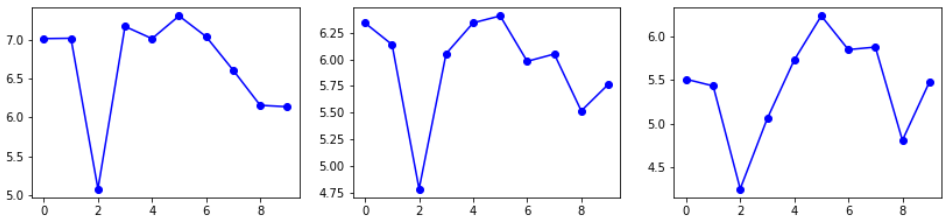

Figure 3: The residuals of self made digit 2 using every digit's U-matrix. The digit was recognized correctly in each case and the clearest minimum was obtained when $k = 10$.

From the results it can be seen that the algortihm recognizes both of the test digits correctly with each tested value for k. The clearest minimun however was obtained

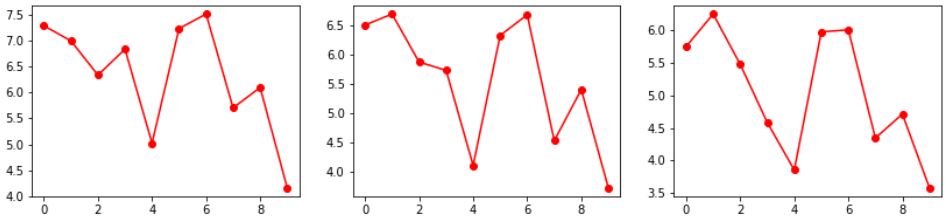

Figure 4: The residuals of self made digit 9. Again the digit was recognized correctly and the residual had clearest minimum when $k = 10$.

when $k = 10$ which is why this cutoff value was also used for testing the algorithms accuracy.

Next the program tests the algorithms accuracy using data sets with 200 images of the NIST data sets. The data sets contain several thousand images of digits so the amount of images used for testing is reasonable. Below is the output of the program.

```
Computing SVD:s of all digits...
Computing the residuals...
Recognizing images of digit: 0
Done. Digits recognized correctly: 196/200

Recognizing images of digit: 1
Done. Digits recognized correctly: 200/200

Recognizing images of digit: 2
Done. Digits recognized correctly: 196/200

Recognizing images of digit: 3
Done. Digits recognized correctly: 191/200

Recognizing images of digit: 4
Done. Digits recognized correctly: 191/200

Recognizing images of digit: 5
```

```
Done. Digits recognized correctly: 189/200

Recognizing images of digit: 6
Done. Digits recognized correctly: 199/200

Recognizing images of digit: 7
Done. Digits recognized correctly: 197/200

Recognizing images of digit: 8
Done. Digits recognized correctly: 189/200

Recognizing images of digit: 9
Done. Digits recognized correctly: 189/200
```

From the program's output it can be seen that the program recognizes on average 96,9 % of the digits with the given test set sizes.

5 Conclusions

The results obtained are quite good since the program seems to recognize close to 100% of the test data with $k = 10$. Part of the reason for this accuracy is certainly the prehandled data, which made sure that both the training and testing data sets had correctly scaled and centered images of the digits. For making the results even more accurate, more extensive testing with different cutoff values should have been done.

The implementation could be improved by including errorhandling and perhaps also generalizing the used functions so that they could be more easily used in other applications as well. Also the implementation of the digit recognizing function needs all of the U-matrices which can take a huge amount of memory if the used images were larger. Still for this project the implementation is reasonable since the image resolution was given and this implementation has to only compute each U-matrix once.

Even though the used algorithm gives good results with data sets of this size, the computed matrices would grow extremely large if very much larger data sets would be

used. This sets limitations for the applicability of this algorithm in the digit recognition and the implementation could either use some other algorithm or would have to be changed to be less memory demanding in some situations.