University of Helsinki

Scientific Computing III

final project

Pattern recognition using singular value decomposition

Author:
Joonas Kylliäinen

May 21, 2015

Contents

1 Introduction

Pattern recognition is a field in data mining and machine learning. When pattern recognition system is trained from existing data, it is called supervised learning. Singular value decomposition or SVD is a technique of this kind [1].

Singular value decomposition's roots are in the late 1800's and early 1900's. Five mathematicians were responsible for establishing the existence of the singular value decomposition and developing its theory: Eugenio Beltrami (1835-1899), Camille Jordan (1838-1921), James Joseph Sylvester (1814-1897), Ernhard Schmidt (1876-1959), and Hermann Weyl (1885-1955) [5].

Efficient use of SVD's potential became possible in 1965 when Golub-Kahan algorithm was developped by Gene Golub and William Kahan [3] but it was further developped in 1970 to Golub-Reinsch algorithm by Gene Golub and Christian Reinsch [4]. It is still often used today.

In this task a series of images of digits were given as a training set for learning purpose and a program was to be written which could recognise any cipher by using singular value decomposition.

2 Methods

In pattern recognition, singular value decomposition is used in order to train the program. A vast amount of pictures of digits is given to the program. Every picture of number is turned into a column vector and all these picture vectors of a single digit are molded into a matrix i.e. every column of this matrix is a picture of a number. To this matrix let's say matrix A a SVD is performed

$$A = USV^{\intercal} \tag{1}$$

Where U and V are unitary matrices, S is a diagonal matrix which values are the singular values. Now the columns of the matrix U are the "eigenpatterns" of the image and to these we shall compare the images we want to recognise.

Let's say we want to recognise picture z. We can approximate the matrix U by choosing only k most dominant singular values. Now we need to simply compare b to every number's U-matrix and find the smallest residual [2]

$$b - \sum_{i=1}^{k} x_j u_j \tag{2}$$

which is equivalent to solving equation

$$r = ||(\mathbb{1} - U_k U_k^\intercal)z||_2 \tag{3}$$

3 Implementation of the methods

3.0.1 Implementation

Method was implemented in octave programming language. The training images and the test images were aquired from MNIST database [9]. The training set image matrices contain 60 000 example pictures. They were required to be converted to .mat-files using a octave script available in here www.cs.toronto.edu/ hinton/code/converter.m.

A SVD was performed to all of these matrices using octave's built-in function **svd()** and produced U matrices were storred in a three dimensional array. Couple "eigenpatterns" were plotted by first reshaping columns to matrices with **reshape()**-function and then using **imagesc()**-function.

Then two digits were recognised by calculating equation (3) with k-values ranging from 1 to 100 with every numbers U-matrices.. The smallest residual of each number and the k-value that produced it were storred in an array. When reducing 1 from the index of that arrays smallest value, correct number was produced.

By storring every numbers every k-value's residuals was a k-value analyze easy to make.

3.0.2 Instructions

Program **analyze.m** needs files **digit1.mat - digit9.mat**, **test1.mat - test9.mat** in the same directory as itself to be able to work. Program is executed by typing "octave analyze.m" in your computers terminal. It will take a minute to calculate all the svds but after that it will start to produce pictures. When a picture has popped, do **not** press **X**. Waitforbuttonpress()-function is being used so press any key on your keyboard or click the picture to close it. After the program has created last plot of the "eigenpatterns", it will take a few extra seconds for it to close it. Don't panic. If it doesn't close but the program already asks you a question on the terminal, you can gently try to answer it. You are asked to pick a number which is then analysed. The program will then plot a figure showing how the residuals have changed when k has changed. Finally the program tells you on the terminal, which number it thinks the figure represent.

4 Results

Figure 3 is an example of the pictures being used. The first eigenpatterns of numbers 1 and 8 are presented in Figures 1 and 2. The effect of the size of k to the residuals can be seen in Figure 4. When k increses the residuals decrease, but what most significantly, the difference between the residuals of different numbers seem to get bigger. Which logically means, bigger training set yields more accurate results. With a training set size of 60 000 pictures, the program was able to recognize numbers 1,2,4,5,6,7,9 but not 3 and 8. Every digit was tried with one picture, so it can not be stated that program work always on numbers 1,2,4,5,6,7,9 and never on 3 and 8. However result 8 out of 10 suggest that the succeeding percentage is somewhere around 80 %. In the succesful recognitions the difference between right answer and wrong answers was significant.In Figure 5 can be seen that in the failed recognitions, the difference was extremely small which means failures can be recognised.

The clear disadvantage of the SVD, is its time complexity which is for Golub-Reinsch somewhere around $\mathcal{O}(mn^2)$ [6][7][8]. Golub-Reinsch is implented at least in EISPACK and LINPACK. Some algorithms are however more efficient: even $\mathcal{O}(mn)$ is possible [8]. In this case SVD need to be calculated only once. Because real-time SVD is not needed, its time complexity is subsidiary.

Though my program calculates residuals with multiple values of k, it is not needed. One calculation of equation (3) per number is enough so real-time time complexity of this method is linear.

5 Conclusions

Singular value decomposition is a technique which can be used in systems which doesn't require real-time calculation of SVD. The learning i.e. the eigenpatterns it creates can be used succesfully in moderate time and in adequate success precision in pattern recocnition. I.g. postal office can use this technique since the failures can be spotted and sorted by hand. For improving this technique obvious solutions are to increase the amount of training images and using large values of k. Also using right SVD algorithms in right situations and in right environments, can the efficience of the calculation of SVD be maximised.

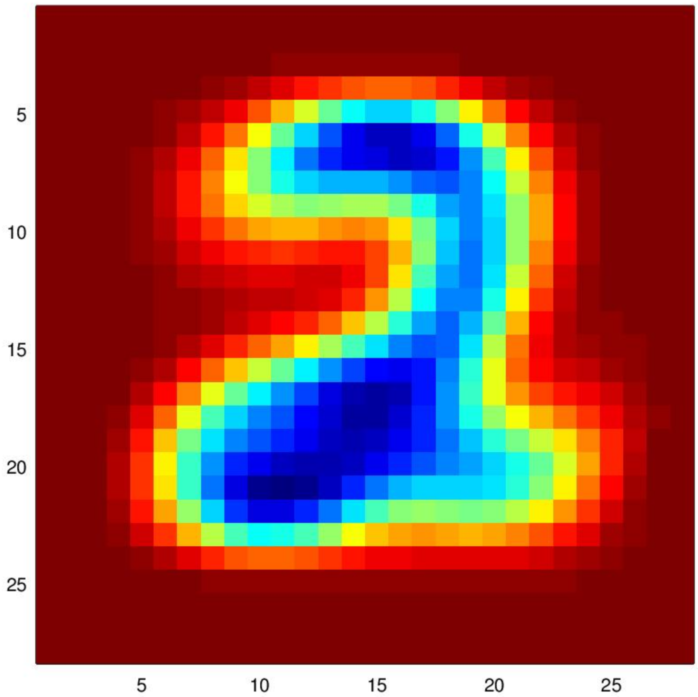

Figure 1: One of the eigenpatterns of number 2

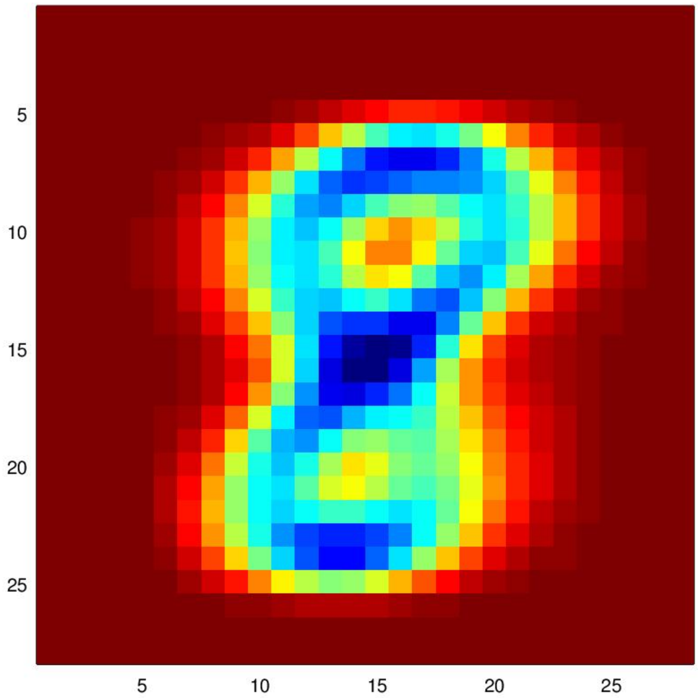

Figure 2: One of the eigenpatterns of number 8

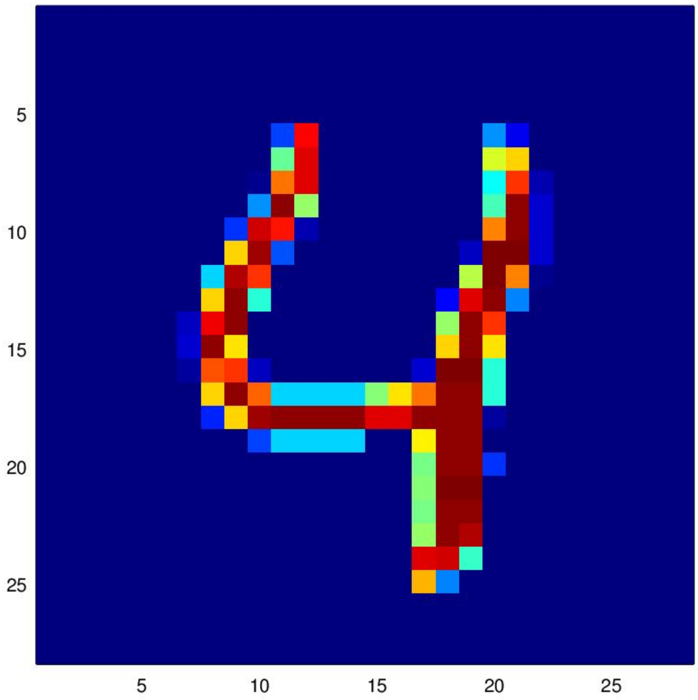

Figure 3: Test image of number 4

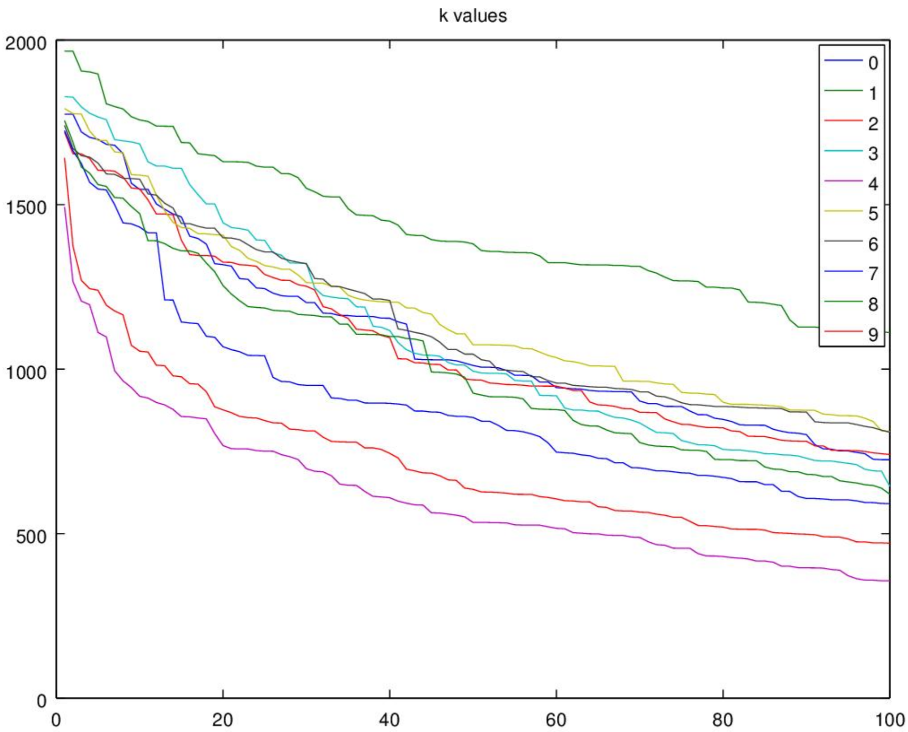

Figure 4: K values, succesfull recognition with number 4

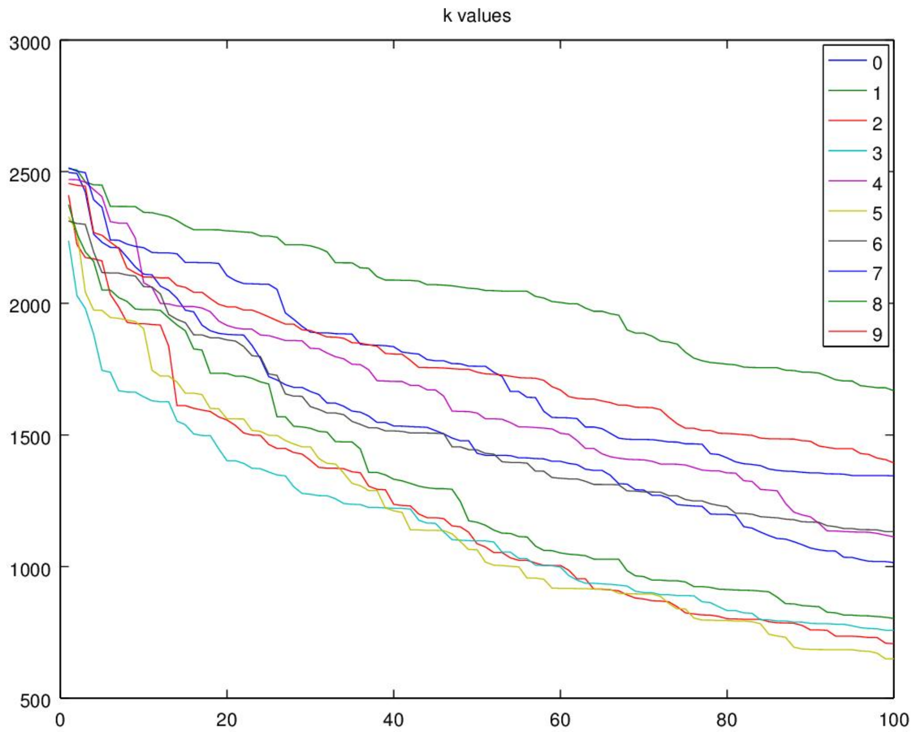

Figure 5: K values with a failed recognition with number 3

References

[1] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 1 edition, 2006.

[2] Lars Eldn. Numerical linear algebra in data mining. *Acta Numerica*, 15:327–384, 5 2006.

[3] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, 2(2):205–224, 1965.

[4] G.H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[5] G.W.Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35, 1993.

[6] Michael P. Holmes. Fast svd for large-scale matrices. `http://sysrun.haifa.il.ibm.com/hrl/bigml/files/Holmes.pdf`. [Online; accessed 21.5.2015].

[7] Anthony A. Maciejewski and Charles A. Klein. The singular value decomposition: Computation and applications to robotics. *The International Journal of Robotics Research*, 8(6), 12.

[8] Franklin T. Luk Richard P. Brent and Charles Van Loan. Computation of the singular value decomposition using mesh-connected processors. *Journal of VLSI and Computer Systems*, 1(3), 3.

[9] Corinna Cortes Yann LeCun and Critopher J.C. Burges. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`. [Online; accessed 21.5.2015].