Pattern Recognition Using Singular Value Decomposition

Scientific Computing III

Noora Mäkelä

June 10, 2019

Contents

1 Abstract **2**

2 Introduction **2**

3 Methods **2**

4 Implementation **3**

5 Results **4**

6 Conclusions **9**

1 Abstract

The project studied using Singular Value Decompostion in Pattern Recognition (SVD), patterns classified where images of digits. The method was successful when it was applied only to two digits having different shapes. Then the classification results were approximately 90 % correct. The method started having difficulties in classification when the digits had similar shapes or there were more SVD matrices used in the classification than two.

2 Introduction

The problem we want to study is pattern recognition using Singular Value Decomposition. In this project we have classified pictures of digits using the SVD. Pattern recognition is an important topic for many fields and also complicated subject. There are patterns all around us and humans are good and used to classifying them, but it is hard for machines to do the same. For example if a human is driving a car, she or he notices other cars and other objects, but for a machine that is not that easy. Machines do not have thoughs and mind, so they function in a very different way than people. The machine must know what patterns to look for and how to classify them, to use the information in some way. So pattern recognition is very important topic, because it would be highly useful in many places.

3 Methods

The method used in this project was Singular Value Decomposition (SVD). In singular value decomposition we perform factoriazation to a real or complex matrix A into a three matrices product $A = USV^T$. Here A is a set of images, each image interpreted as a column vector. We can use SVD in pattern recognition by using matrix U obtained by SVD. Using matrix U we can express an arbitrary image z in the U basis.

$$z \approx \sum_{i=1}^{k} \alpha_i u_i = U_k \alpha$$

With U matrix we can calculate the residual for each pattern.

$$r = ||(1 - U_k U_k^T)z||_2$$

We can determine the classification result by comparing the residuals, the smallest residual is most probably the pattern in the picture we are trying to classify.

4 Implementation

In this project I have implemented a program for pattern recognition using singular value decomposition (SVD). The program can be ran with $python3 src/final\_project2.py$. In the program I have used numpy.linalg's implementation of singular value decompostion (numpy.linalg.svd). I have used SVD for the MNIST data to train the program to make good predictions. I implemented the calculation of the U matrices in two ways. The first solution was just to solve the U's with SVD for just two patterns, this is implemented in
$singular\_value\_decomposition\_for\_two\_numbers$ func- tion. So then the unknown picture could be classified to be one of these two patterns. The other solution was to calculate the U matrices for every pattern, in this case the patterns are 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. This is implemented in $singular\_value\_decomposition\_for\_all\_numbers$ function. The we could classify the unknown picture using all of these matrices. Then having obtained the U matrices we can move onto calculating the residual. Here we use the U_k matrix. The residual is calculated using numpy.linalg's ready norm function (numpy.linalg.norm) and numpy's matmul function for calculating the matrix multiplications. The cut-off parameter I have found by running the program for digits 4 and 6, and running classification for these digits using multiple values of k and checking for which value of k the classification is most succesful. The optimal value of k I found is 7. This was done in $determine\_cut\_off\_k$ function. It is now commented out in the main program because it is heavy and time consuming to run. But it can be uncommented if it is wanted to be ran.

With the residuals we can now classify unknown pattern (image of number). I have done this in $classify$ function. In the function we go through all of the U matrices and calculate the residual for the unknown pattern. Then we use numpy's argmin function to find the place of the minimum residual value and using it's index we can find out the pattern the unknown image probably is.

5 Results

I performed SVD for digits 4 and 6, and plotted 4 first eigenpatterns for both of the numbers.

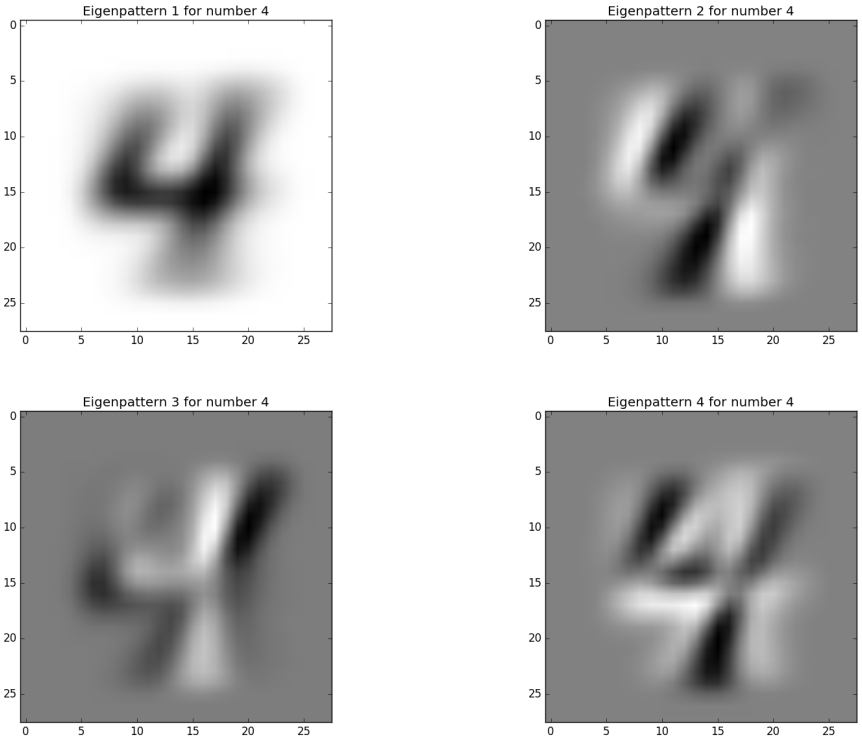


Figure 1: First four eigenpatterns for digit 4

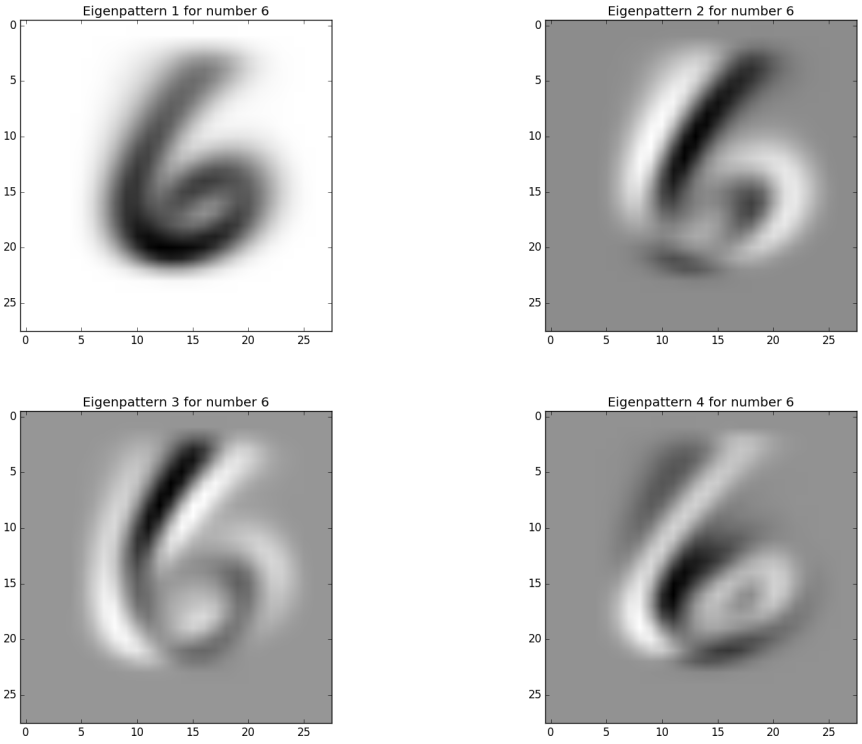


Figure 2: First four eigenpatterns for digit 6

From the eigenpatterns we can clearly see the shapes of 4 and 6. In the project we used these eigenpatterns (in U matrices) to calculate the residuals with some unknown patterns. Then the pattern having the smallest residual would be the right class for the unknown patterns. This makes sense because we want to classify the unknown pattern as the class having similar shaped eigenpattern than the unknown pattern has. So the pattern with the smallest residual has the most similar shape with the unknown pattern. So the idea is to classify the pattern with the most similar class.

Then I classified 500 images of 4 and also 500 images of 6, and checked the percentage when the classification was right i.e. the accuracy. The pictures I classified where from the MNIST test dataset. The classification was rather successful, 88.0 % of the images of fours where classified as fours and 96 % of the images of sixes where classified as sixes. The program had difficulties if the digits had similar shapes, so in those cases the classification was many

times wrong. For example when classifying images of 3 and 8, the results were not good. Only 69.0 % of images of threes were classified as threes and similarly only 57.9 % images of eights where classifies as eights. After this I found three images, one of digit 8 and two of digit 2 from the Internet and classified them using the program.

Figure 3: The images for classification

The picture of eight was classified correctly and also correctly classified was the image of two in the right upper corner in Figure 3. But the first

image of two was classified as an eight. This was the image in the left upper corner in the Figure 3. Clearly we can see that that image has similar shapes as an eight, so the program is fooled by these similarities. So to get right classification result we would have to choose two digits that have minimal similar features.

I also tried calculating all U matrices in $singular\_value\_decomposition\_for\_all\_numbers$ function, and then calculating the residual of an unknown pattern with all of these matrices. Then the right class would be the class having the smallest residual. This was not very succesful, because different digits have similar shapes and the program is easily distracted by those similarities. With these matrices I classified 500 patterns of every digit and the results were:

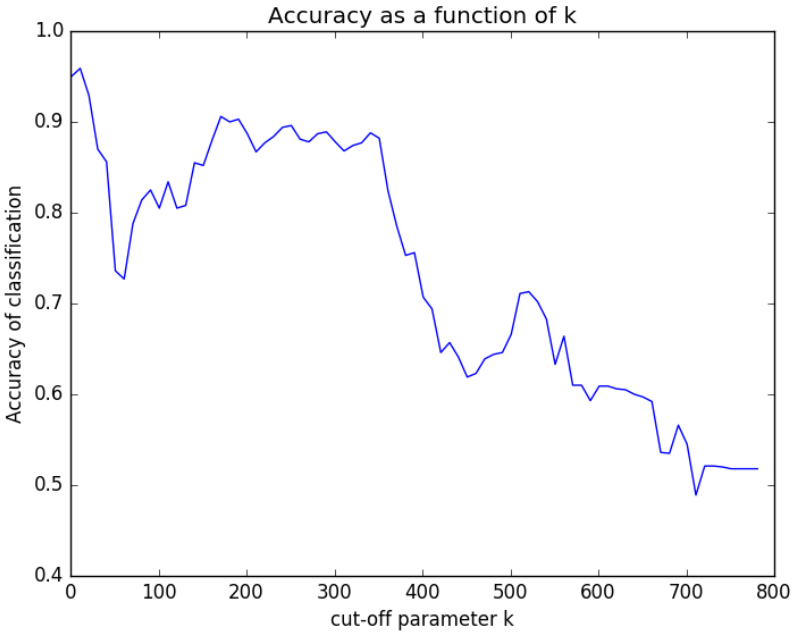
* Classified pictures of number 0, accuracy 75.8 %
* Classified pictures of number 1, accuracy 6.0 %
* Classified pictures of number 2, accuracy 80.6 %
* Classified pictures of number 3, accuracy 37.4 %
* Classified pictures of number 4, accuracy 52.0 %
* Classified pictures of number 5, accuracy 67.8 %
* Classified pictures of number 6, accuracy 51.2 %
* Classified pictures of number 7, accuracy 51.4 %
* Classified pictures of number 8, accuracy 51.6 %
* Classified pictures of number 9, accuracy 40.8 %

From the results we can see that the classification was not very successful in most of the cases. The digit that was classified right the most times is two, with accuracy of 80.6 % and the digit that was classified wrong the most times is one with just 6 % accuracy. This although makes sense because digit one has just a straight shape and many other digits have also straight shapes. Because it is the only shape of the digit, then it is hard for the program to find any other shapes that would help the identification, that would make the program to be sure that the digit is one and nothing else.

From the results we saw that using SVD in classification is most successful, if we find the SVD matrices for only two digits instead of all the digits. This is because with only two SVD matrices, it is easier to classify a pattern to be one of two digits, than to be one of 10 digits. When using SVD matrices for all digits, there is a lot of room for error and that was seen from the classification accuracy results when using 10 SVD matrics.

In the program I also determined the value of cut-off parameter k. This was done by creating SVD matrices for 4 and 6, and iterating through values of k. Classification was performed for 500 patterns using these values of k, and then searching k that had the biggest classification accuracy. When the accuracy was the biggest, then the classification was the most successful. I first looped through k values from 1 to 780 with step size of 10. This was done just to get a rough estimate at what would be a good place to perform this same search with smaller step size. This whole interval was not feasible with step size 1, because it is heavy and time consuming to run it even with step size of 10.

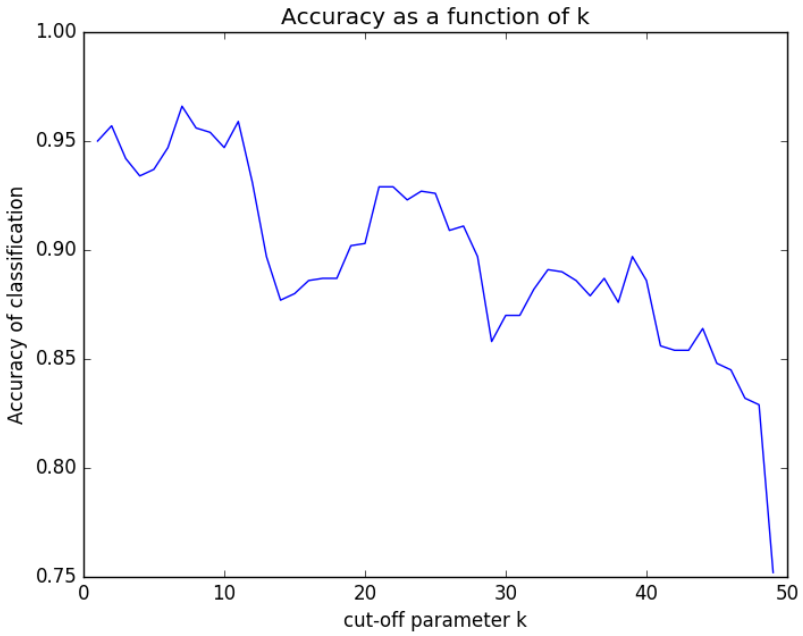
Figure 4: Finding the value of k with step size of 10



From the figure 4 we can see that the biggest accuracies are obtained

when k is smaller than 50. After this the value of accuracy decreases. After this I performed the same search, looping k values from 1 to 50 with step size of 1 and calculating the classification accuracy.

Figure 5: Finding the value of k with step size of 1



From the figure 5 we can see that the optimal value of k is between 1 and 10. My program found the exact optimal value of the cut-off parameter k to be 7. So all of the classification accuracy results presented here are obtained by using 7 as the value of the cut-off parameter k.

6 Conclusions

Using SVD in pattern recognition was useful when performing it only to a couple of digits, which do not have similar shapes. But when used digits had similar shapes then the program had difficulties in classifying these digits correctly. This was seen when I created SVD matrices for 10 digits and classified patterns using all of these. The classification results were not good,

because many digits have similar shapes as I already mentioned. I determined also the optimal value for the cut-off parameter k and that was seven.

I also would have liked to make the classification using $singular\_value\_decomposition\_for\_all\_numbers$ function work properly, so that we could classify unknown pattern to be any of the digits from 0 to 9. But maybe this is not possible when using SVD, where the classification is based on comparing the shapes together. Handling different digits with similar shapes is difficult with this approach.

Also there are places for improvement in the code quoality, but there was not enough time to perfect it.