Assignment A4

MAD 2018

Department of Computer Science
University of Copenhagen

Casper Lisager Frandsen <fsn483@alumni.ku.dk>

Version 1
Due: January 2nd 23:59

Contents

1. Exercise

(a)

Most of this has been completed in the "madA4" Jupyter Notebook. There are a few things that should be noted though:
Since the model has to pass through $(0,0)$, and our result is a function with a nonzero value for c, we have to disregard it. This makes the model less precise. The *augment* function in this exercise has been copied from the lecture slides/notebooks.

(b)

To estimate the location where he falls down, we simply have to look at the weights of w. This leads us to a function that looks as follows:

$$f(x) = -0.9788095238095242x^2 + 9.982809523809527x - 0.4659999999999926$$

We make sure the model intersects $(0,0)$ by setting $c = 0$, and solve for $f(x) = 0$ using Wolfram Alpha and get the following value for the landing:

$$x = 10.19892970080272$$

This approach has the unfortunate side effect of giving the model an inherent bias. But the alternative of having some points in the model we know for a fact are wrong seems to be worse.

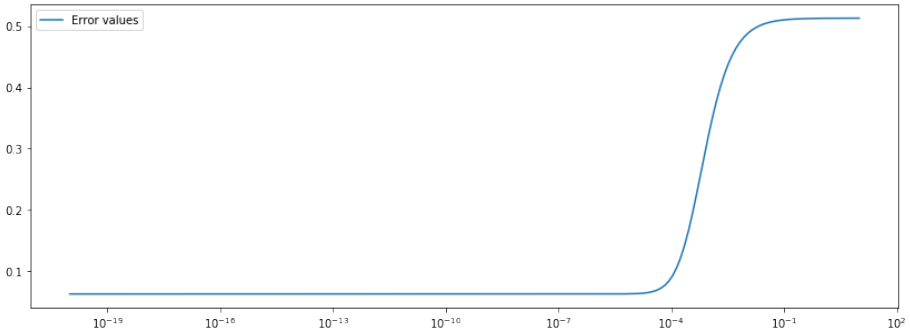
(c)

This part of the exercise has been completed in the attached Jupyter Notebook.

2. Exercise

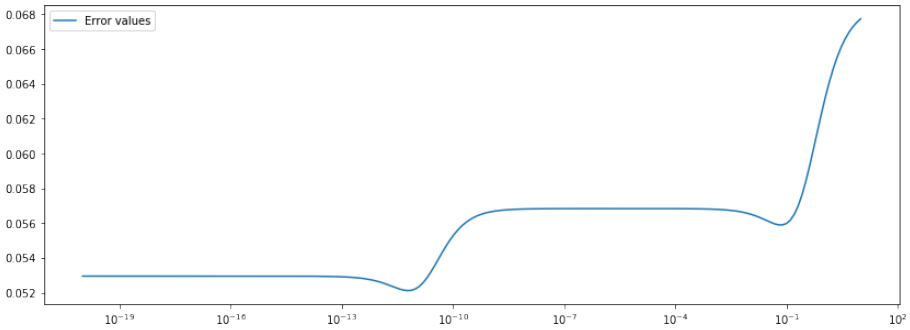
(a)

This part of the exercise has been completed in the attached notebook. The resulting plots will be displayed both here and there:



(b)

This part of the exercise has been completed in the attached notebook. The resulting plots will be displayed both here and there:



3. Exercise

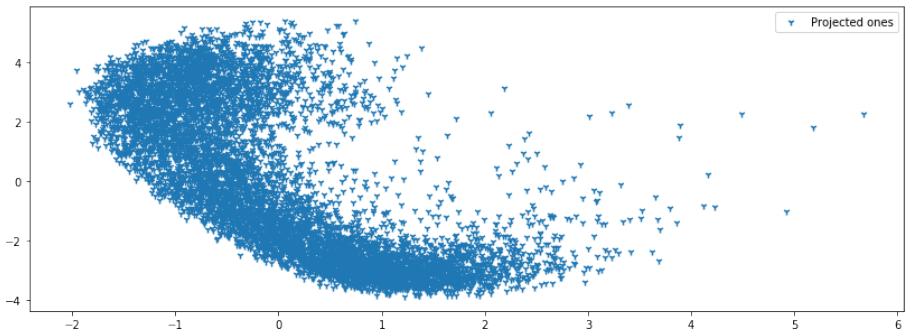
(a)

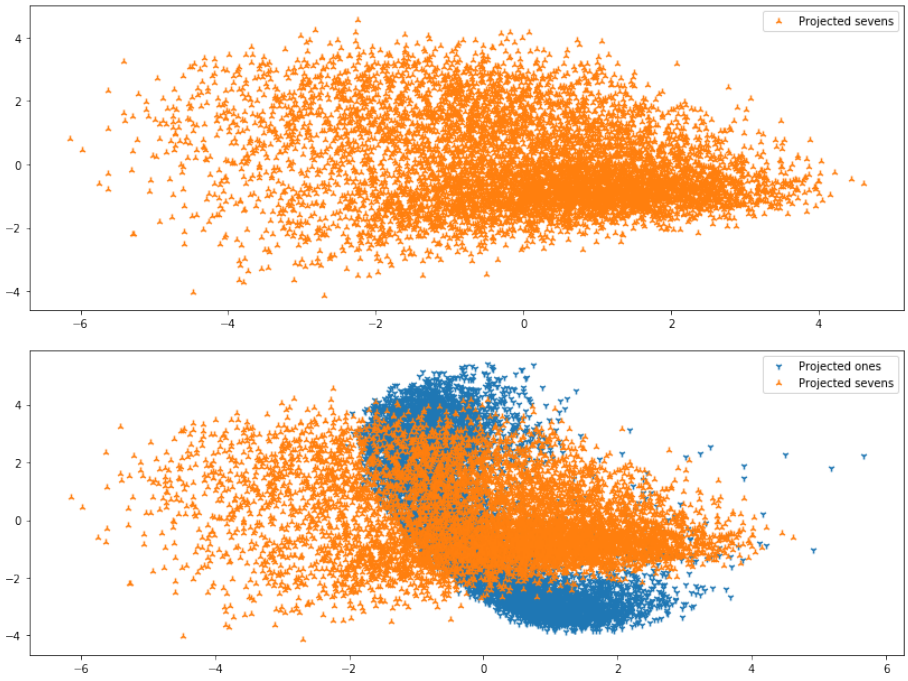
This part of the exercise has been completed in the attached notebook. The largest eigenvalue found is:

135532.65713932805

(b)

This part of the exercise has been completed in the attached notebook. The resulting plots will be displayed both here and there:





We can see from these plots that there is a significantly smaller variance in the "ones" set than in the "sevens" set. This is likely because the difference between the different sevens is larger than between the different ones. Something like the middle line through the seven, and the angle at which you draw it can vary significantly from person to person, whereas the with ones the only significant difference is whether they draw the small tip at the top.
The relevant code can be seen here:

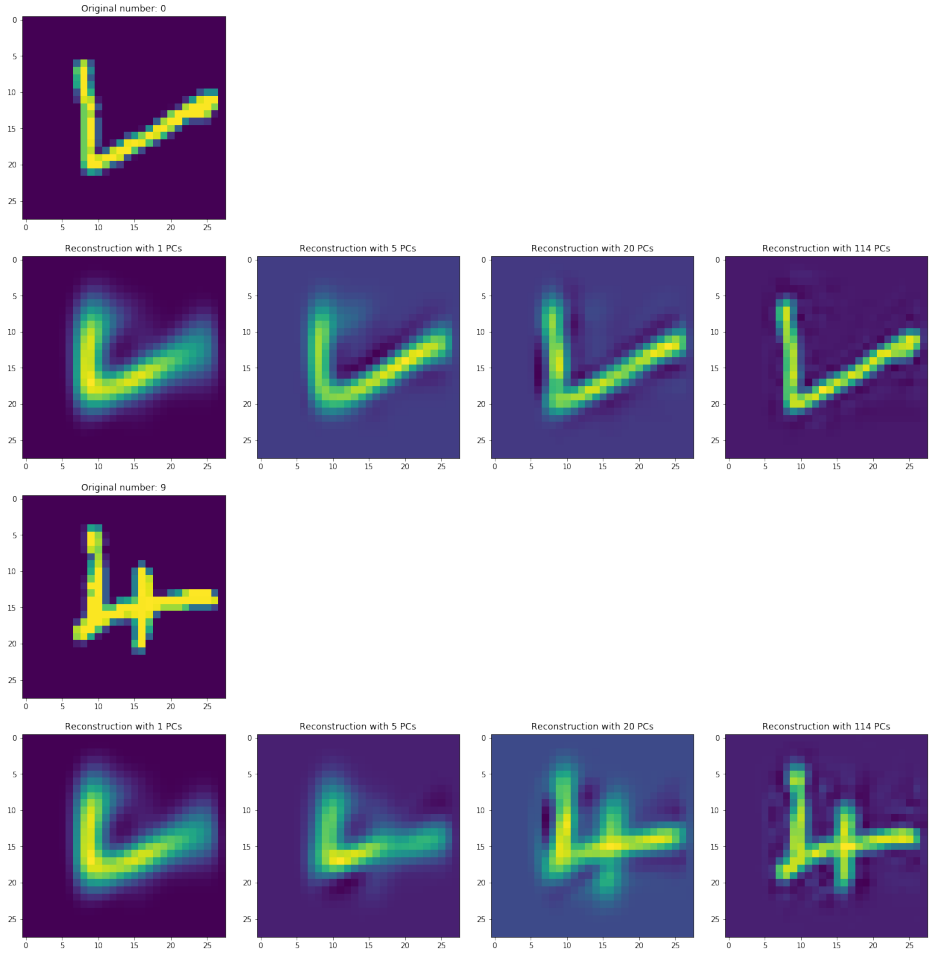
```
proj1Ones = center(ones) @ eigVecs[:,0]
proj2Ones = center(ones) @ eigVecs[:,1]
proj1Sevens = center(sevens) @ eigVecs[:,0]
proj2Sevens = center(sevens) @ eigVecs[:,1]

plt.scatter(proj1Ones, proj2Ones)
plt.scatter(proj1Sevens, proj2Sevens)
plt.show()
```

(c)

4. Exercise

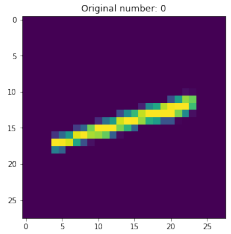
(a)

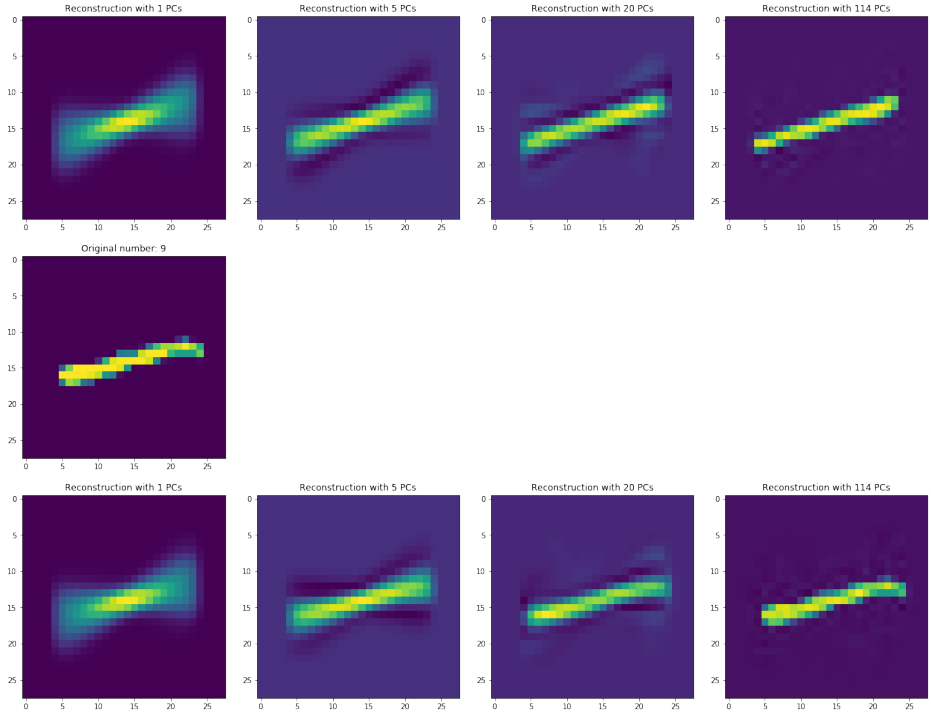
We can observe that as we increase the amount of principal components we project onto, we get more and more precise images.



(b)

There is a significant improvement in how quickly the image converges to something that looks like the number in question. We can assume that this is because there is much less variance in how the number itself is written. When writing a 1, there are much fewer variations such as a horizontal line through the middle, like in the sevens. This means that it will converge to something useful more quickly.




Reconstruction with 1 PCs
Reconstruction with 5 PCs
Reconstruction with 20 PCs
Reconstruction with 114 PCs
Original number: 9
Reconstruction with 1 PCs
Reconstruction with 5 PCs
Reconstruction with 20 PCs
Reconstruction with 114 PCs