Principal Components Analysis

-Experimental Statistics II-

Lecturer: Darren Homrighausen, PhD

Overview

Principal Components Analysis (PCA) is a very popular tool for..

- Data visualization
- Dimension reduction
- Accounting for multicollinearity
- Clustering
- Representation learning

The idea is as old as statistics itself, really,

(E.g. Pearson (1901), where PCA was first introduced)

However, the idea is constantly revisited in a variety of fields and contexts

Overview

Commonly, these learned representations capture 'low level' information like overall shape types

Other sharp features, such as lines in images, aren't captured

It is possible to quantify this intuition for PCA at least

PCA

PCA

Principal components analysis (PCA) is a computed using only the explanatory variables $X_1, \ldots, X_n$

It is the solution to various equivalent optimization problems

(Maximize variance, minimize squared error distortions, find closest subspace of a given rank,...)

At its core, we are finding linear combinations of the original (centered)

$$Z_{ij} = \alpha_j^\top X_i$$

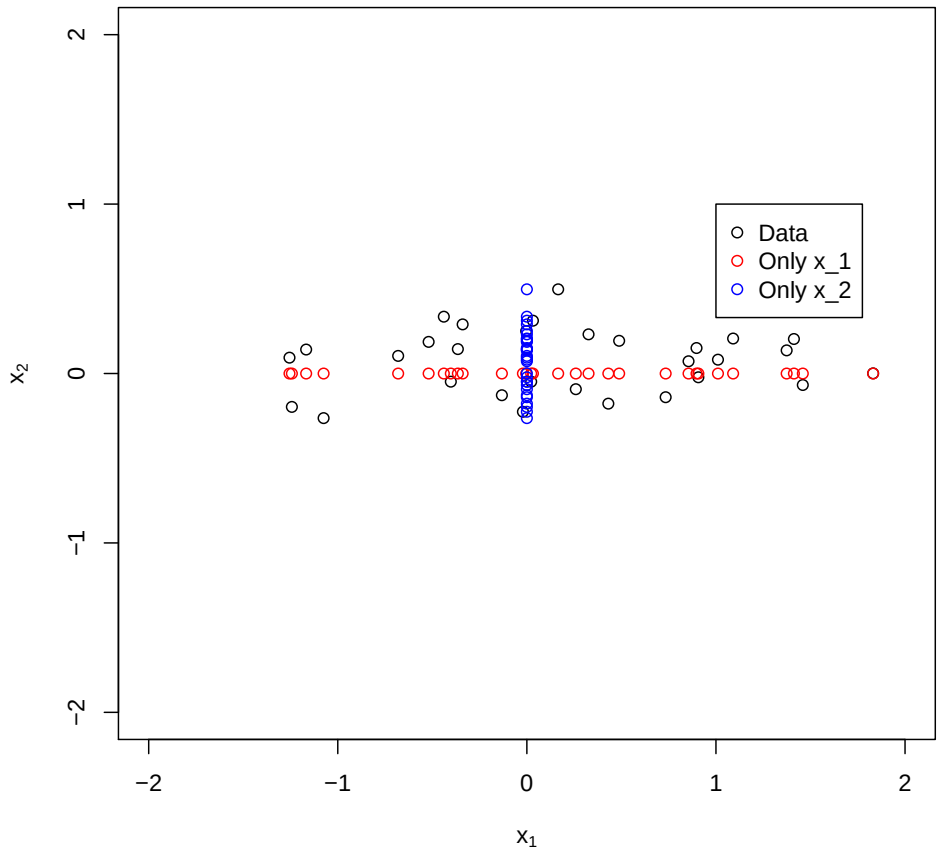
The goal (hope?) is that these new Z_{ij} are provide a new representation of the explanatory variables that more useful than the original one

PCA

Dimension reduction example

Suppose we have predictors x_1 and x_2

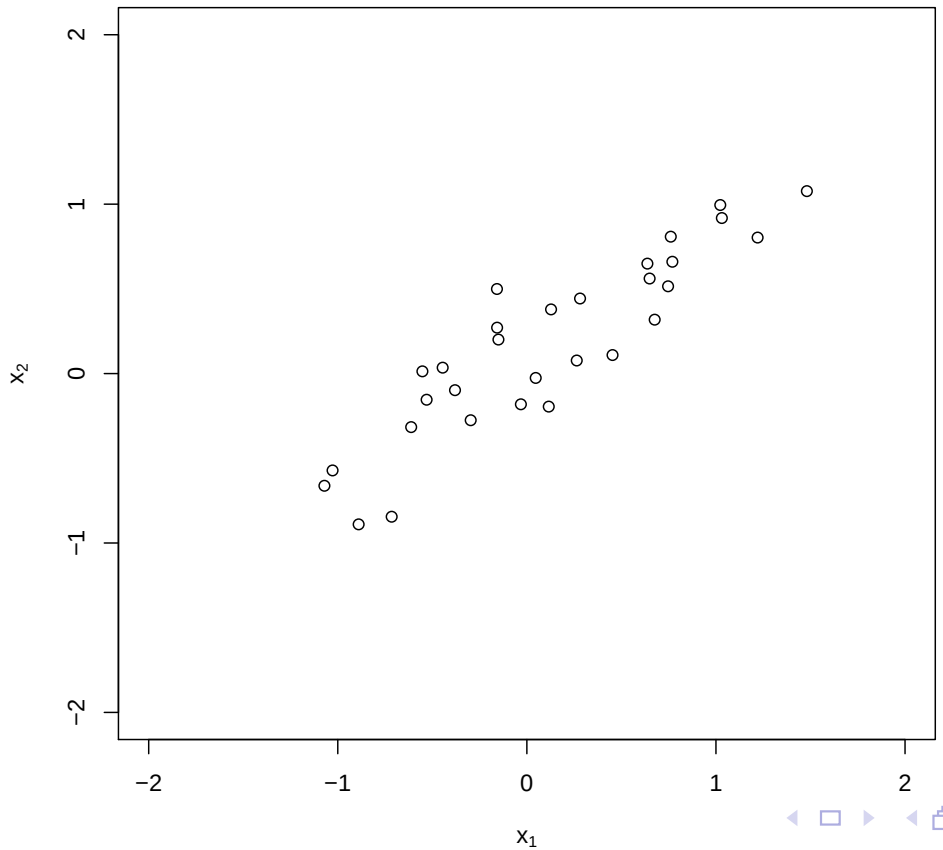
- We more faithfully preserve the structure of the data by keeping x_1 and setting x_2 to zero than the opposite



Dimension reduction example

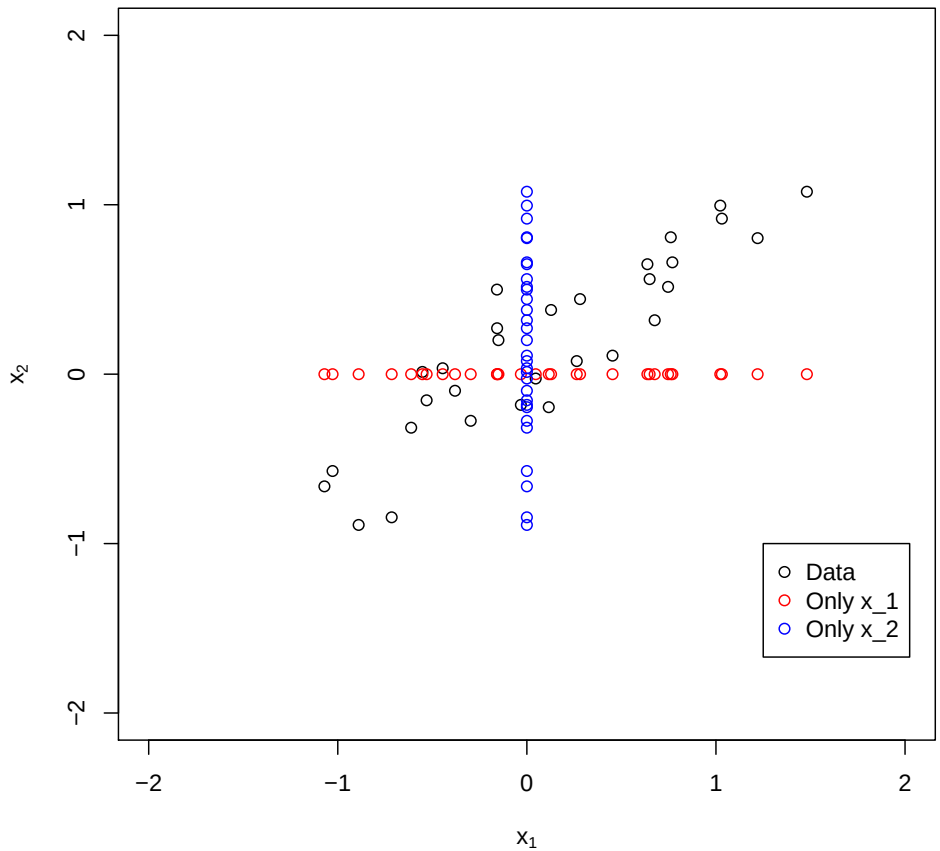
An important feature of the previous example that x_1 and x_2 aren't correlated

What if they are?



Dimension reduction example

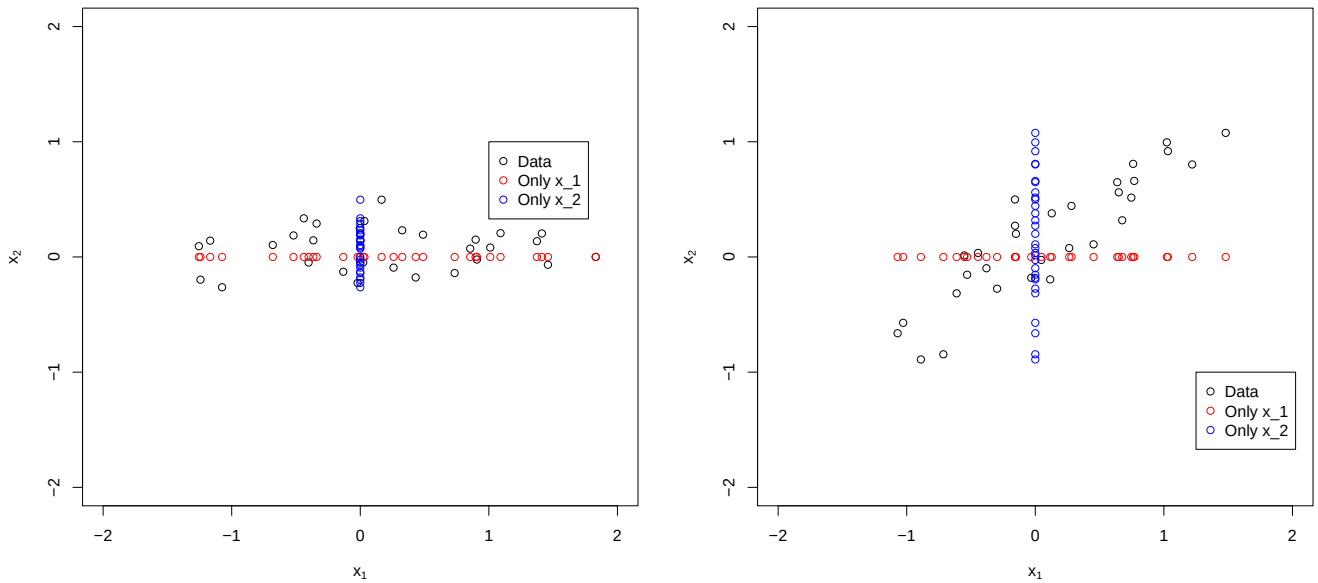
We do lose a lot of structure by setting either x_1 or x_2 to zero



Dimension reduction example

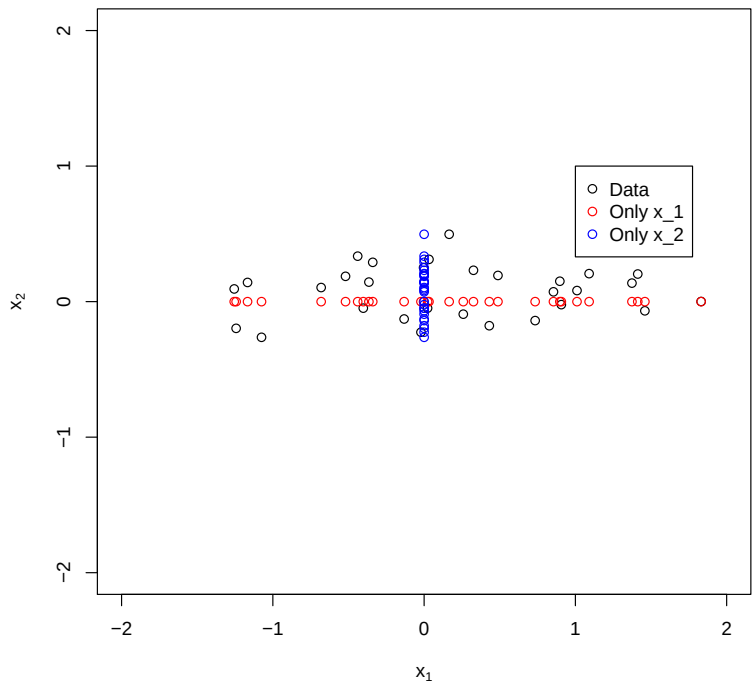
There isn't that much structurally different between the examples

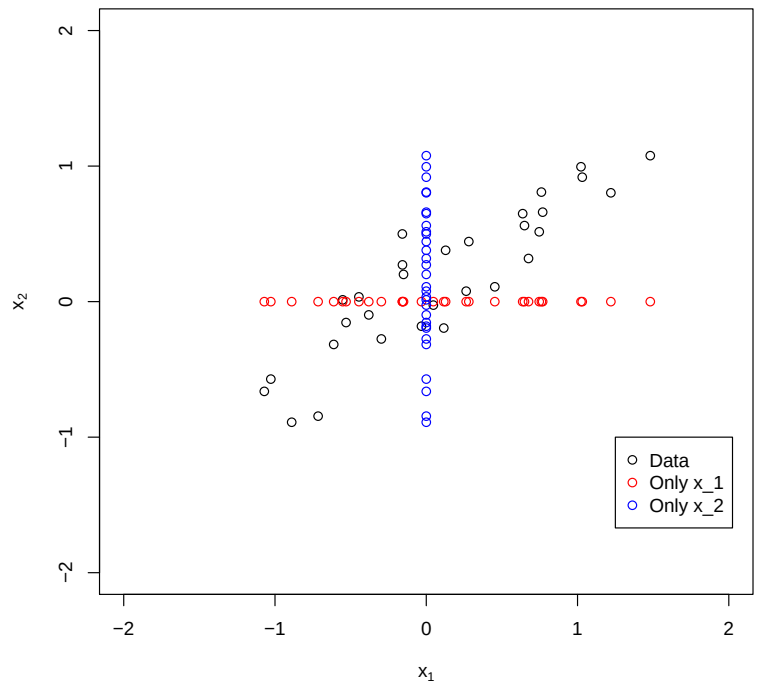
One is just a rotation of the other



Dimension reduction example

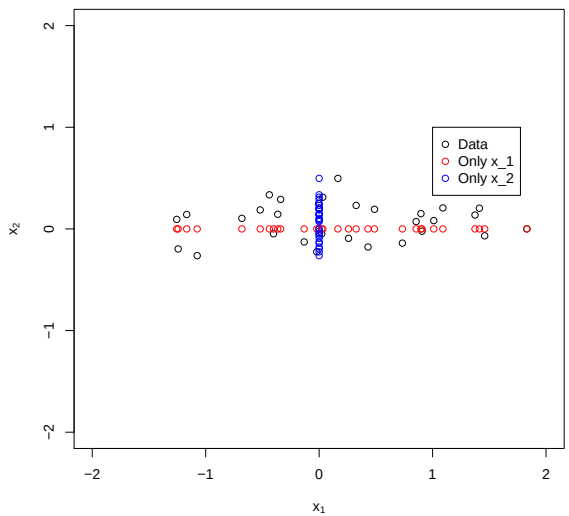
If we knew how to rotate our data, then we would be able to preserve more structure

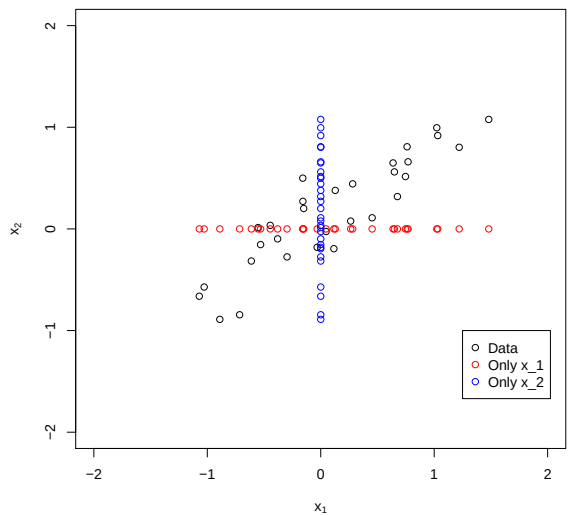


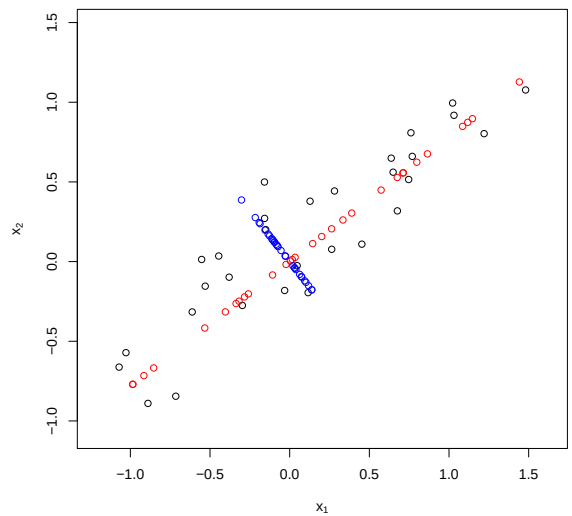


Dimension reduction example

It turns out that PCA gives us exactly this rotation.

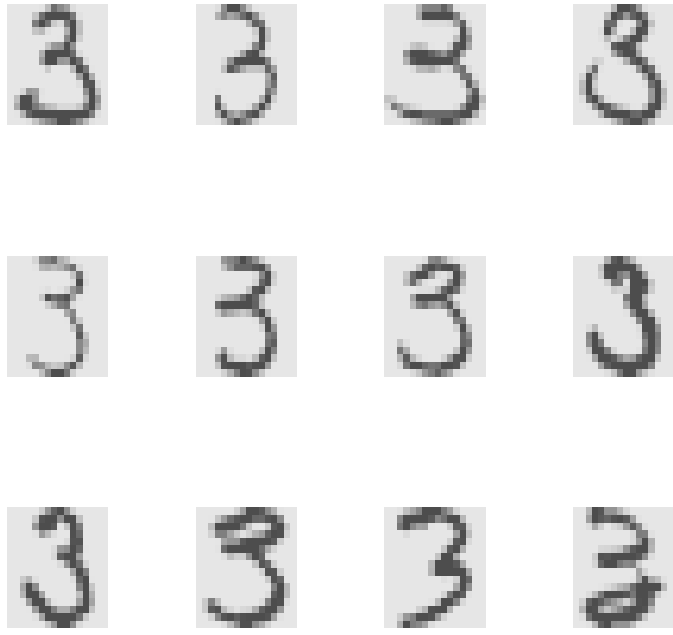






Digits example

PCA



Source: http://www-stat.stanford.edu/∼tibs/ElemStatLearn/

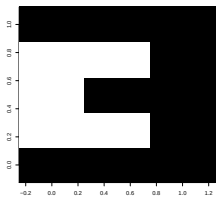
The data: 658 handwritten 3s each drawn by a different person

Each image is 16x16 pixels, each taking grayscale values between -1 and 1.

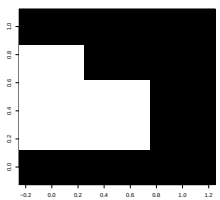
PCA

Think about each pixel location as a measurement

Consider these simple drawings of 3's. We convert this to an observation in a matrix by unraveling it along rows

$X_1 =$ 

$$X_1 = [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1]^\top$$

$X_2 =$ 

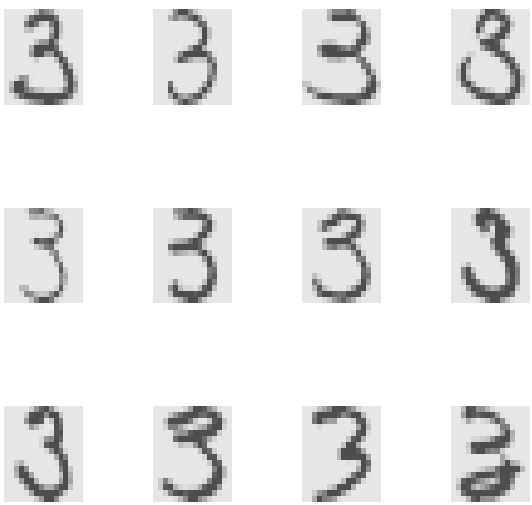
$$X_2 = [1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1]^\top$$

(Here, let black be 1 and white be 0)

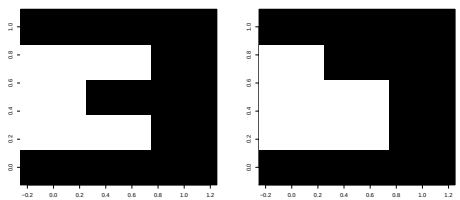
PCA

We will consider digits with...

- more pixels ($p = 256$)
- a continuum of intensities



Vs.



PCA

Here is how to compute PCA via the SVD

```
#subtract off the column means
threesCenter = scale(threes,scale=FALSE)

svd.out = svd(threesCenter)

pcs    = svd.out$v
scores = svd.out$u%*%diag(svd.out$d)
```

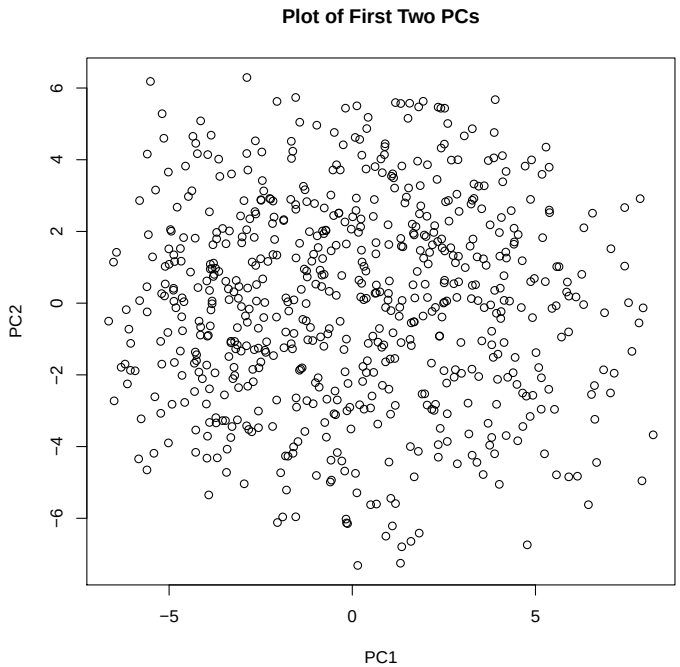
Or, using SAS

```
proc factor;
run;
```

(Technically, these are doing two different things as SAS is also dividing by the sample standard deviation)

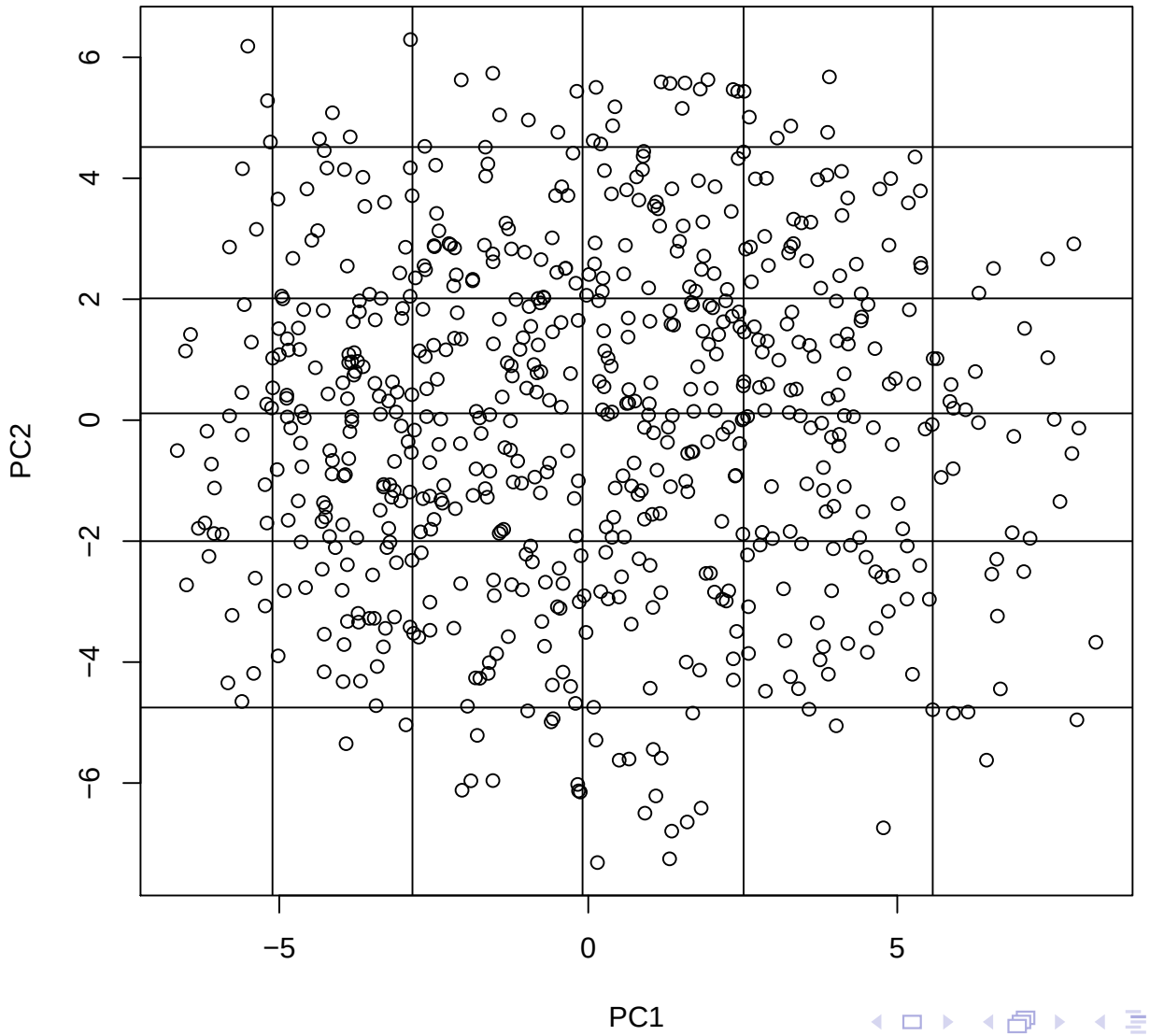
PCA

We can plot the scores of the first two principal components versus each other:

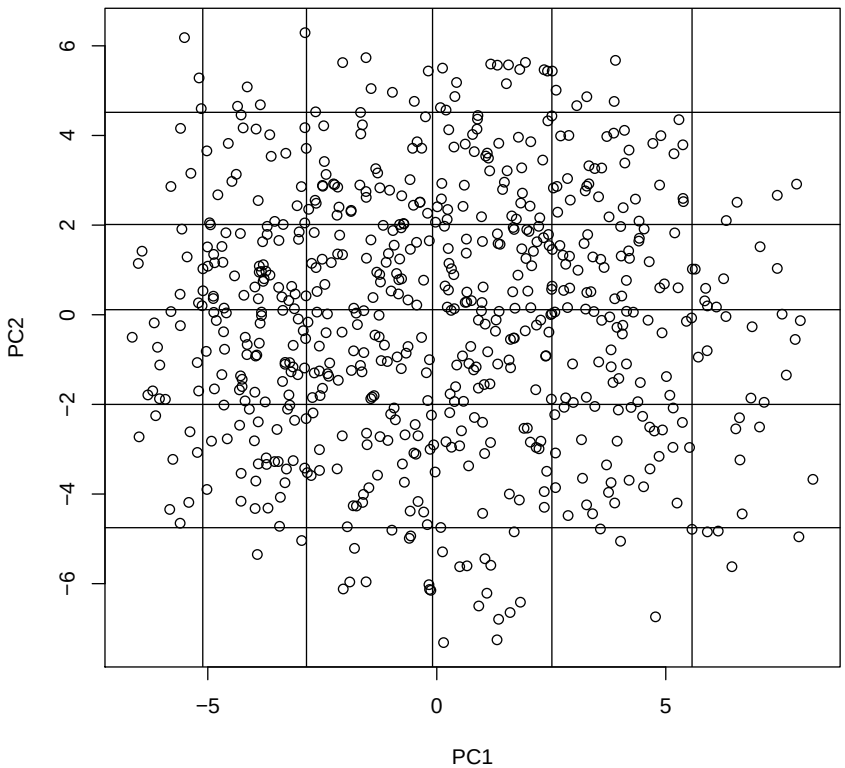


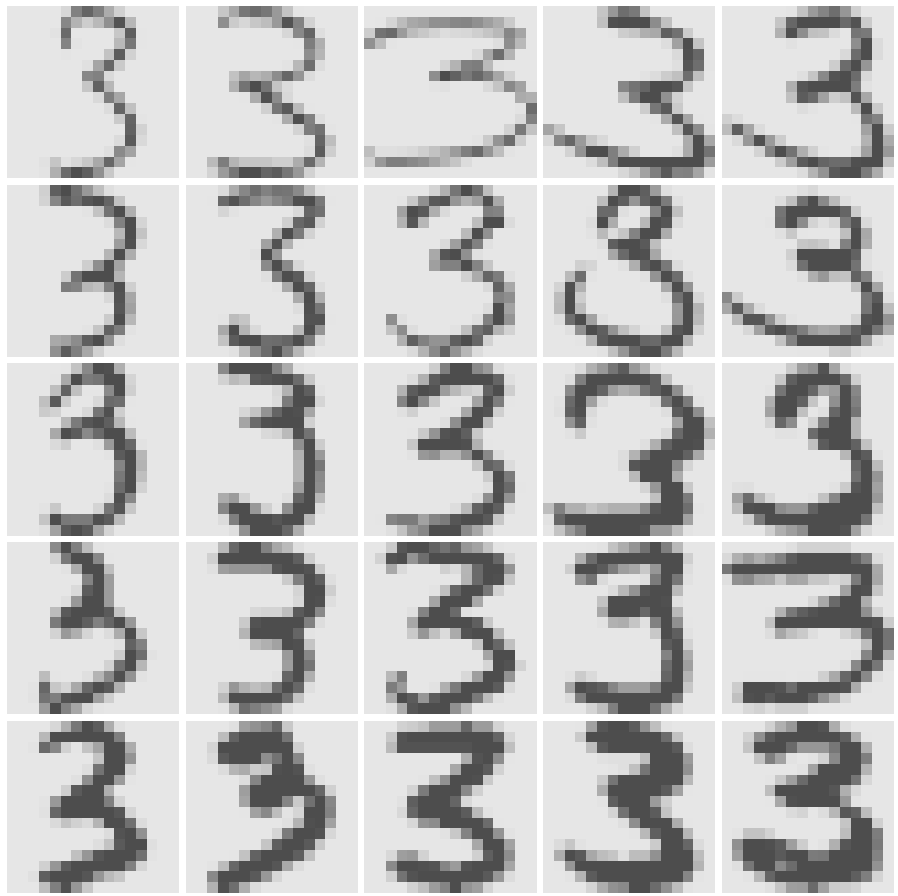
Note: Each circle in this plot represents a hand written '3'.

PCA



PCA

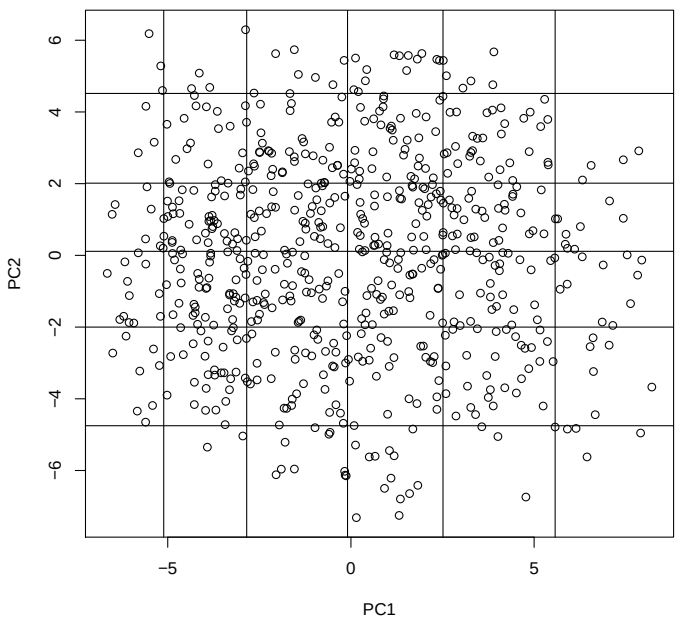


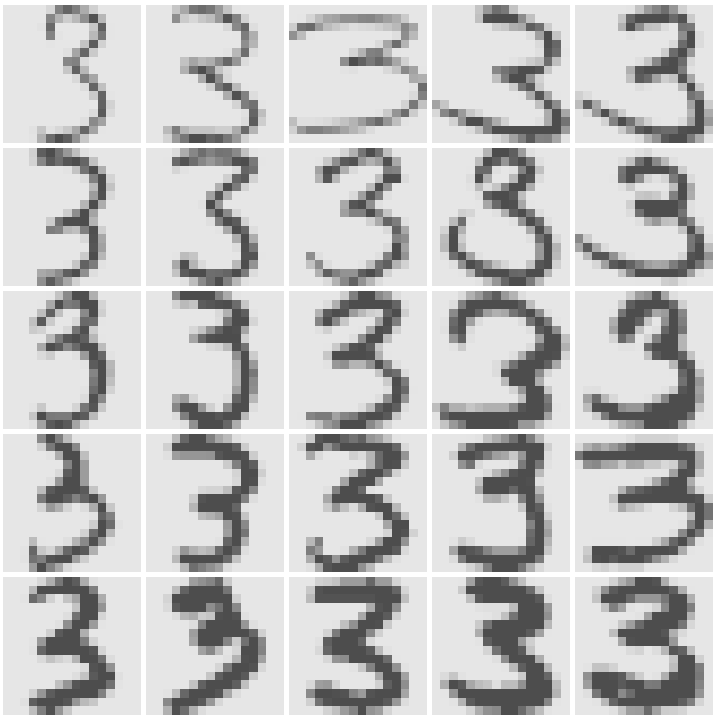


PCA

The 3's get lighter as the location on PC2 increases.

The 3's get more elongated as the location along PC1 increases





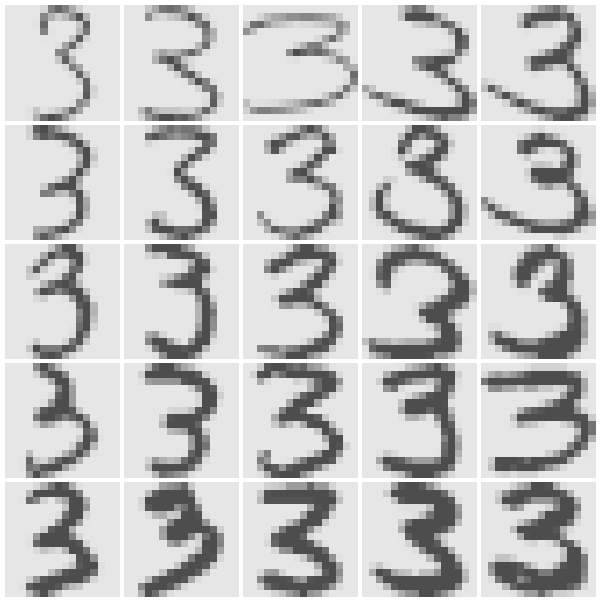
PCA

Each number represents a vector in $\mathbb{R}^{256}$

(as each square is 16x16 pixels)

However, hopefully we can reduce this number by re-expressing the digits in PC-land

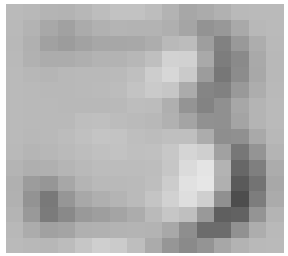
(For instance, the top-right pixel is always 0 and hence that explanatory variable is uninteresting)



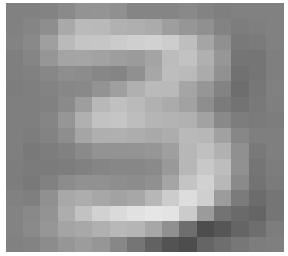
PCA loadings

Lastly, we can also look at the loadings as well:

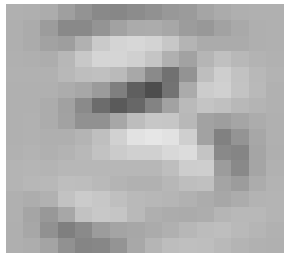
(White means a small number, black a large number)



1st PC: Takes a compact 3 and elongates it



2nd PC: Takes a typical 3 and shifts it



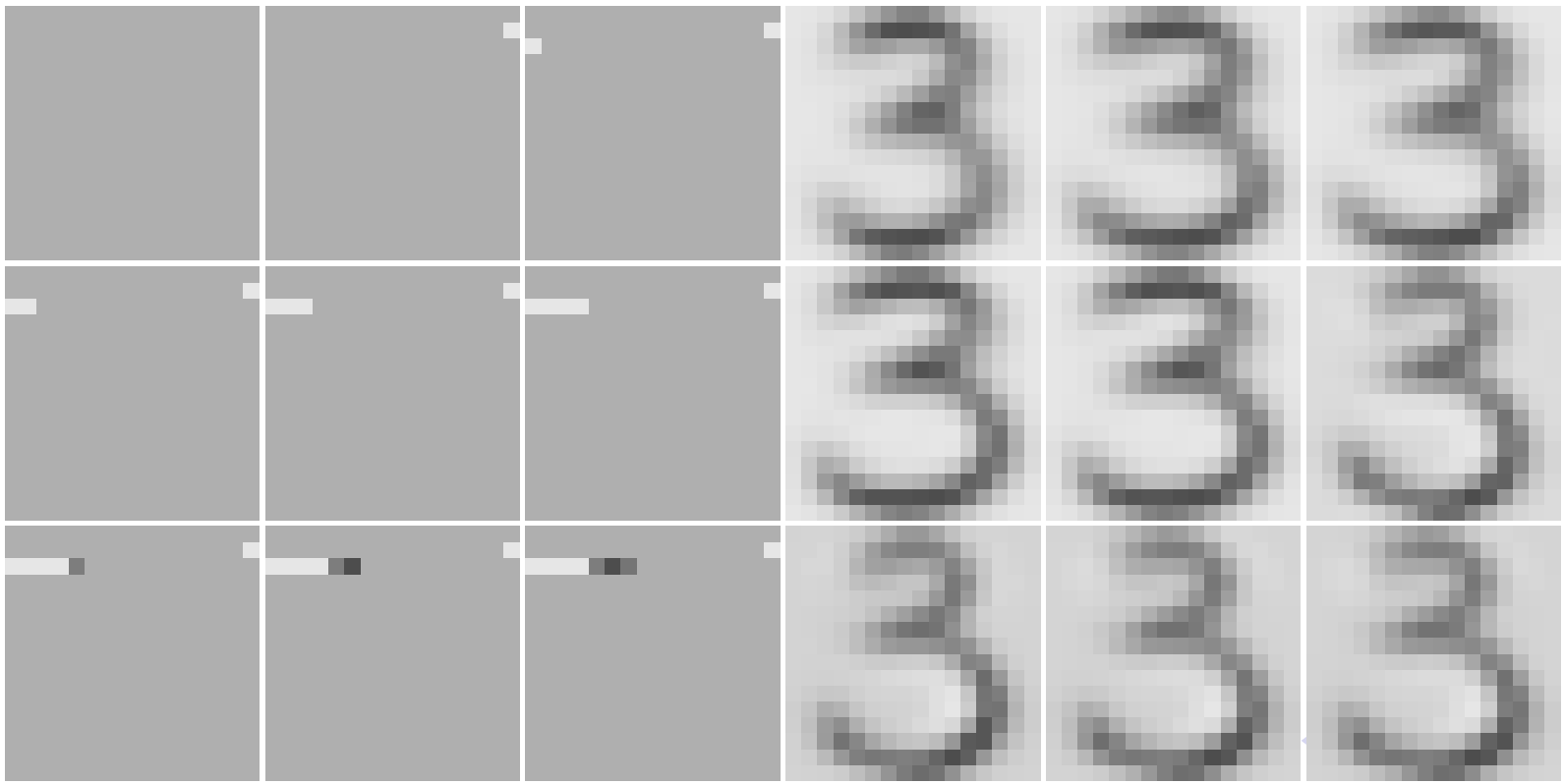
3rd PC: Takes a typical 3 and tips it

Reconstruction

Let's pick one of the 3's in the data set and "rebuild" it using two different representations:

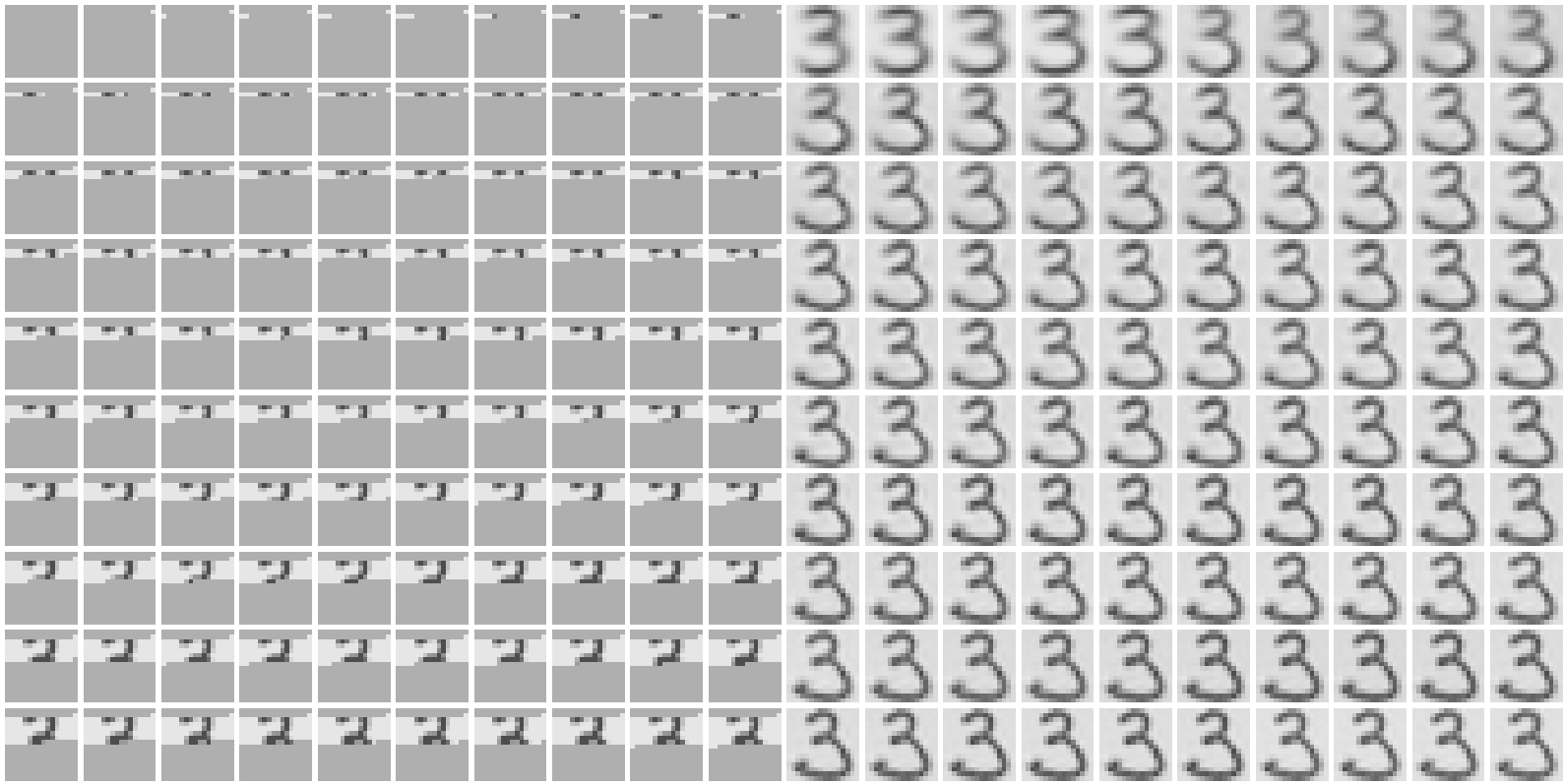
- Pixels
- PCA

Using 9 axis dimensions



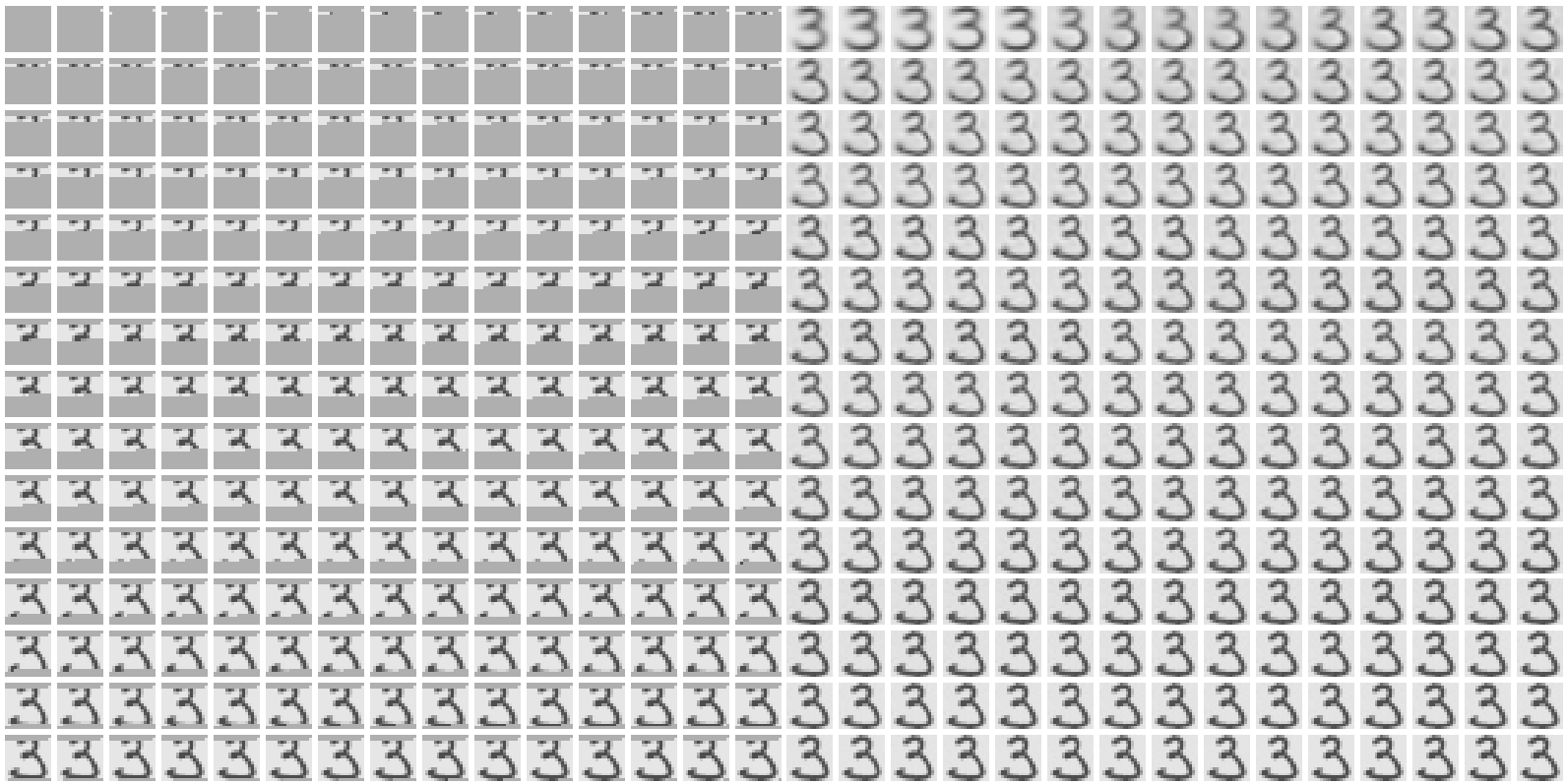
Reconstruction

Using 100 axis dimensions

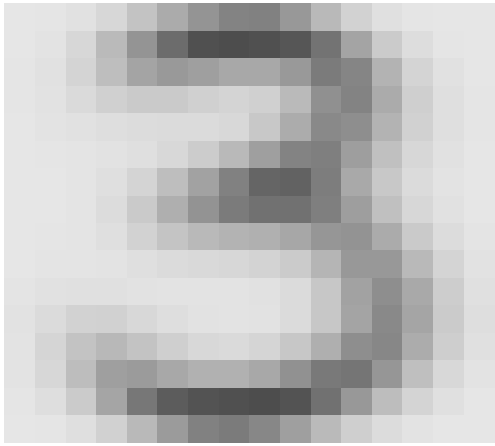


Reconstruction

Using 225 axis dimensions



Reconstruction



This is the mean (From centering $\mathbb{X}$: $(\mathbb{X} - \overline{\mathbb{X}}) = UDV^\top$)
(that is, the origin of the PCA axis, or $\overline{\mathbb{X}}$)[1]

[1] Technically, $\overline{X}_i$ for any i

PCA

(Warning: I'm just including this for interested parties)

If we want to find the first K principal components, the relevant optimization program is:

$$\min_{\mu,(Z_i),V_K} \sum_{i=1}^{n} ||X_i - \mu - V_K Z_i||^2$$

This representation is important

It shows that we are trying to reconstruct lower dimensional representations of the explanatory variables

PCA

$$\min_{\mu,(Z_i),V_K} \sum_{i=1}^{n} ||X_i - \mu - V_K Z_i||^2$$

We can partially optimize for μ and (Z_i) to find

- $\hat{\mu} = \overline{X}$
- $\hat{Z}_i = V_K^\top (X_i - \hat{\mu})$

We can find

$$\min_{V} \sum_{i=1}^{n} \left|\left|(X_i - \hat{\mu}) - VV^\top (X_i - \hat{\mu})\right|\right|^2$$

where V is constrained to be orthogonal

(This is the so called Steifel manifold of rank-K orthogonal matrices)

The solution is given by the singular vectors V

Example: Facial recognition

Images

- There are 575 total images
- Each image is 92×112 pixels and grey scale
- These images come from the Sheffield face database

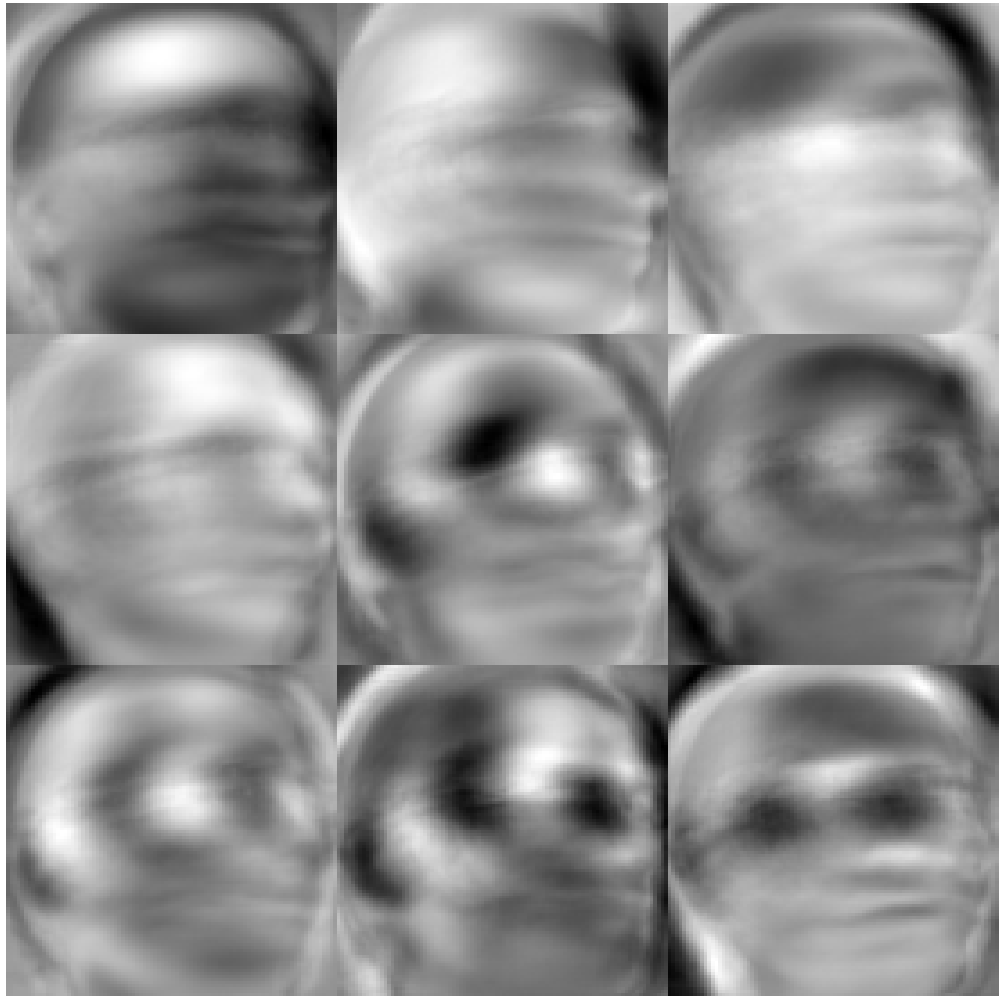
  (See `http://www.face-rec.org/databases/` for this and other databases)

Faces






PCA loadings



Reconstruction

"Rebuilding" a particular image using the PCA representation

Reconstruction