LECTURE 22

Principal Component Analysis

Decomposing high-dimensional data with the tools of linear algebra.

Data 100/Data 200, Fall 2020 @ UC Berkeley

Anthony D. Joseph and Fernando Pérez

(content by Josh Hug, John DeNero, Sam Lau, Suraj Rampure)

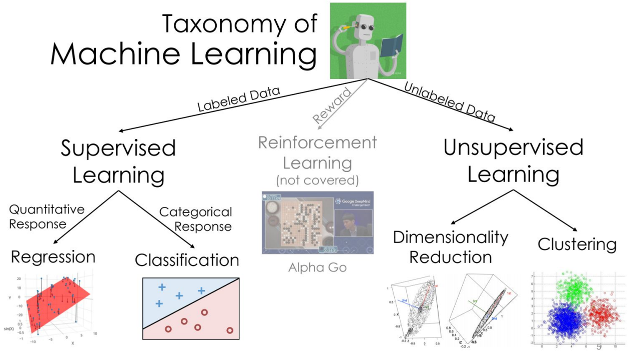
Overview

This lecture is the first on **Unsupervised Learning**

- See the previous lecture for a more detailed look at the Taxonomy of ML
- It will be very linear algebra heavy – refer to the linear algebra resources post on Piazza if necessary

Agenda:

- Motivating PCA
- Introducing the Singular Value Decomposition and its theory
- Defining principal components and exploring their properties and usefulness



from Joseph Gonzalez

Motivation

Dimensionality

duplicated

Consider the data shown. How many dimensions does this data have?

- 3, because 2 weight columns are redundant not 4!
- In linear algebra terms, we'd observe that this matrix has **rank** 3

- More generally: Can think of a dataset's *dimensionality* as the ***rank*** of the matrix representing the data

Height (in)	Weight (kg)	Weight (lbs)	Age
65.8	51.3	113.0	17
71.5	61.9	136.5	21
69.4	69.4	153.0	18

3 dimensions

Visualizing High-Dimensional Data

In lectures/labs/HWs, we've used visualizations for Exploratory Data Analysis

- Works very well for 2 dimensional data
- Gets harder as dimensionality goes above 2
  - Can use hue, size, time, etc. to show more dimensions, but only poorly

To explore associations between two quantitative variables, can do scatterplots of only those two variables at a time

(Demo: Body Measurements)

Visualizing High-Dimensional Data

To explore clusters of similar observations: try reducing to two dimensions

- If the rank of the data matrix is larger than 2, you will lose information!
- One idea: Pick two attributes, namely those with highest variance
  - Intuition: More likely to differentiate observations
- Another idea: Principal Component Analysis

(Demo: House of Representatives Votes)

Visualizing High-Dimensional Data

To explore clusters of similar observations: try reducing to two dimensions

- If the rank of the data matrix is larger than 2, you will lose information!
- One idea: Pick two attributes, namely those with highest variance
    - Intuition: More likely to differentiate observations
- Another idea: Principal Component Analysis

Goal: Plot high dimensional data as a 2 dimensional approximation that results from a linear combination of attributes

Related Goal: Determine whether this two-dimensional plot is really showing the variability in the data. (If not, be wary of conclusions drawn using this plot)

Warmup

You measure the width, length, area, and perimeter of 100 rectangular backyards.

What do you expect to be the rank of the observed data matrix?

If less than 4, what smaller matrices could you multiply together to recover the data matrix?

width	length	area	perimeter
20	20	400	80
16	12	192	60
24	12	288	72
...			
25	24	600	98

100 x 4

Question: What's the Rank?

The rank is 3 - while area is redundant, it cannot be computed using linear operations

There are many possible matrices we could multiply to get back full results

- Most natural choices are given below

100 x 4

width	length	area	perimeter
20	20	400	80
16	12	192	60
24	12	288	72
...			
25	24	600	98

=

100 x 3

width	length	area
20	20	400
16	12	192
24	12	288
...		
25	24	600

x

3 x 4

1	0	0	2
0	1	0	2
0	0	1	0

Singular Value Decomposition

Singular Value Decomposition

Singular value decomposition is an important concept in linear algebra

- We assume you have taken (or are taking) a linear algebra course
- Will not explain in its entirety

Instead, we will present things backwards from how you'd usually learn SVD

- Will give a better context for what SVD means in the context of data science
- Not recommended as a first exposure to SVD

Redundancy and Decomposition

Earlier, we saw that our rectangle data could be decomposed into two matrices:

- The data without the perimeter
- A matrix that transforms the data from 3D into 4D, computing the perimeter in the process

Total amount of information stored is less!

Required manual work on our part to identify the magical transformation matrix

rectangle

m x 4

100 x 4 = 400

=

rectangle.iloc[:, 0:3]

m x 3

x

Transformation matrix

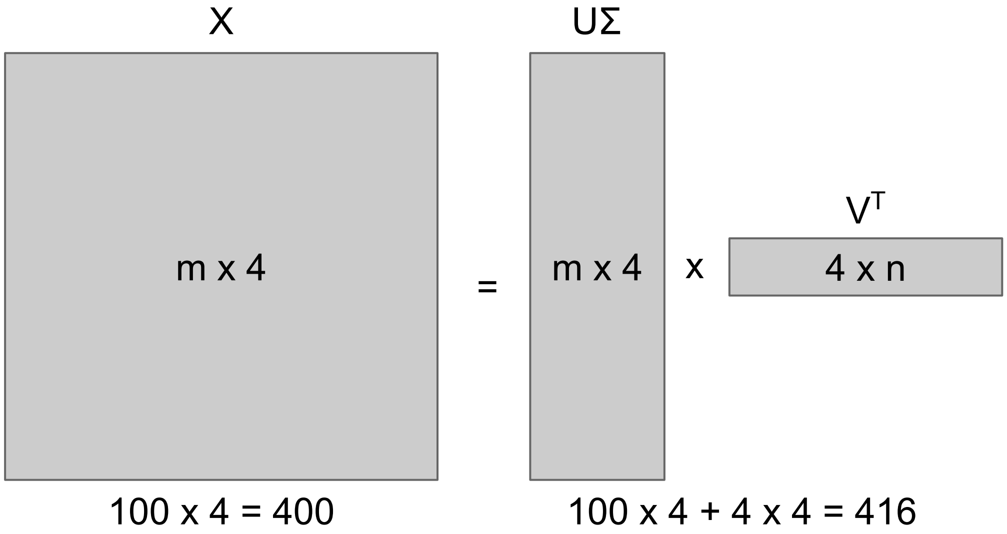
1	0	0	2
0	1	0	2
0	0	1	0

100 x 3 + 3 x 4 = 312

Singular Value Decomposition

The “Singular Value Decomposition” technique will automatically do a similar transformation for us

- Let’s try it out in Python



Singular Value Decomposition Output

100 x 3

width	length	area
20	20	400
16	12	192
24	12	288
...		
25	24	600

x

3 x 4

1	0	0	2
0	1	0	2
0	0	1	0

The “Singular Value Decomposition” technique will automatically do a similar transformation for us

What is different about this decomposition from the one we did manually?

X

m x 4

100 x 4 = 400

=

UΣ

m x 4

100 x 4

-56	4.1	-0.77	0

-19	-6.7	3.07	0

x

V^T

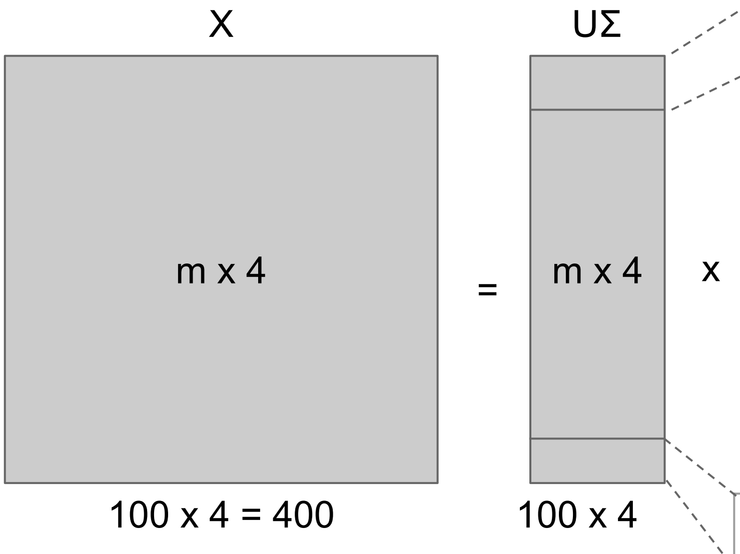
-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.0008	0.0008
-0.67	-0.67	0	0.33

+ 4 x 4 = 416

Singular Value Decomposition Output

What is different about this decomposition from the one we did manually?

- The 4th dimension is all zeroes
- The other 3 dimensions of our “data” are NOT the width, length, and area
- The SVD results are bigger (100 x 4 instead of 100 x 3, 4 x 4 instead of 3 x 4)
- *Transformation matrix has smaller, strange values*



-56	4.1	-0.77	0

V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.0008	0.0008
-0.67	-0.67	0	0.33

+ 4 x 4 = 416

-19	-6.7	3.07	0

SVD Question

For this data, it is guaranteed that the last column of UΣ is all zeros. Suppose we simply delete it from the matrix. What else can we throw away?

A. First row of V^T

B. First column of V^T

C. Last row of V^T

D. Last column of V^T

last col

-56	4.1	-0.77	0

X: m x 4, 100 x 4 = 400

=

UΣ: m x 4, 100 x 4

x

V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.0008	0.0008
-0.67	-0.67	0	0.33

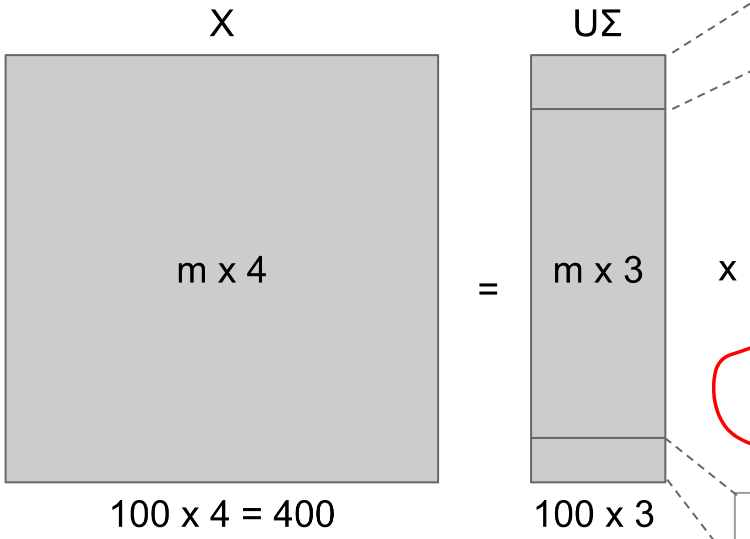
+ 4 x 4 = 416

-19	-6.7	3.07	0

SVD Question

For this data, it is guaranteed that the last column of UΣ is all zeros. Suppose we simply delete it from the matrix. What else can we throw away?

C. Last row of V^T



-56	4.1	-0.77

V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.0008	0.0008
-0.67	-0.67	0	0.33

+ 3 x 4 = 312

-19	-6.7	3.07

Singular Value Decomposition Theory

Manual Decomposition

Before, we took our data and manually created a 3D to 4D transformation matrix

Decomposed data into:

- Truncated data
- Transformation matrix

width	length	area	perimeter
8	6	48	28
2	4	8	12
1	3	3	8
...			
2	6	12	16

=

width	length	area
8	6	48
2	4	8
1	3	3
...		
2	6	12

x

Transformation matrix

1	0	0	2
0	1	0	2
0	0	1	0

Singular Value Decomposition

Then we used SVD which automatically created a 3D to 4D transformation matrix

Decomposed data into:

- Mysteriously rescaled data UΣ
- Different transformation matrix V^T

width	length	area	perimeter
8	6	48	28
2	4	8	12
1	3	3	8
...			
2	6	12	16

=

UΣ

??	??	??
-56	4.1	-0.77
-1.4	-5.6	1.6
-7.4	-5.1	1.5
...		
-19	-6.7	3.07

x

V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.0008	0.0008

Singular Value Decomposition

The truth about UΣ and V^T:

- Σ is a **diagonal** matrix - Contains the so-called "**singular values**" of X
- The columns of U and V are each an **orthonormal set**

Let's define these bolded terms

width	**length**	**area**	**perimeter**
8	6	48	28
2	4	8	12
1	3	3	8
...			
2	6	12	16

=

UΣ

??	**??**	**??**
-56	4.1	-0.77
-1.4	-5.6	1.6
-7.4	-5.1	1.5
...		
-19	-6.7	3.07

x

V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.00 08	0.00 08

Diagonal Matrices and Σ

A diagonal matrix is a matrix with zeros everywhere except possibly the diagonal

- Multiplying by a diagonal matrix is equivalent to scaling the columns

$$\begin{bmatrix} | & | & | \\ \vec{c}_1 & \vec{c}_2 & \vec{c}_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix} = \begin{bmatrix} | & | & | \\ a_1\vec{c}_1 & a_2\vec{c}_2 & a_3\vec{c}_3 \\ | & | & | \end{bmatrix}$$

In Σ, the singular values appear in decreasing order

- Singular values are always non-negative
- Singular values beyond rank r will be zero
- Example of singular values for our rectangle data:
    - Note the 4th singular value is 0

$$\Sigma = \begin{bmatrix} 363 & 0 & 0 & 0 \\ 0 & 63 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(python)

Singular Value Decomposition

The truth about UΣ and V^T:

- Σ is a **diagonal** matrix. Contains the so-called "**singular values**" of X
- The columns of U and V are each an **orthonormal set**

Let's define these bolded terms

width	length	area	perimeter
8	6	48	28
2	4	8	12
1	3	3	8
		...	
2	6	12	16

=

UΣ

??	??	??
-56	4.1	-0.77
-1.4	-5.6	1.6
-7.4	-5.1	1.5
	...	
-19	-6.7	3.07

x

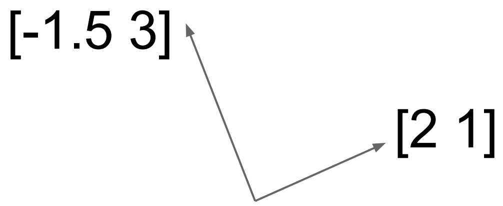
V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.00 08	0.00 08

Orthogonality, Dot Products, Vector Length

Two **orthogonal** vectors:

- Meet at a right angle
- Have a dot-product of zero



$(-1.5) \cdot (2) + (3) \cdot (1) = 0$

A **unit vector** has length 1

Side fact: The length of a vector v is the square root of v • v (i.e. the square root of its L2 norm)

Orthonormality

A set of vectors is said to be an **orthonormal set** if:

- All of the vectors are unit vectors, i.e. have length 1
- All of the vectors are orthogonal

Orthonormality

A set of vectors is said to be an **orthonormal set** if:

- All of the vectors are unit vectors, i.e. have length 1
- All of the vectors are orthogonal

V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.00 08	0.00 08

Given V^T, to determine if the columns of V form an orthonormal set, we verify that:

- Dot product of any row of V^T with itself is 1
- Dot product of any row of V^T with any other row is 0

Orthonormality

A set of vectors is said to be an **orthonormal set** if:

- All of the vectors are unit vectors, i.e. have length 1
- All of the vectors are orthogonal

The columns of U and V each form an orthonormal set

- Dot product of any column with itself is 1
- Dot product of any column with any other column is 0

Side fact: If the matrix is square and its columns form an orthonormal set, then the transpose of such a matrix is also its inverse

V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.00 08	0.00 08

Let's try verifying these properties in Python

Principal Components

Principal Component

*: There's an important technical detail we'll need to fix first (coming in a few slides)

When performing SVD, we've left the columns in the "data" matrix UΣ unnamed.

- Their common name: "**Principal Components**"*
- Example: Second column of UΣ is "2nd principal component" of original matrix

width	length	area	perimeter
8	6	48	28
2	4	8	12
1	3	3	8
...			
2	6	12	16

=

UΣ

PC1	PC2	PC3
-56	4.1	-0.77
-1.4	-5.6	1.6
-7.4	-5.1	1.5
...		
-19	-6.7	3.07

x

V^T

-0.15	-0.13	-0.81	-0.55
-0.19	-0.18	0.59	-0.76
-0.71	0.71	0.0008	0.0008

Interpreting Principal Components

Let’s look at a geometric interpretation of:

- Principal Components
- The rank-k approximation as a whole

This is easiest if we work with a simple dataset with 2 attributes

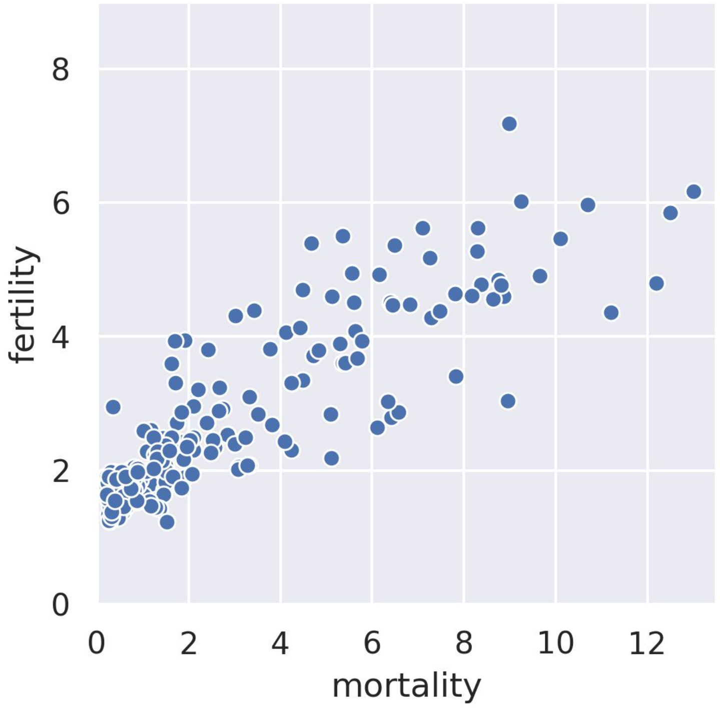
Interpreting Principal Components

Let's look at a geometric interpretation of:

- Principal Components
- The rank-k approximation as a whole

This is easiest if we work with a simple dataset with 2 attributes

We'll use the dataset to the right showing the age 5 mortality rates and fertility rates for most of the countries around the world



Data, Rank 2, and Rank 1 Approximation

Below, we see the original data, along with rank 2 and rank 1 approximations

- As before, we see that approximation at rank 1 is still fairly decent
- The degree to which it is decent depends on how strong the relationship is between mortality and fertility
- We can get a lot more insight by looking at things visually

country	mortality	fertility
Afghanistan	6.820	4.48
Albania	1.330	1.71
Algeria	2.390	2.71
Angola	8.310	5.62
Antigua and Barbuda	0.816	2.04

Original Data

country	mortality	fertility
Afghanistan	6.820	4.48
Albania	1.330	1.71
Algeria	2.390	2.71
Angola	8.310	5.62
Antigua and Barbuda	0.816	2.04

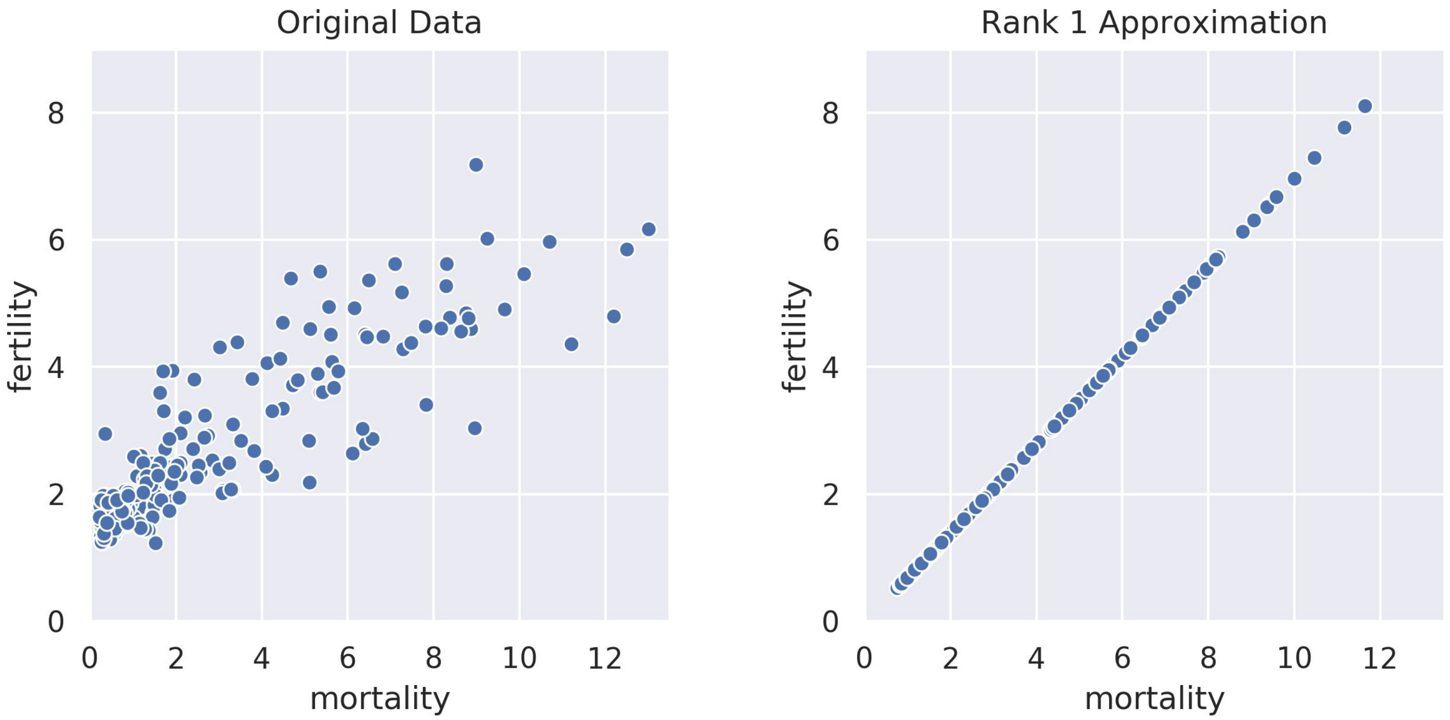
Rank 2 Approximation

country	mortality	fertility
Afghanistan	6.694067	4.660869
Albania	1.697627	1.182004
Algeria	2.880467	2.005579
Angola	8.232160	5.731795
Antigua and Barbuda	1.506198	1.048719

Rank 1 Approximation

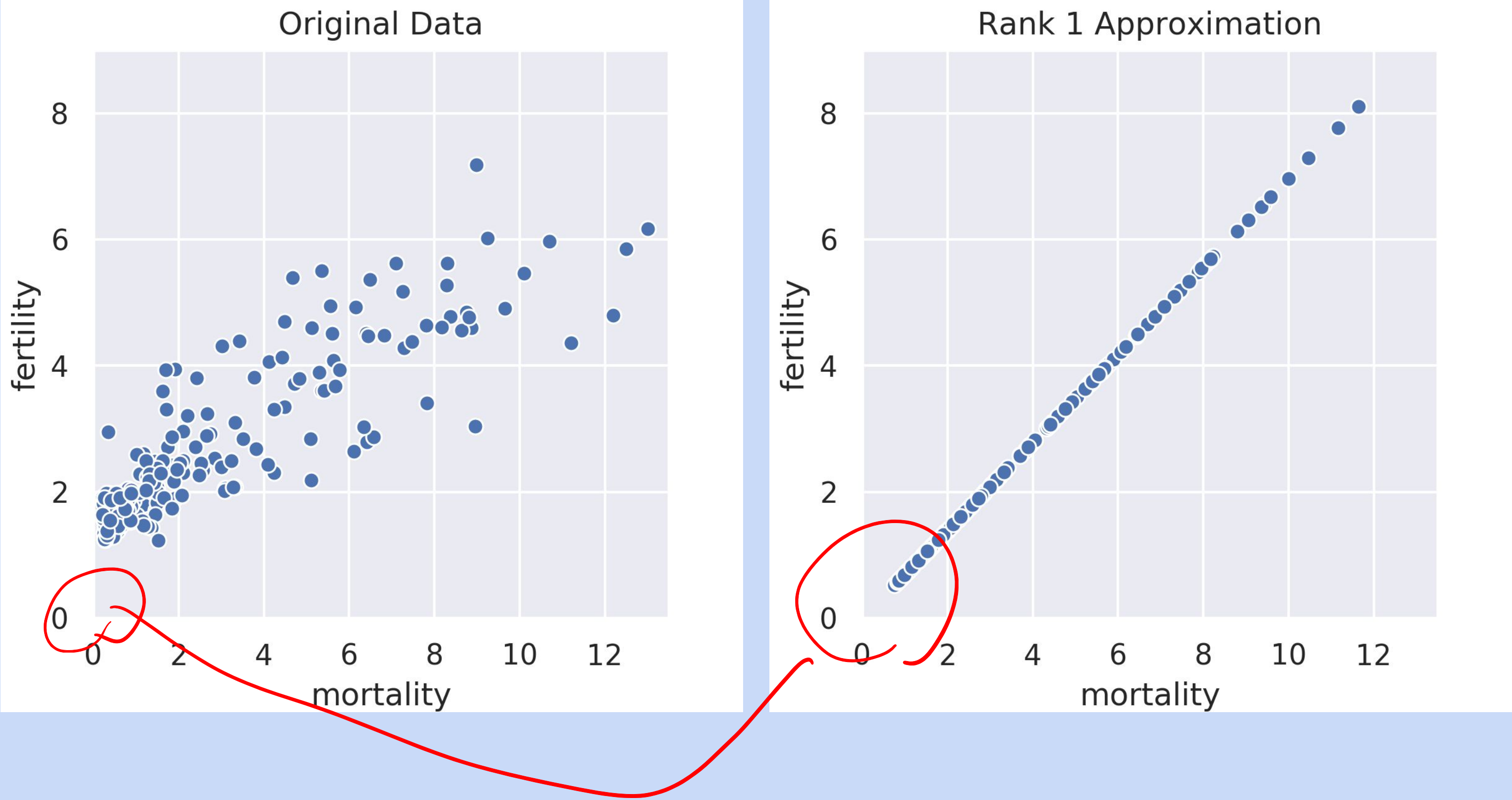
The Rank 1 Approximation Visually

We see that the rank 1 approximation projects the data on to a 1D subspace



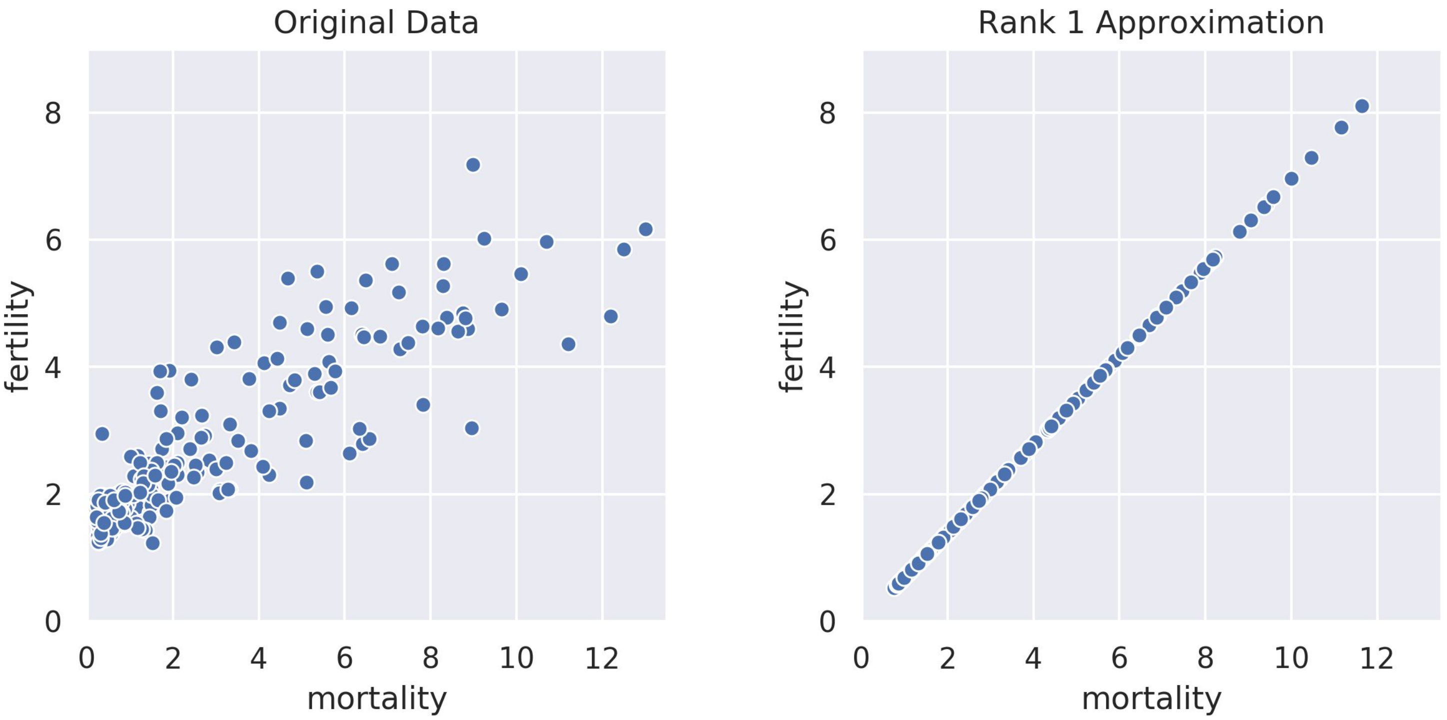
The Rank 1 Approximation Visually

There's a significant issue with our rank 1 approximation. What is it?



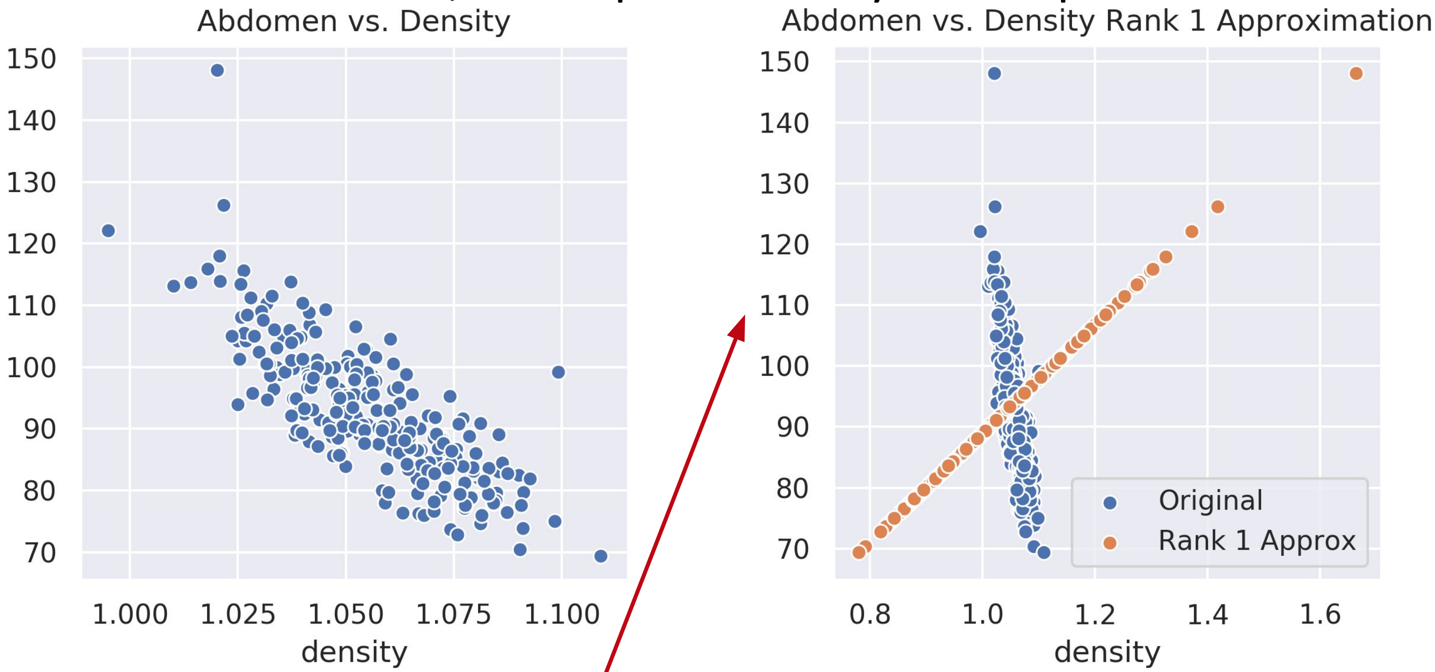
The Rank 1 Approximation Visually

Our approximation goes through the origin, but original data has non-zero y-intercept



More Flawed Example

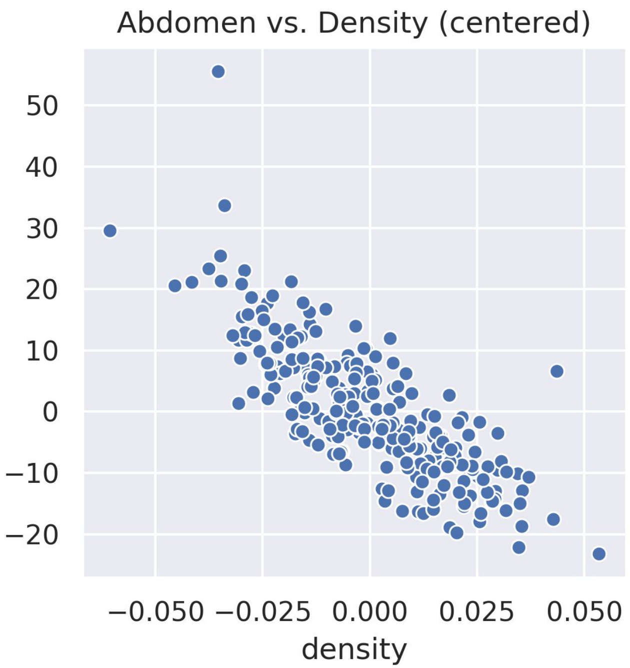
For other datasets, the impact of this y-intercept mismatch can be severe

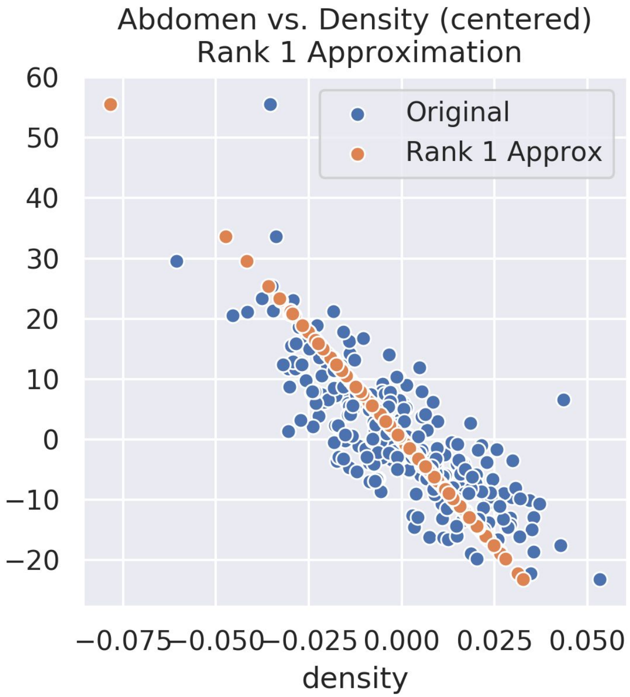


Data looks different on right because of different x-axis

Data Centering

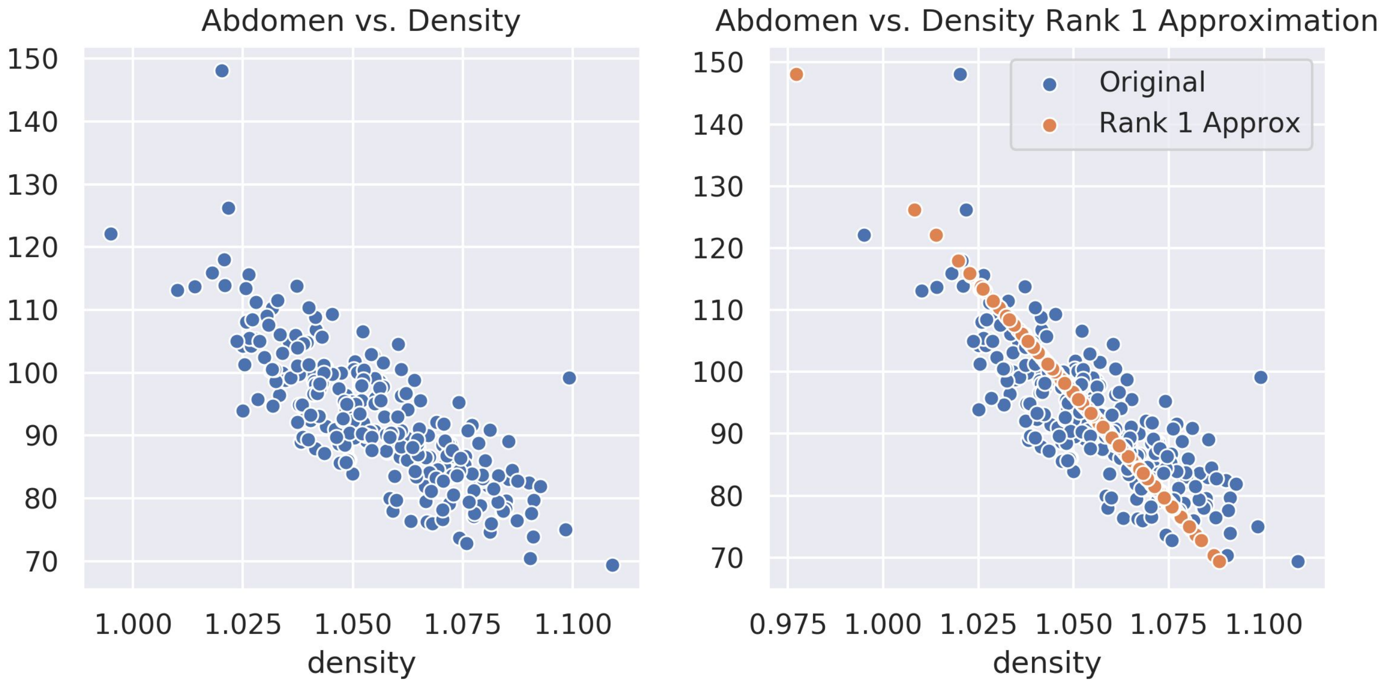
Typically, to deal with this issue we recenter the data by subtracting the mean of each column for all values in that column. Resulting projection is much better!





Data Uncentering

After performing the approximation, you can add back the means to get back to the original x and y scale



Principal Component

When performing SVD, we've left the columns in the "data" matrix UΣ unnamed

- Their common name: "**Principal Components**"
- To get the correct principal components, it's important to center data first!
- Example: Second column of UΣ is "2nd principal component" of centered matrix

width	length	area	perimeter
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
...			
-3.03	1.35	-11.22	-3.36

=

UΣ

PC1	PC2	PC3
-26	0.16	0.81
17	-2.18	3.48
2.32	-3.54	2.00
...		
11.8	-1.61	-2.51

x

V^T

-0.099	-0.073	-0.93	-0.34
0.67	-0.37	-0.26	0.589
0.314	-0.640	0.257	-0.652

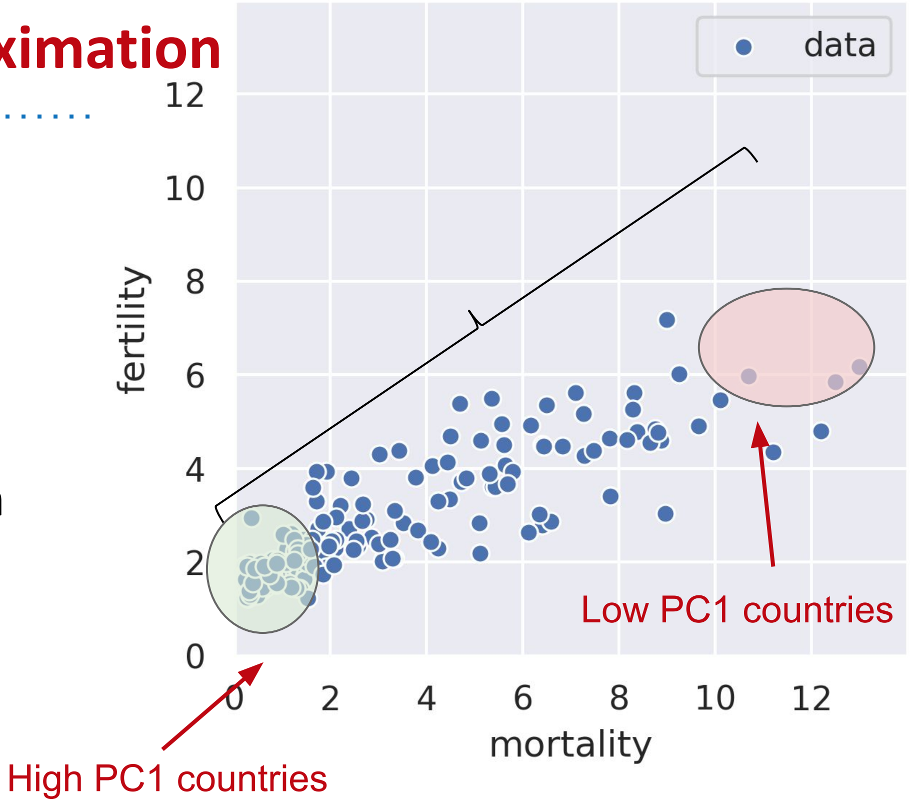
Intuitive Picture of Low Rank Approximation

Consider the maternal fertility vs. child mortality data we discussed earlier

Technically, every country has its own separate fertility and mortality

- Looking at the plot, we see that we can approximate by saying that each country lies on a continuum between "bottom left" and "top right"

In essence, the 1st principal component gives each country a single score along this intuitive axis!



Computing PC1

Given U, Σ, and V^T below, write an expression to compute **PC1** for **Afghanistan**

- Hint the first column of UΣ is the 1st PC (for all countries)

	mortality	fertility
country		
Afghanistan	3.78541	1.704044
Albania	-1.70459	-1.065956
Algeria	-0.64459	-0.065956
Angola	5.27541	2.844044
Antigua and Barbuda	-2.21859	-0.735956
Argentina	-1.94459	-0.495956
Armenia	-1.71459	-1.175956
Australia	-2.68259	-0.935956

U

-0.095374	0.023040
0.045479	-0.044248
0.014340	0.021510
-0.136893	0.085339
0.053670	0.016303
0.045778	0.031204
...	

Σ

43.481242	0.000000
0.000000	8.195009

V^T

-0.929593	-0.368588
-0.368588	0.929593

Computing PC1

Since the principal components are just the columns of UΣ, we just compute the top left entry of UΣ:

- -0.0954 * 43.48 + 0.023 * 0 = -4.15

U

-0.095374	0.023040
0.045479	-0.044248
0.014340	0.021510
-0.136893	0.085339
0.053670	0.016303
0.045778	0.031204
...	

Σ

43.481242	0.000000
0.000000	8.195009

	mortality	fertility
country		
Afghanistan	3.78541	1.704044
Albania	-1.70459	-1.065956
Algeria	-0.64459	-0.065956

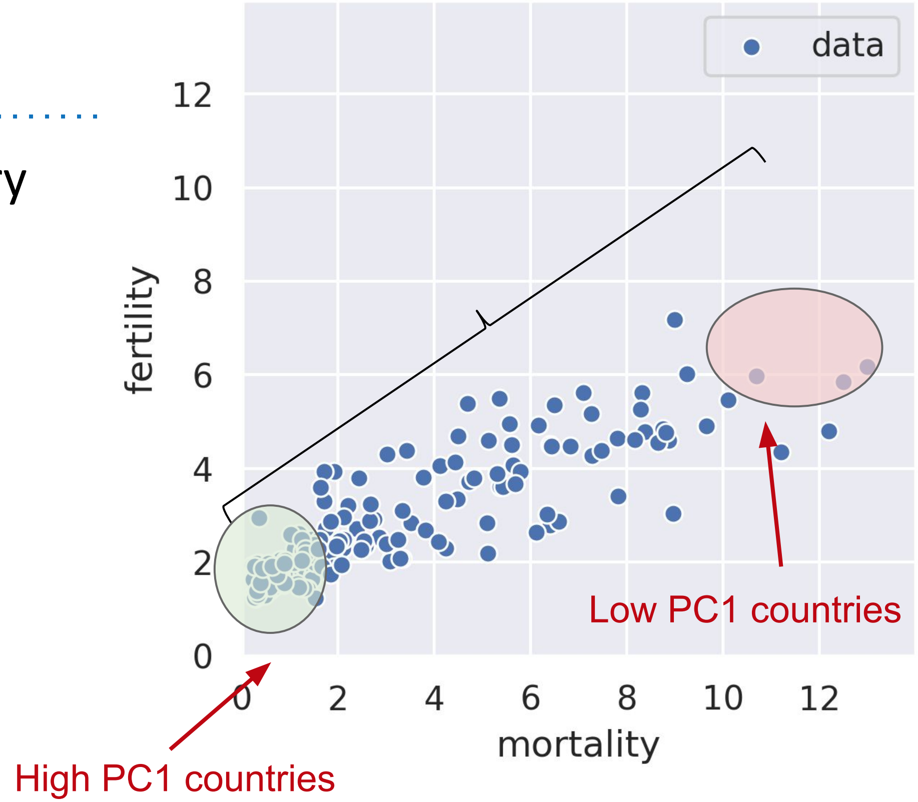
V^T

-0.929593	-0.368588
-0.368588	0.929593

PC1 Computation

Below, we see a table showing PC1 for every country

country	mortality	fertility	pc1
Afghanistan	3.78541	1.704044	-4.146980
Albania	-1.70459	-1.065956	1.977473
Algeria	-0.64459	-0.065956	0.623517
Angola	5.27541	2.844044	-5.952264
Antigua and Barbuda	-2.21859	-0.735956	2.333650
Argentina	-1.94459	-0.495956	1.990481
Armenia	-1.71459	-1.175956	2.027314
Australia	-2.68259	-0.935956	2.838699

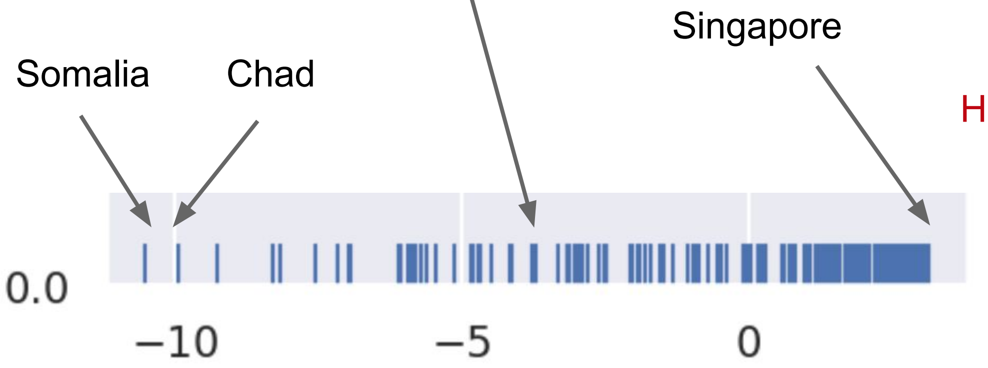


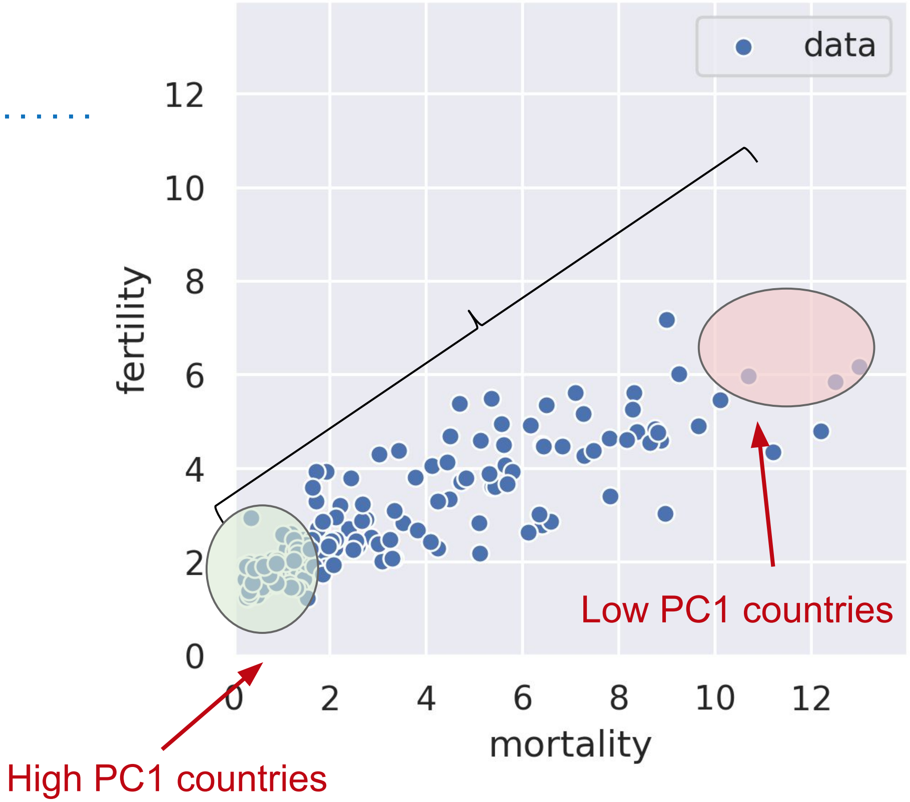
Plotting Countries vs. PC1

Naturally, we can plot countries vs. PC1.

- Only one dimension so we end up with a rug plot

	pc1
country	
Afghanistan	-4.146980





PC1 Back to 2D

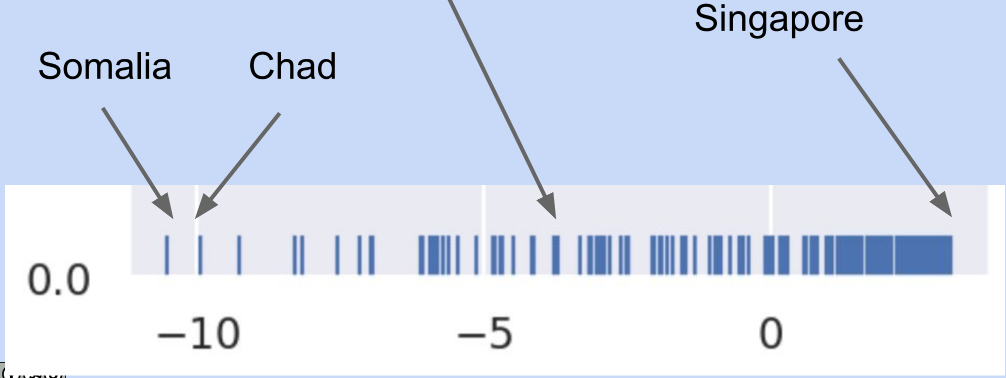
Our rank 1 approximation is essentially just going from PC1 back into 2 dimensions

Give expressions that compute the rank 1 approximation for the fertility and mortality rates for Afghanistan

	pc1
country	
Afghanistan	-4.146980
Albania	1.977473

V^T

-0.929593	-0.368588
-0.368588	0.929593



PC1 Back to 2D

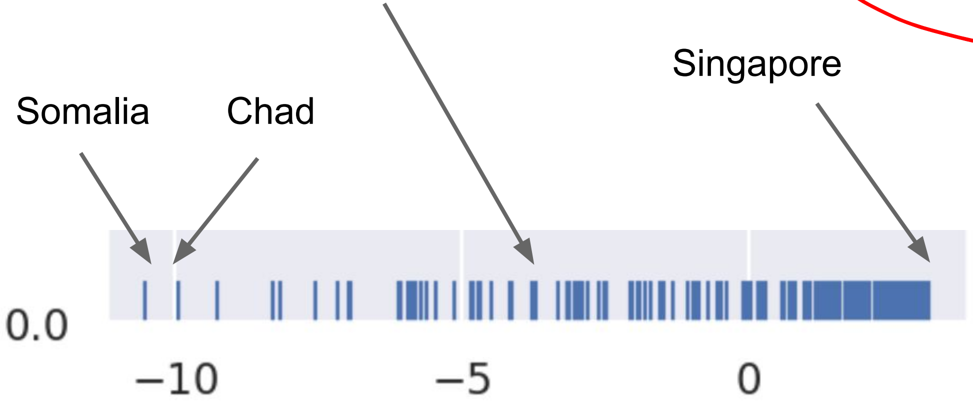
To get our rank 1 estimates for Afghanistan:

- Recall our original data $X = U\Sigma V^T$.
- So just multiply the first value of the column (principal component) of $U\Sigma$ by the first row of V^T.
- Mortality: -4.14 * -0.929 = 3.86
- Fertility: -4.14 *-0.368 = 1.53

	pc1
country	
Afghanistan	-4.146980
Albania	1.977473

V^T

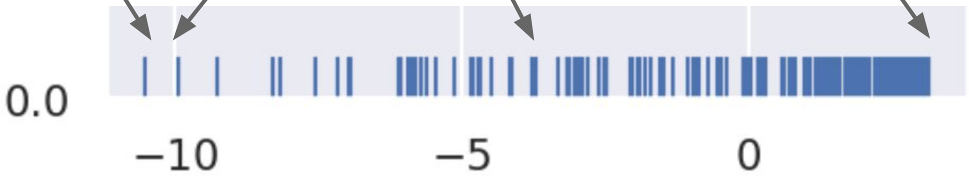
-0.929593	-0.368588
-0.368588	0.929593



PC1 Back to 2D

To get our rank 1 estimates for Afghanistan:

- Recall our original data $X = U\Sigma V^T$.
- So just multiply the first value of the column (principal component) of $U\Sigma$ by the first row of V^T.
- Mortality: -4.14 * -0.929 = 3.86
- Fertility: -4.14 *-0.368 = 1.53



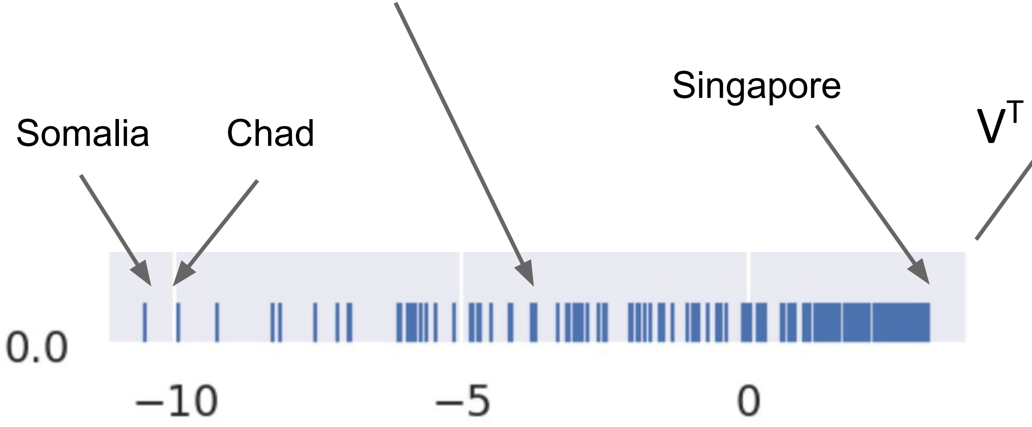
V^T

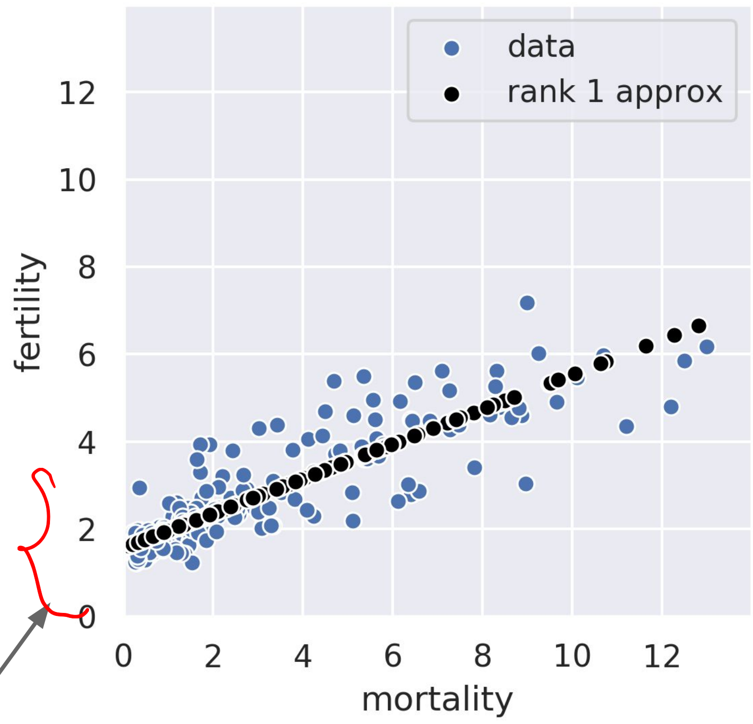
PC1 Back to 2D

The first principal component captures much of the variance, but not all

- Differently worded: Countries live near the line, but not exactly on it

By using 2nd principal component, we can get back original data exactly





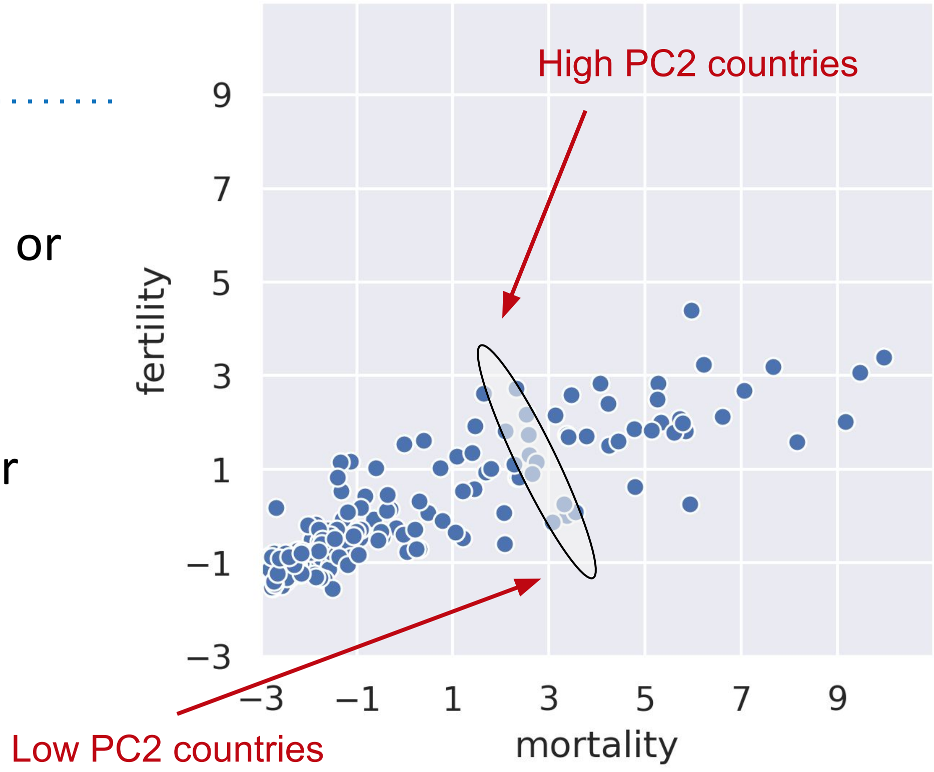
PC2

PC2 is harder to interpret

- It tells you how much you lie “above” or “below” the PC1 line

The gray area shows countries with similar values of PC1

- Countries at the top left of the oval have “large” PC2
- Countries at the bottom left have “small” PC2



Principal Components and Variance

Principal Components

Back to our original rectangle data

Recall: Before we can identify the principal components of this data, we must first center the data by subtracting the mean of each column from that column

width	length	area	perimeter
8	6	48	28
2	4	8	12
1	3	3	8
...			
2	6	12	16

Principal Components

Our centered data is easy to interpret:

- The first observation had a width that was 2.97 greater than the mean, length 1.35 greater than the mean, etc.
- Each entry is given in its original pre-centering units (e.g. meters, square meters, etc.)

width	length	area	perimeter
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
...			
-3.03	1.35	-11.22	-3.36

Singular Value Interpretation (Informally)

SVD decomposes our data into U, Σ and V^T

- Let's talk more about the singular values Σ

width	length	area	perimeter
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
...			
-3.03	1.35	-11.22	-3.36

$$\Sigma = \begin{bmatrix} 197.39 & 0 & 0 & 0 \\ 0 & 27.43 & 0 & 0 \\ 0 & 0 & 23.26 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

U

V^T

Singular Value Interpretation (Informally)

Informally, the i^{th} singular value tells us how valuable the i^{th} principal component will be in reconstructing our original data

- First principal component does most of the work
- Next two principal components contribute about equally
- Fourth principal component does nothing (4th singular value is zero)

width	length	area	perimeter
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
...			
-3.03	1.35	-11.22	-3.36

$$\Sigma = \begin{bmatrix} 197.39 & 0 & 0 & 0 \\ 0 & 27.43 & 0 & 0 \\ 0 & 0 & 23.26 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

U

V^T

Singular Value Interpretation (Informally)

Suppose we do SVD of a 6 dimensional dataset and get back the singular values shown, what might be the most appropriate rank to use for our approximation?

$$\Sigma = \begin{bmatrix} 805 & 0 & 0 & 0 & 0 & 0 \\ 0 & 719 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Singular Value Interpretation (Informally)

Suppose we do SVD of a 6 dimensional dataset and get back the singular values shown, what might be the most appropriate rank to use for our approximation?

- 2? We don't get nearly as much additional accuracy by also using the next 4 principal components

$$\Sigma = \begin{bmatrix} 805 & 0 & 0 & 0 & 0 & 0 \\ 0 & 719 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Singular Value Interpretation (More Formally)

Formally, the i^{th} singular value tells us how much of the **variance** is captured by the i^{th} principal component

To understand this, first let's define the **total variance** of your data as the sum of the individual variances of the attributes

width	length	area	perimeter
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
...			
-3.03	1.35	-11.22	-3.36

Width variance: 7.7

Length variance: 5.3

Area variance: 338.7

Perimeter variance: 50.8

(You don't have enough information on this slide to compute these quantities; we used the full notebook to do so.)

Total variance: 402.56

Singular Value Interpretation (More Formally)

Formally, the i^{th} singular value tells us how much of the **variance** is captured by the i^{th} principal component

- The amount of variance captured by the i^{th} principal component is equal to: (i^{th} singular value)^2 / N

Variance captured by 1st PC: $197.39^2/100 = 389.63$

Variance captured by 2nd PC: $27.43^2/100 = 7.52$

Variance captured by 3rd PC: $23.26^2 / 100 = 5.41$

$$\begin{bmatrix} 197.39 & 0 & 0 & 0 \\ 0 & 27.43 & 0 & 0 \\ 0 & 0 & 23.26 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Total variance: 402.56

width	**length**	**area**	**perimeter**
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
...			
-3.03	1.35	-11.22	-3.36

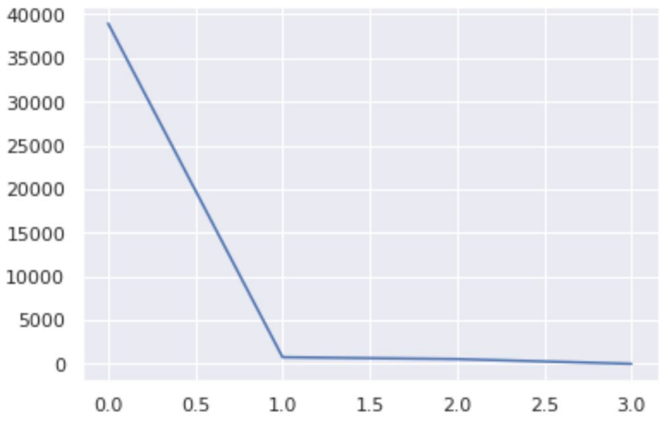
Variance Fraction Arrays and Scree Plots

To get a sense of the relative importance of each principal component, we can compute the fraction of the variance captured in each coordinate:

```
np.round(s**2 / sum(s**2), 2)
```

```
array([0.97, 0.02, 0.01, 0.  ])
```

Or we can plot them on an axis in what is known as a “scree plot”




```
plt.plot(s**2);
```

Principal Component Analysis

Principal Component Analysis (PCA) is the name for what we've done today

- PCA is the process of linearly transforming data into a new coordinate system such that the greatest variance occurs along the first dimension, the second most along the second dimension, and so on

Variance captured by 1st PC: $197.39^2/100 = 389.63$

Variance captured by 2nd PC: $27.43^2/100 = 7.52$

Variance captured by 3rd PC: $23.26^2 / 100 = 5.41$

width	**length**	**area**	**perimeter**
2.97	1.35	24.78	8.64
-3.03	-0.65	-15.22	-7.36
-4.03	-1.65	-20.22	-11.36
		...	
-3.03	1.35	-11.22	-3.36

PCA in Practice, Summary

Principal Component Analysis

"The purpose of computing is insight, not numbers." - Richard Hamming

So far we've done a lot of numbers. Why is this useful?

Principal Component Analysis for Exploratory Data Analysis

Goal: Plot high dimensional data as a 2 dimensional approximation that results from a linear combinations of attributes

Related Goal: Determine whether this two-dimensional plot is really showing the variability in the data (If not, be wary of conclusions drawn using PCA)

PCA is appropriate for EDA when:

- Visually identifying clusters of similar observations in high dimensions
- You are still exploring the data
  (If you already know what to predict, you probably don't need PCA)
- You have reason to believe that the data are inherently low rank: there are many attributes, but only a few (perhaps unobserved) attributes mostly determine the rest through a linear association

SVD for PCA

Singular value decomposition (SVD) describes a matrix decomposition:

- $X = U\Sigma V^T$ (or $XV = U\Sigma$) where U and V are orthonormal and Σ is diagonal
- If X has rank r, then there will be r non-zero values on the diagonal of Σ
- The values in Σ, called *singular values*, are ordered from greatest to least
- The columns of U are the *left singular vectors*
- The columns of V are the *right singular vectors*

Principal component analysis (PCA) is a specific application of SVD:

- X is a data matrix centered at the mean of each column
- The largest n singular values are kept, for some choice of n;
  All other singular values are set to zero: **the dimensionality reduction**
- The first n **rows of V^T** are **directions** for the n principal components
- The first n **columns of $U\Sigma$** (or XV) contain the n **principal components** of X
- Primarily utilizes $U\Sigma$ (by plotting first two columns and scree plot of Σ)

Inspecting Principal Component Directions (Axes)

Instead of looking at $U\Sigma$, we can instead look at V^T

A principal component direction is a linear combination of attributes

- Given as rows of V^T

Plotting the values of the first principal component direction can provide insight into how attributes are being combined

- High variance attributes are typically included (but not always)
- Many attributes are often included, even if only a few are really important

Interpreting other principal components is challenging; the constraint that they are orthogonal to prior components strongly influences their directions

Summary

- PCA is a technique to summarize data
  - PCA has one goal stated two different ways:
    - Find directions that minimize projection error
    - Find directions that maximize captured variance
  - To conduct PCA, we use SVD (alternatives possible too)
- If PCA is successful, the 2D plot will still preserve structure of original data
- Scree plots tell us how much information lost in PCA

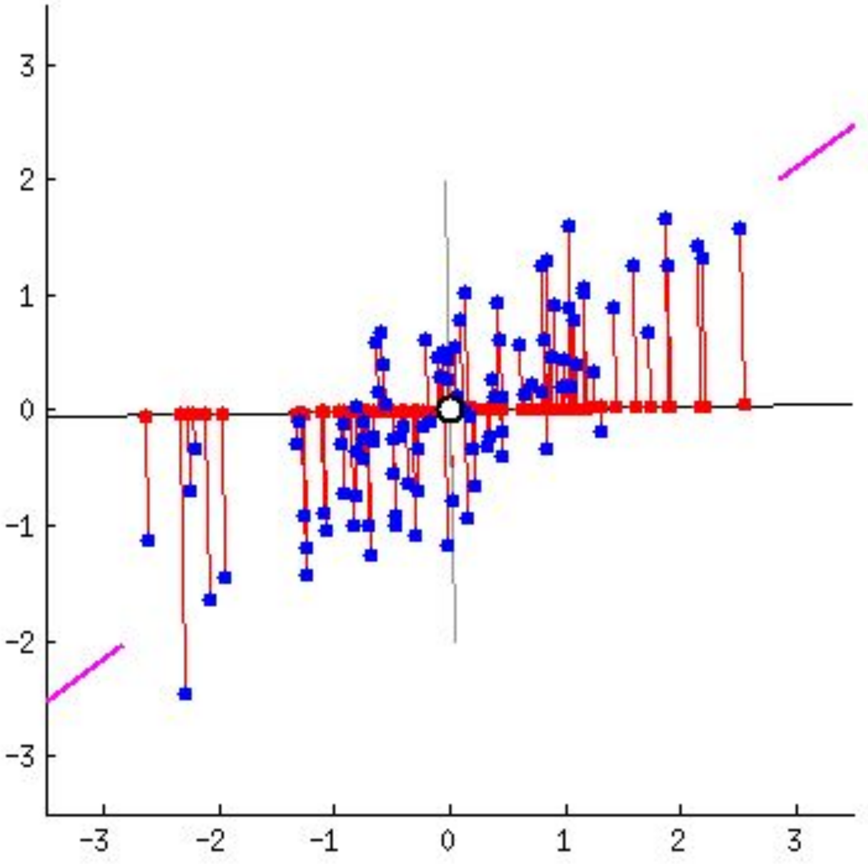
Bonus

As an aside...

Why are these two goals equivalent?

Maximizing variance = spreading out red dots

Minimizing error = making red lines short



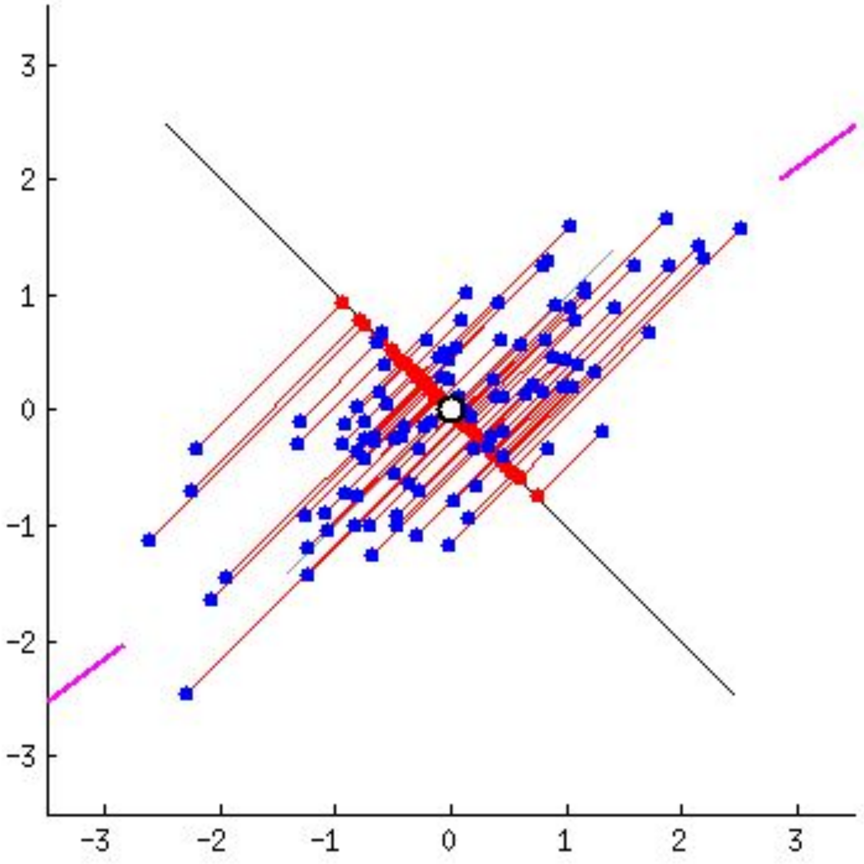
https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues/140579#140579

As an aside...

Imagine that the black line is a stick, and the red lines are springs attached to the stick from the points.

The first PC is where the stick comes to rest.

SVD finds this for us.



https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues/140579#140579

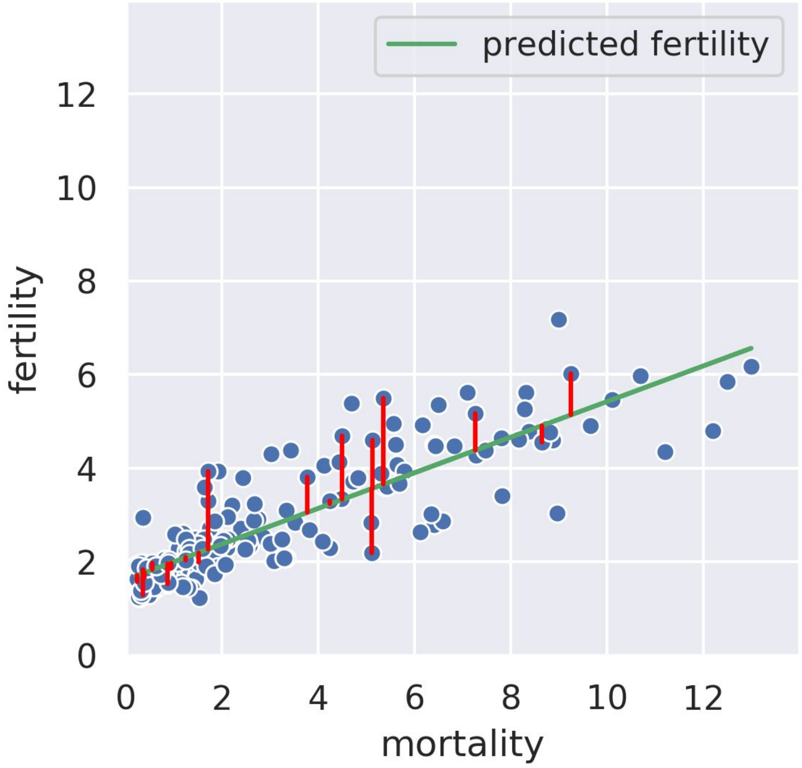
Regression: The Big Idea

Suppose we know the child mortality rate of a given country

- Linear regression tries to predict the fertility rate from the mortality rate
- For example, if the mortality is 6, we might guess the fertility is near 4

The regression line tells us the “best” prediction of fertility given all possible mortality values

- Minimizes the root mean squared error [see vertical red lines, only some shown]

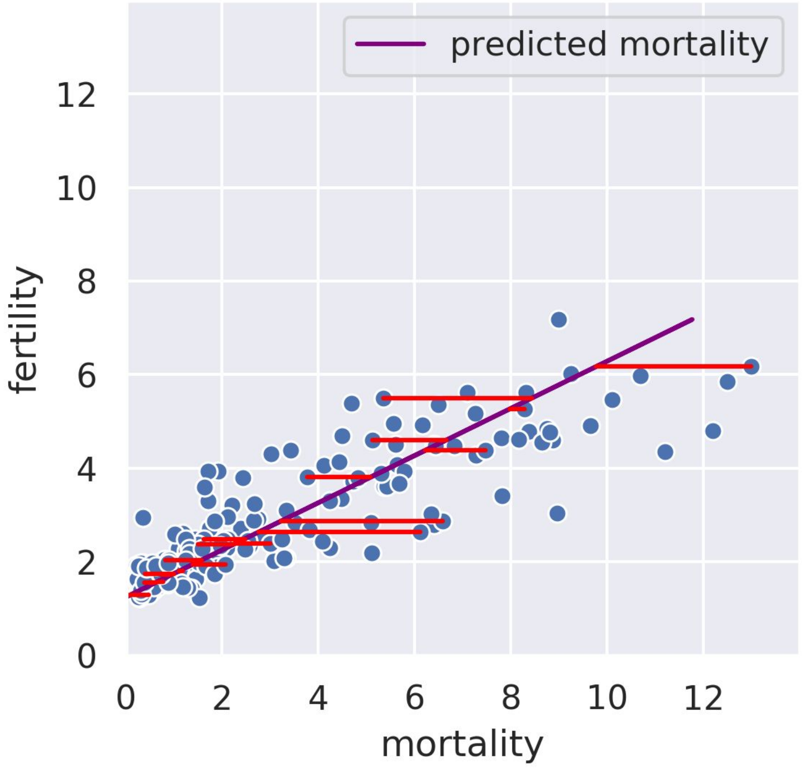


Regression: The Big Idea

We can also perform a regression in the reverse direction.

That is, given the fertility, we try to predict the mortality.

In this case, we get a different regression line which minimizes the root mean squared length of the *horizontal* lines.



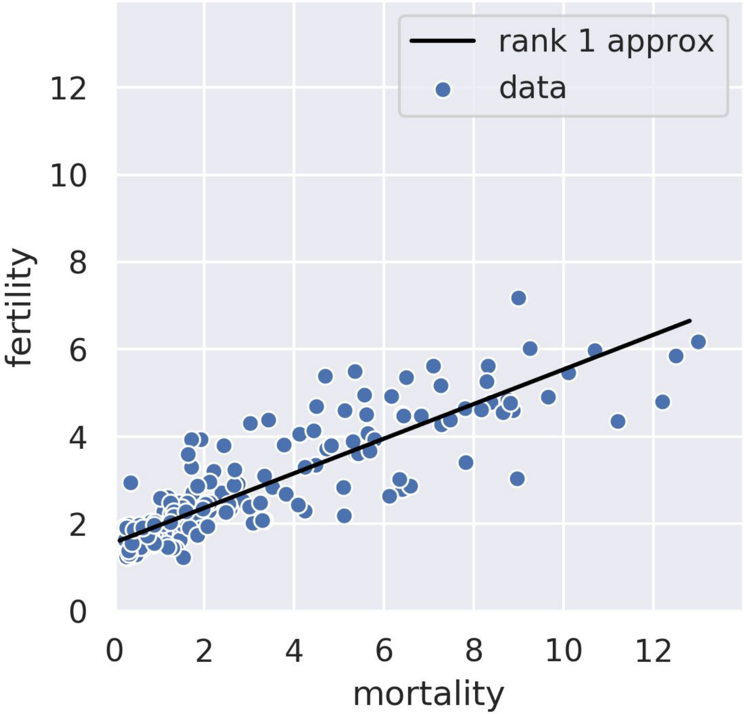
Rank 1 Approximation of Fertility / Mortality Data

Below we see our data and the rank 1 approximation.

Rank 1 Approximation of Fertility / Mortality Data

If we make a line plot instead, this starts to look a lot like a regression line.

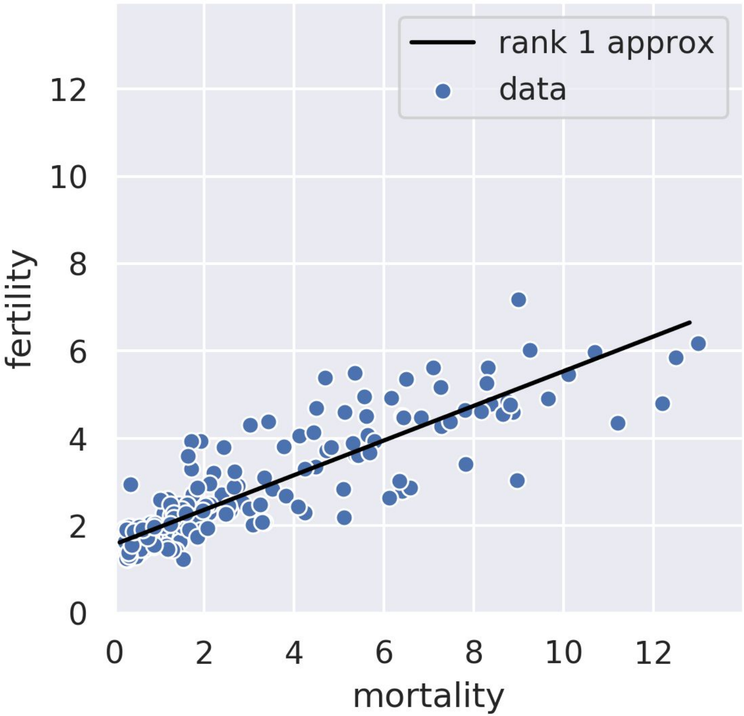
- Note: The approximation is just the data projected onto this line.

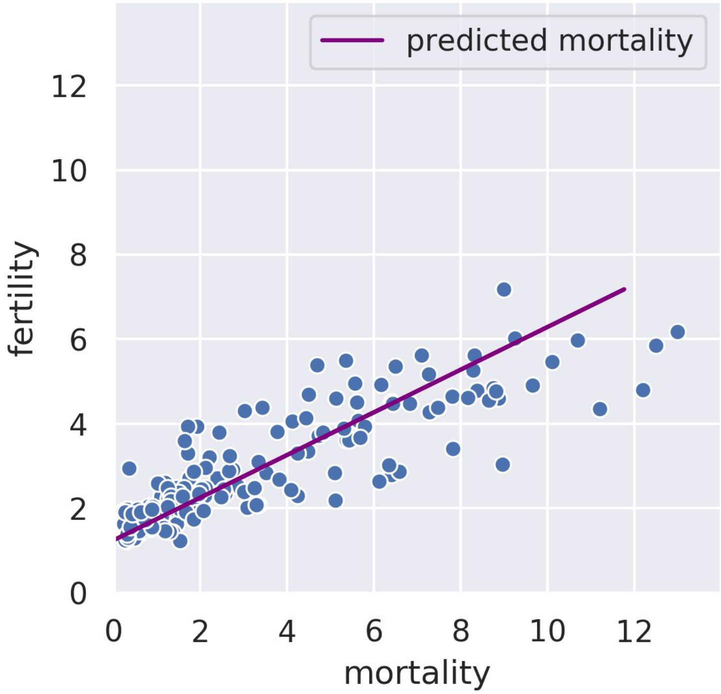


Rank 1 Approximation vs. Predicted Mortality Regression Line

The rank 1 approximation is not the same as the mortality regression line.

- They're vaguely close, but not the same.

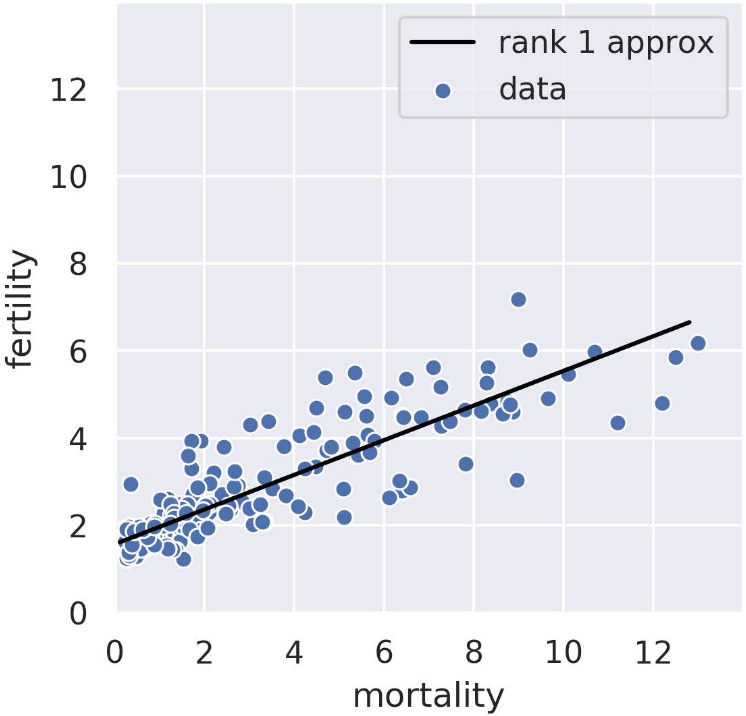


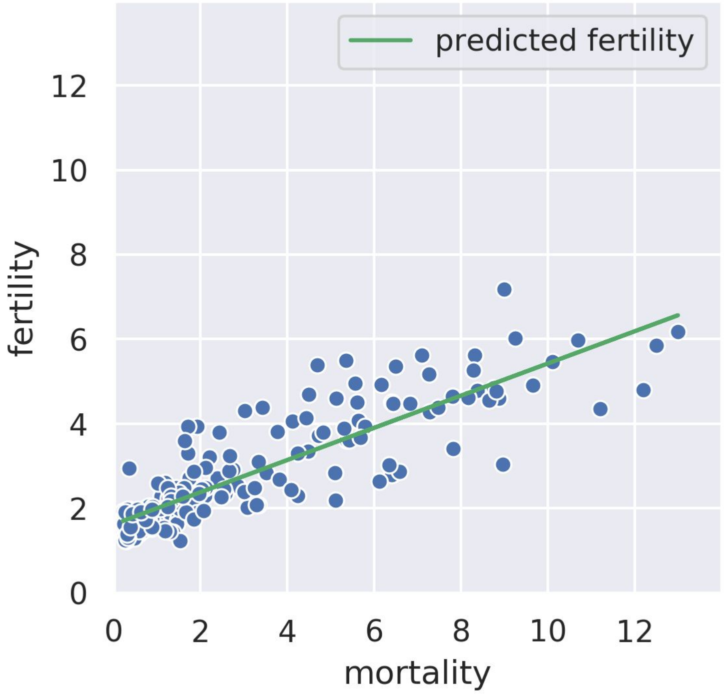


Rank 1 Approximation vs. Predicted Fertility Regression Line

The fertility regression line is pretty close, but is also different.

- The closeness is just a coincidence.

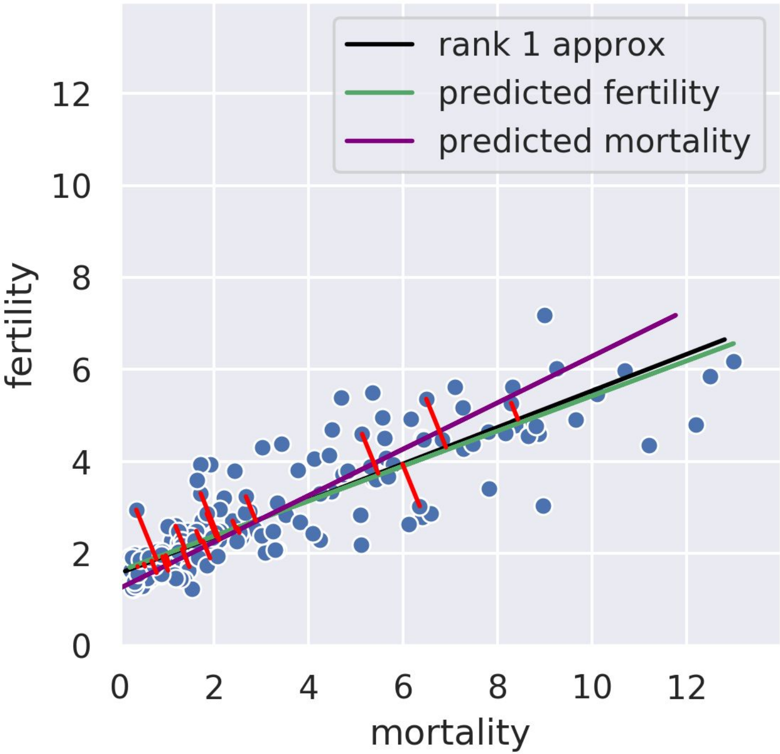




SVD: Minimizing the Perpendicular Error

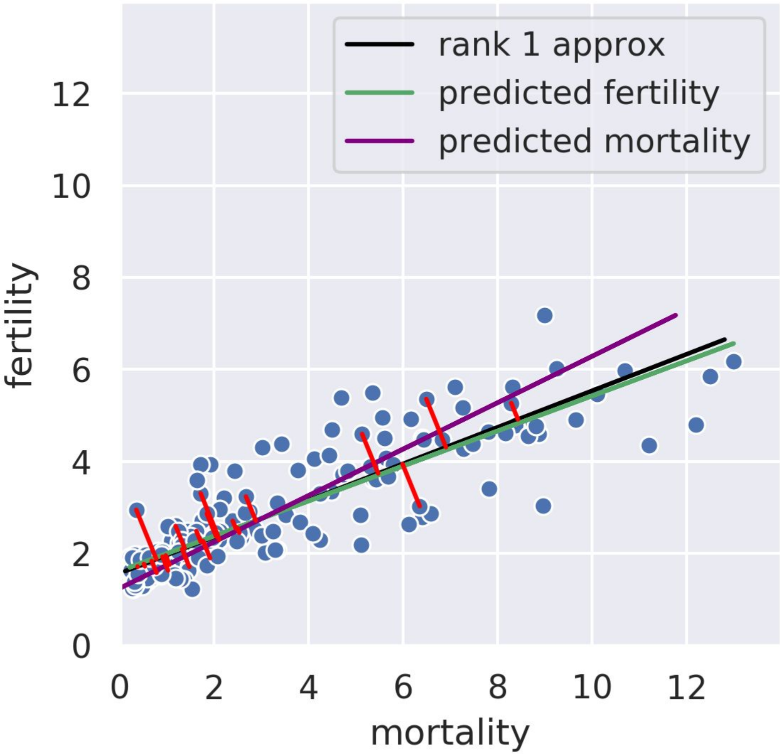
Instead of minimizing *horizontal* or *vertical* error, our rank 1 approximation minimizes the error ***perpendicular*** to the subspace onto which we're projecting

That is, SVD finds the line such that if we project our data onto that line, the error between the projection and our original data is minimized



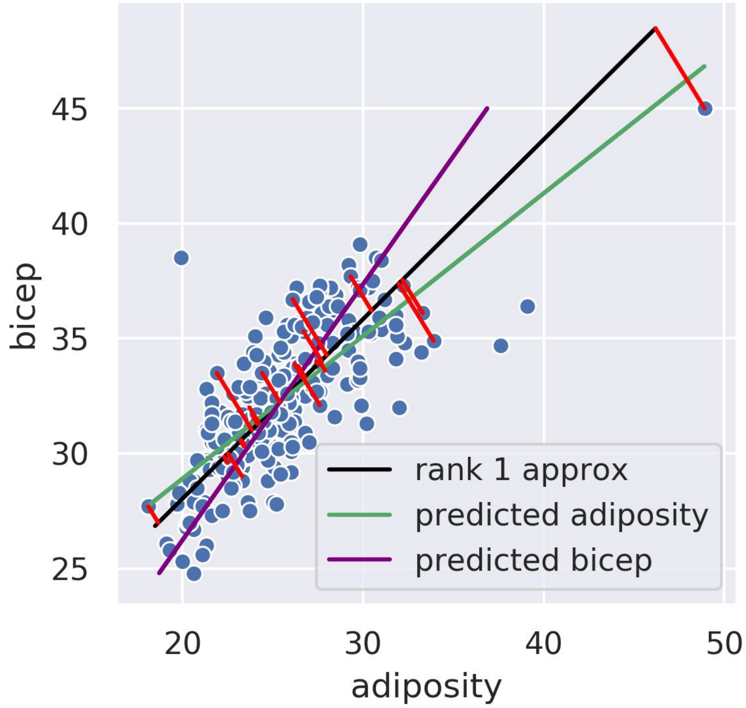
SVD: Minimizing the Perpendicular Error

Reminder: The similarity of the rank 1 approximation and the fertility was just a coincidence.



SVD: Minimizing the Perpendicular Error

Looking at adiposity and bicep size from our body measurements dataset, we see the 1D subspace onto which we are projecting is between the two regression lines.



Beyond 1D/2D

In higher dimensions the idea behind principal components is just the same!

Example: Suppose we have 30 dimensional data and decide to use the first 5 principal components.

- Our procedure minimizes the error between:
  - The original 30 dimensional data.
  - The projection of that 30 dimensional data on to the "best" 5 dimensional subspace. See CS 189 or EECS 127 for more details. The relevant CS 189 reading is available here.