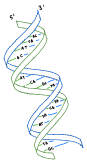
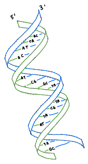
DATA CREATED: 03-13-2023
LAST MODIFIED: 07-19-2023
AUTHOR: KATYA KHVATKOVA



SINGULAR VALUE DECOMPOSITIONS, EIGENVALUE DECOMPOSITIONS, AND THEIR RELATIONSHIP TO PCA



SUMMARY:

Across different branches of -omic studies, principal component analysis is a common technique used in both quality control and exploratory data analysis to study features of -omic data relative to samples and phenotypes. Because -omic datasets are large and often substantially differ in row and column dimensions, interesting mathematics must be exploited by PCA algorithms. Further, many popular PCA algorithms produce different outputs for estimates of loadings, PC scores, and eigenvalues.

Here, we provide some mathematical framework that connects the concepts of PCA and SVD. This document, while primarily a mathematical text, caters to individuals with a math or a stats background at an advanced undergraduate to graduate level.

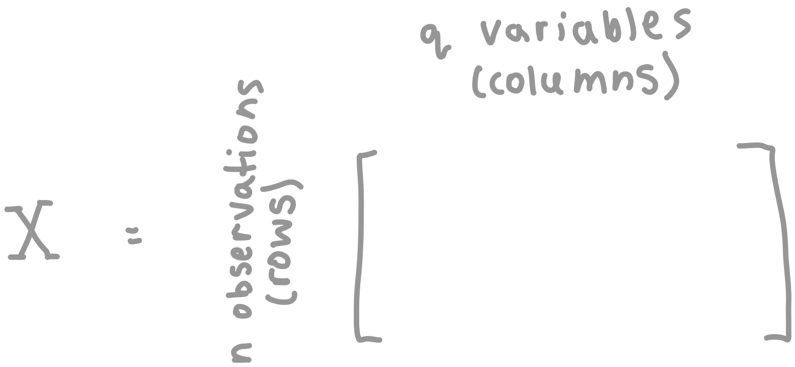
ACKNOWLEDGEMENTS:

These notes were taken for a task within a research project I worked on under advisement of Carl Langefeld and Hannah Ainsworth at Wake Forest School of Medicine. John Gemmer, a mathematician at Wake Forest University, helped me with this document through sharing background knowledge and notes with me on this topic. Throughout this text, I acknowledge several additional online sources I used as inspiration, particularly for graphics and visuals, to illustrate key concepts.

Section 1: Some Housekeeping, Introductions, and a Brief Overview of What We Will Learn in These Notes

#1. Principal component analysis (PCA) is centralized around a special case of a singular value decomposition (SVD) of a matrix X_{nxq}. Simply put, an SVD of a matrix is just a way to write a matrix in terms of several other matrices. One reason that we are interested in finding the SVD of a matrix is that an SVD can reveal qualitative information about this matrix. SVD's are also important in the field of numerical linear algebra, as they can be exploited to change the computational complexity of certain algorithms.

#2. Suppose you have a matrix full of data - we will call this matrix X.



We can observe the following three cases for dimensions of X:

- $n > q$ (nrows > ncols)
- $q > n$ (ncols > nrows)
- $n = q$ (nrows = ncols)

I use ">>" here to mean "much larger than"

In -omic datasets (and in big data in general) it is common to work in cases where $n >> q$ or $q >> n$

(in other words, the dimensions greately differ).

Further,

In PCA, our data matrix X is viewed as a real-valued matrix instead of random variables. Here I will note that we are using $X_{n \times q}$ (the stats matrix notation convention) instead of $A_{n \times m}$ (the math matrix notation convention). This means that when we talk about independence and dependence of the rows and columns of $X_{n \times q}$ throughout this document, we are referring to linear independence and linear dependence (not probablistic independence) dependence).

#3. In PCA, the eigenvalues and eigenvectors of X are of keen interest to us. Moreover, eigenvalues and eigenvectors are essential to the analysis of linear transformations. Coming from the German word "eigen" meaning "characteristic", eigenvalues and eigenvector (once we extract them using some mathematical or computational technique) tell us things about the "characteristics" of the matrix.

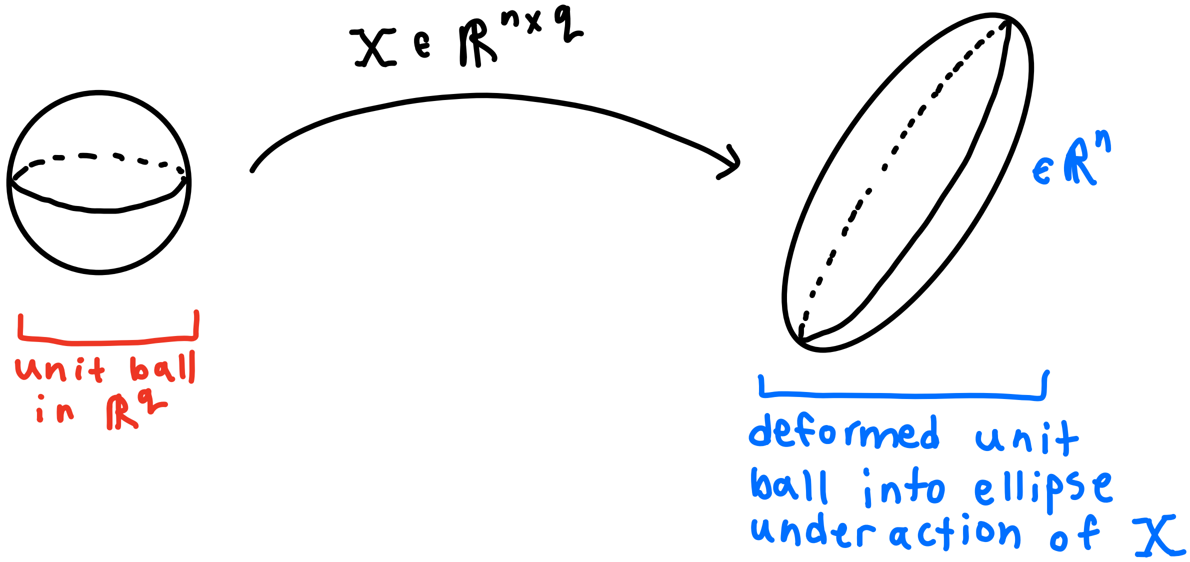
#4. When you use eigenvalue calculators (or various functions to do PCA like 'prcomp( )' and 'PCA( )' in R), you might not always get exactly the same answers or approximations. While we won't focus directly on the algorithms and processes used to generate eigenvalues or eigenvectors, it is important to note that not all matrices work well or are compatible with every algorithm.

For most matrices (especially non-square ones), it is hard/computationally complex to find eigenvalues and eigenvectors, even with a great set of algorithms. If you are interested, there is a concept in numerical linear algebra called the condition number which is the ratio of the largest to smallest eigenvalue. when the smallest eigenvalue of a matrix is equal to zero, the condition number is infinity ∴ standard eigenvalue solvers are ill-conditioned and unreliable. Iterative methods to approximate these values are fantastic these days, but can still fail when you have one or many zero or nearly zero eigenvalues.

Section 2: Singular Value Decomposition (SVD)

We will introduce this fact about the geometric interpretation of a matrix and let it sit for a minute. as it might be new to some folks.

$\boxed{\text{FACT}}$ Any $n \times q$ matrix X transforms the unit circle in $\mathbb{R}^q$ into an ellipsoid in $\mathbb{R}^n$


$X \in \mathbb{R}^{n \times q}$
$\in \mathbb{R}^n$
unit ball in $\mathbb{R}^q$
deformed unit ball into ellipse under action of X

This is one way to conceptualize what a matrix is.

To further illuminate what this means in an algebraic way (rather than geometric), we will be considering the following three cases in these notes:

- ★ case 1: when $n > q$
- ★ case 2: when $q > n$
- ★ case 3: when $n = q$

But first, we will review some linear algebra

Section 2.1: Linear Algebra Review

A loose definition of linear dependence: If you can express two columns (or rows) j and k in a matrix X like so:

$$\underbrace{\begin{bmatrix} x_{1,j} \\ \vdots \\ x_{n,j} \end{bmatrix}}_{j^{th}\text{ column}} = \underset{\text{scalar}}{\underset{\uparrow}{c}} \underbrace{\begin{bmatrix} x_{1,k} \\ \vdots \\ x_{n,k} \end{bmatrix}}_{k^{th}\text{ column}}$$

$\Rightarrow$ j^{th} column is lin. dep. with k^{th} column

Now recall the concept of rank:

$$\begin{aligned} \text{rank}(X) &= \dim\left(\text{span}\{\text{columns of } X\}\right) \\ &= \dim\left(\text{span}\{\text{rows of } X\}\right) \end{aligned}$$

where the dimension of the span of the columns is the maximal number of linearly independent columns of X

THM I will not prove this here, but you can find proofs of this in books or online:

$$\text{column rank}(X) = \text{row rank}(X)$$

Thinking back to linear algebra, remember performing row reduction on a matrix where $n \neq q$ and/or that had linearly dependent rows or columns and you end up getting zero vectors when in reduced row echelon form... this is a time where you may or may not have discussed this idea. For our purposes, just hold onto this idea.

DEF An $n \times q$ matrix X is said to have full rank if its rank equals the largest possible rank for a matrix of $n \times q$.

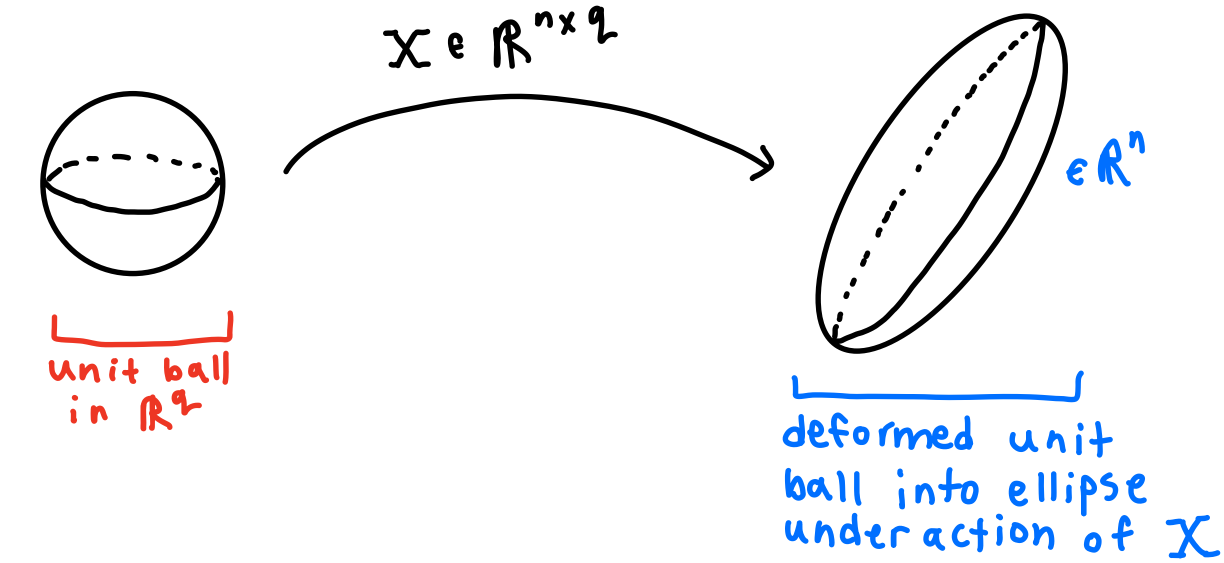
Recall that $\text{rank}(X) \leq \min(n,q)$ (this is true for any matrix $X_{n\times q}$).

$\Rightarrow$ A matrix X is full rank when

$$\text{rank}(X) = \min(n,q) \quad (*)$$

DEF A $n \times q$ matrix X is said to be rank deficient if $\text{rank}(X) < \min(n,q)$

Section 2.2 : Back to SVD


$X \in \mathbb{R}^{n \times q}$
unit ball in $\mathbb{R}^q$
$\in \mathbb{R}^n$
deformed unit ball into ellipse under action of X

TERMINOLOGY / DEFINITIONS

$\boxed{\text{DEF}}$ The principal axes of the ellipsoid are denoted by $\vec{u}_i$ and can be ordered by their lengths which are denoted by θ_i s.t. $\underline{\text{for } i = 1, \dots, n \text{ , } \theta_{i+1} \geq \theta_i}$. In total, there are n principal axes.

In other words, we label each principal axis by their ordered lengths

$\boxed{\text{DEF}}$ The preimages of each $\vec{u}_i$ are denoted by $\vec{v}_i$. In total, there are q preimages.

THEORUM:

I will not prove this here, but it has been shown.

$\boxed{\text{THM}}$ The principal axes $\{\vec{u}_1, \dots, \vec{u}_n\}$ and the preimages $\{\vec{v}_1, \dots, \vec{v}_q\}$ are orthonormal.

Length = 1 (by definition), and they all meet at right angles

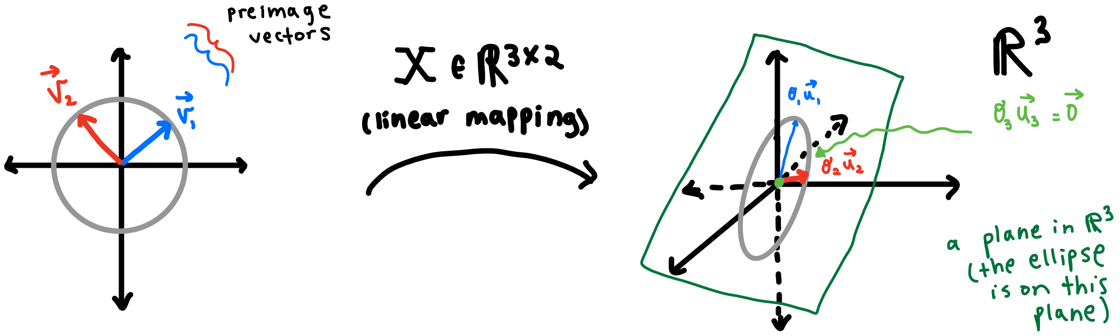
TO UNDERSTAND WHAT THIS MEANS, WE WILL FIRST GEOMETRICALLY SHOW THE THREE POSSIBILITIES:

For the sake of this illustration, we are going to assume all of the rows & columns of X are linearly independent

CASE 1: when $n > q$

EX: $X = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ x_{3,1} & x_{3,2} \end{bmatrix} \in \mathbb{R}^{3\times 2}$

statement: X maps the unit circle in $\mathbb{R}^2$ to an ellipse in $\mathbb{R}^3$



preimage vectors
$\vec{v}_2$
$\vec{v}_1$
$X \in \mathbb{R}^{3\times 2}$
(linear mapping)
$\mathbb{R}^3$
$\theta_1\vec{u}_1$
$\theta_3\vec{u}_3 = \vec{0}$
$\theta_2\vec{u}_2$
a plane in $\mathbb{R}^3$ (the ellipse is on this plane)

UNIT CIRCLE in $\mathbb{R}^2$

AN ELLIPSE THAT IS ONLY 2D THAT EXISTS IN $\mathbb{R}^3$

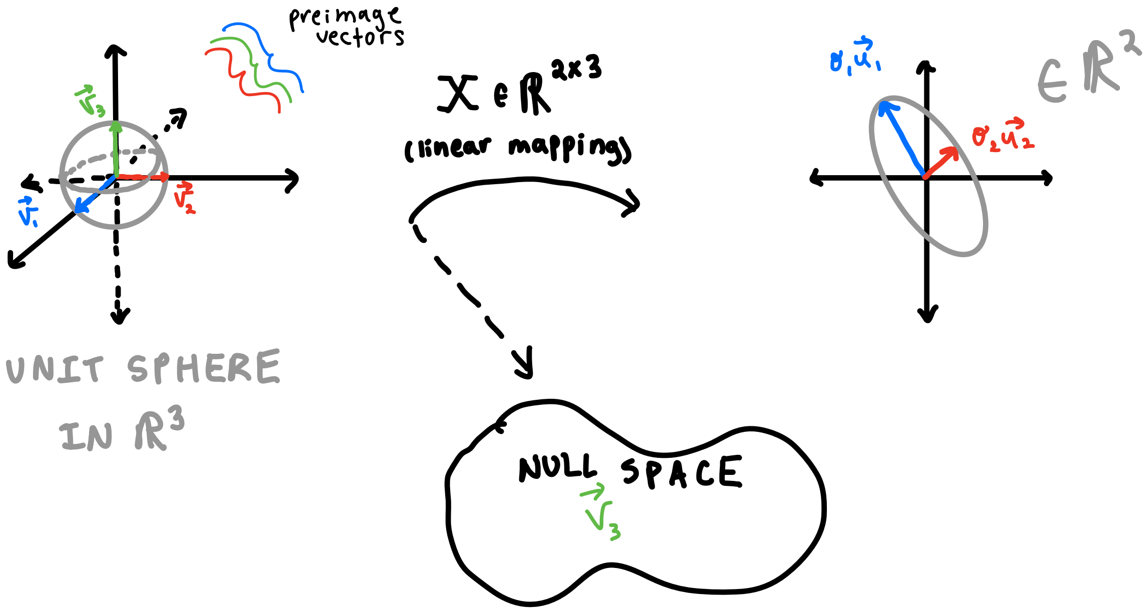
In case 1, there is a third principal axis $\vec{u}_3$ (in green) that is orthormal to the other 2 vectors $\vec{u}_1$ and $\vec{u}_2$. However θ_3 (constant) $= 0$!

Therefore, the third principal axis u_3 is stiffled by $\theta_3 = 0$ into the zero vector

CASE 2: when $q>n$

EX: $X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{bmatrix} \in \mathbb{R}^{2\times 3}$

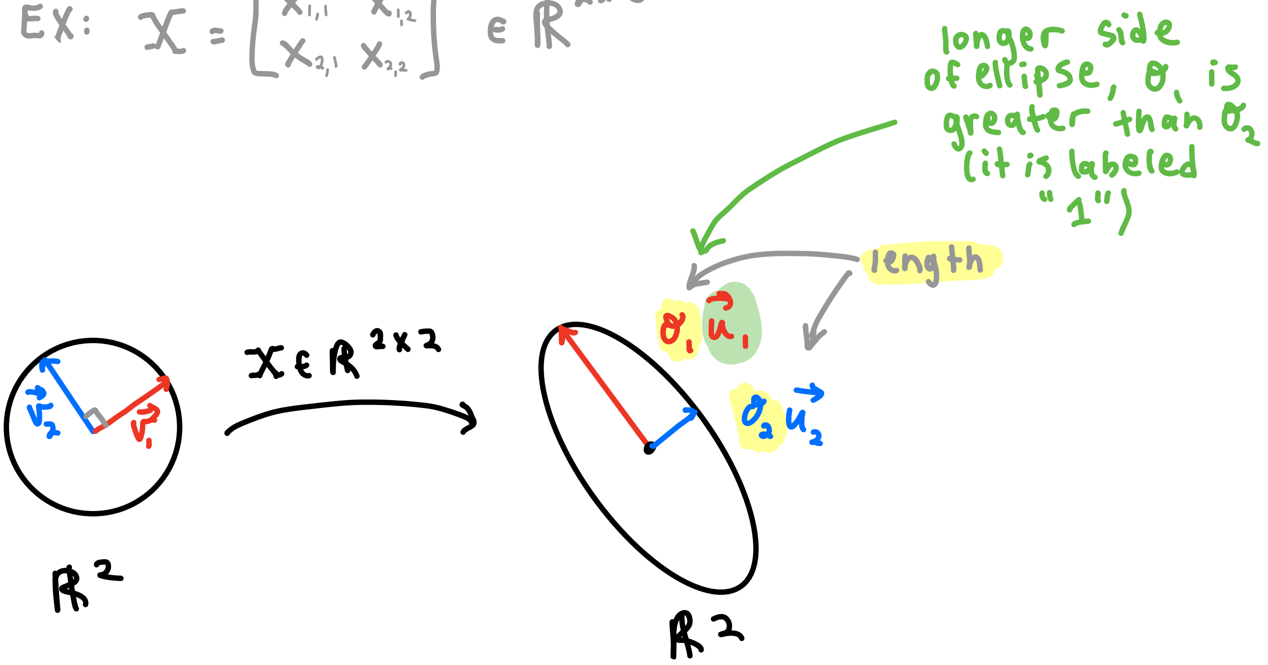
Statement: X maps the unit sphere in $\mathbb{R}^3$ to an ellipse in $\mathbb{R}^2$



In this case where we are mapping from a higher to a lower dimension, our vector $\vec{V}_3$ maps to the null space – and our matrix X takes us from 3D to 2D.

CASE 3: when n=q

EX: $\mathcal{X} = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{bmatrix} \in \mathbb{R}^{2\times 2}$



longer side of ellipse, σ_1 is greater than σ_2 (it is labeled "1")

This case is more straightforward (again... for this illustration, we assumed all of our vectors in $\mathcal{X}$ were linearly independent. Dependence just tacks on additional vector mapping to null space in ANY case)

Note: There are many ways to understand what a matrix is... this idea is one of them (a mapping of unit ball in $\mathbb{R}^q$ to ellipsoid in $\mathbb{R}^n$). This way is the key way to understanding the math of SVDs

Section 2.3: The Core Idea of SVD

We're going to pose some definitions for matrices.

Let $r = \text{rank}(X)$ where $X \in \mathbb{R}^{n \times q}$.

By stacking the vectors v_i and u_i into the colums of matrices $\hat{V}$ and $\hat{U}$:

It holds that

$$\underbrace{X}_{(n \times q)} \underbrace{\hat{V}}_{(q \times r)} = \underbrace{\hat{U}}_{(n \times r)} \underbrace{\hat{\Sigma}}_{(r \times r)} \qquad (*)$$

We call $(*)$ the reduced singular value decomposition of X.

NOW WE WILL DEFINE THE FOLLOWING CONCEPT:

$\boxed{\text{DEF}}$ The (real)* matrices U and V are said to be unitary if

$$\begin{cases} U^{-1} = U^T \\ V^{-1} = V^T \end{cases}$$

In other words $V^T V = V V^T = I$

*The definition of unitary is slightly different when we have complex #'s involved; but in our statistical works, this doesn't matter!

KEY IDEA:

The "full" SVD is obtained by "completing" $\hat{U}$ and $\hat{V}$ into unitary matrices U and V and padding Σ with 0's if necessary:

$$\underbrace{X}_{n\times q}\,\underbrace{V}_{q\times q} = \underbrace{U}_{n\times n}\,\underbrace{\Sigma}_{n\times q} \qquad (**)$$

To understand what we mean by this, we need to consider the three cases above AND we need to understand 2 ideas of what's going on:

① some of the vectors in V, Σ, and U must be turned into zero vectors (depending on the case and rank)

② Algorithms (like the Graham-schmidt process*) must be used to obtain vectors orthonormal to the other vectors inside $\hat{V}$ and $\hat{U}$ (or V and U) respectively (depending on the case and rank)

we will illustrate this in the next section.

Algebraically (and computationally) this form of the SVD is more appealing because U and V are square!

* Note: If you are not familiar with Graham-schmidt processes, it is basically just a way to produce vectors orthogonal to each other.

Section 2.4 : The Core Idea of SVD (Expanded)

Recall from the diagrams, regardless of of X's dimensions, we always have q preimages and n principal axes...

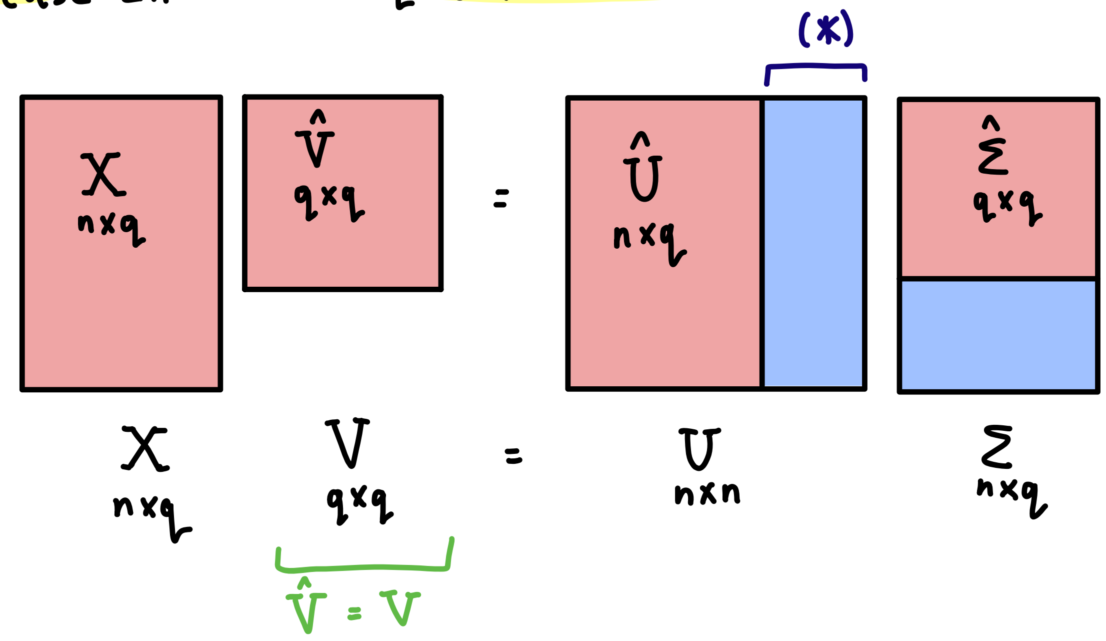
Most SVD algorithms focus on obtaining V, and then, moreover, the obtain $\vec{u}_i$ through

$$X\vec{v}_i = \theta_i \vec{u}_i \Rightarrow \vec{u}_i = \frac{X\vec{v}_i}{\theta_i}$$

ACKNOWLEDGEMENTS: These matrix sketches below are inspired by the article "How are principal component analysis and singular value decomposition related?" by Andre Perunicic at the following link: https://intoli.com/blog/pca-and-svd/

CASE 1: When $n > q$

case 1A: with $r = q$ (full rank)



First, notice that $\hat{V} = V$ when $r = q$ (ie, there are no linearly dependent columns) second, notice that this, right away means that if our algorithm can find us $\hat{U}$, it can find us U!

Third, look at (*)

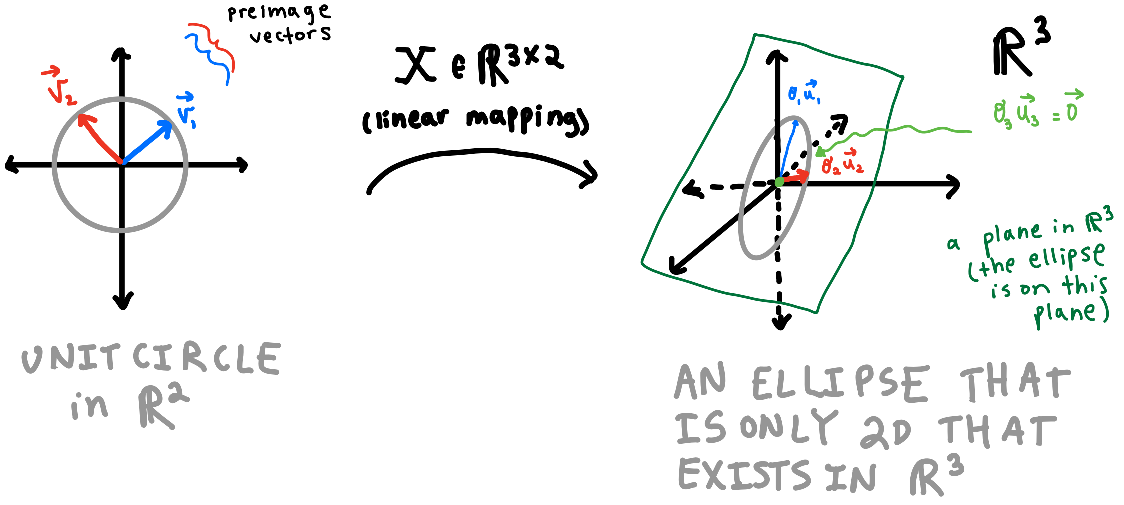
This blue rectangle in the complete matrix U contains additional vectors that are orthogonal to the vectors in $\hat{U}$. These vectors can be obtained through a method like the Graham-Schmidt process

★ For computational purposes, sometimes algorithms require that these $\vec{u}_i$ vectors in U and not $\hat{U}$ must be obtained (U is desirable because it is square!)

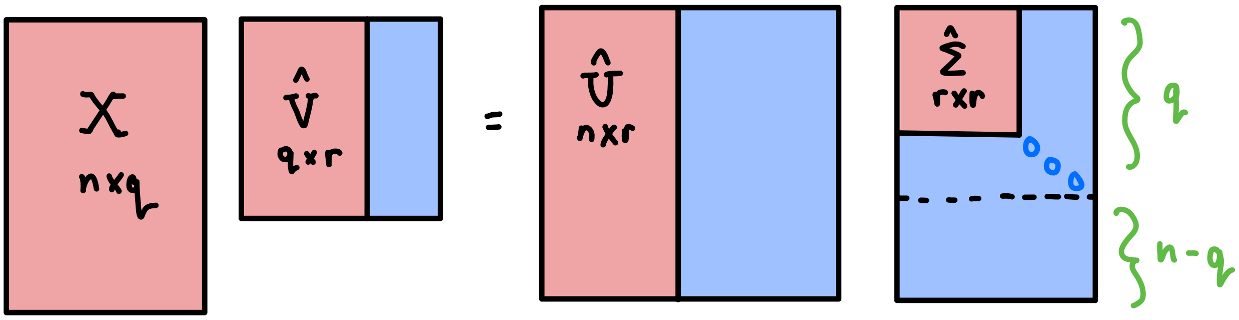
★ Generally, behind the scenes, we might care about all of U

VISUALIZATION : $X \in \mathbb{R}^{3 \times 2}$

Statement: X maps the unit circle in $\mathbb{R}^2$ to an ellipse in $\mathbb{R}^3$


preimage vectors
$\vec{v}_2$
$\vec{v}_1$
UNIT CIRCLE in $\mathbb{R}^2$
$X \in \mathbb{R}^{3 \times 2}$
(linear mapping)
$\mathbb{R}^3$
$\theta_1 \vec{u}_1$
$\theta_2 \vec{u}_2$
$\theta_3 \vec{u}_3 = \vec{0}$
a plane in $\mathbb{R}^3$ (the ellipse is on this plane)
AN ELLIPSE THAT IS ONLY 2D THAT EXISTS IN $\mathbb{R}^3$

case 1B: with $r<q$ (rank deficient)



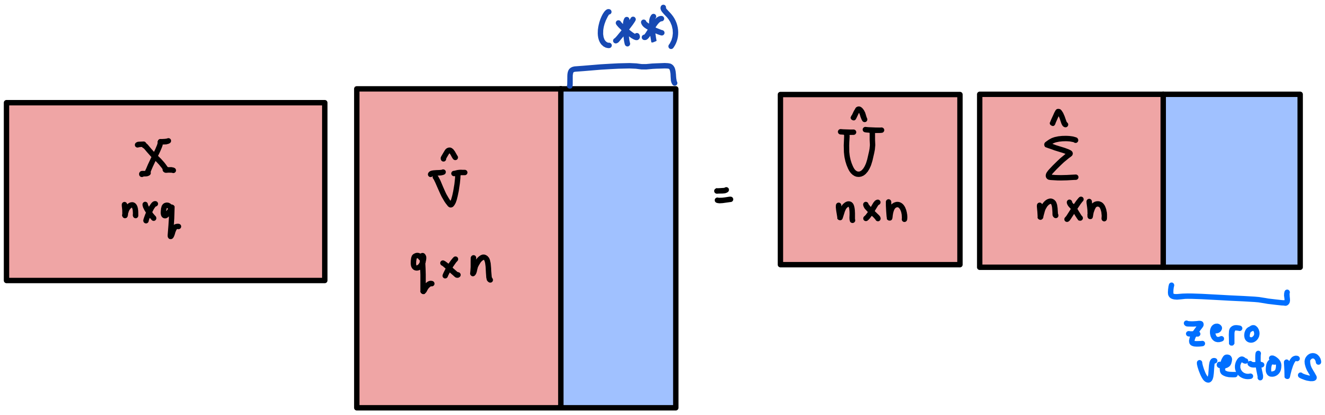
$$X_{n\times q} \quad V_{q\times q} \quad = \quad U_{n\times n} \quad \Sigma_{n\times q}$$

In the $\hat{\Sigma}$ matrix reduces to $r\times r$ (from $q\times q$)... this means we have $q-r$ σ_i's that are $=0$.

when $r<q$, in $\hat{V}$, we don't have several columns due to linear dependency.

CASE 2 : when $q>n$

case 2A: with $r=n$ (full rank)



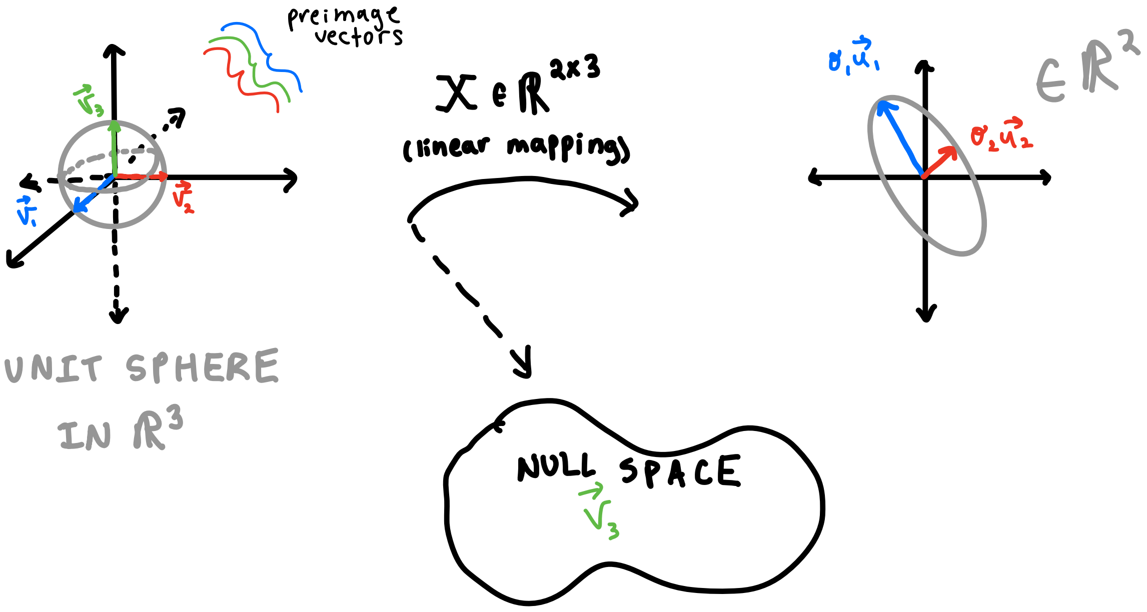
$$\underset{n\times q}{X}\qquad \underset{q\times q}{V} \;=\; \underbrace{\underset{n\times n}{U}}_{\hat{U}=U}\qquad \underset{n\times q}{\Sigma}$$

First, notice that $\hat{U}=U$ when $r=n$ (ie. there are no linearly dependent columns). HOWEVER!!! If an algorithm must start with a complete V and not a complete U (ie, it can't find vectors in (**) that are orthonormal to vectors in $\hat{V}$), then it will fail.

vectors in (**) are not of interest to statisticians, but they are computationally necessary at times.

VISUALIZATION : $X \in \mathbb{R}^{2\times3}$

Statement: X maps the unit sphere in $\mathbb{R}^3$ to an ellipse in $\mathbb{R}^2$

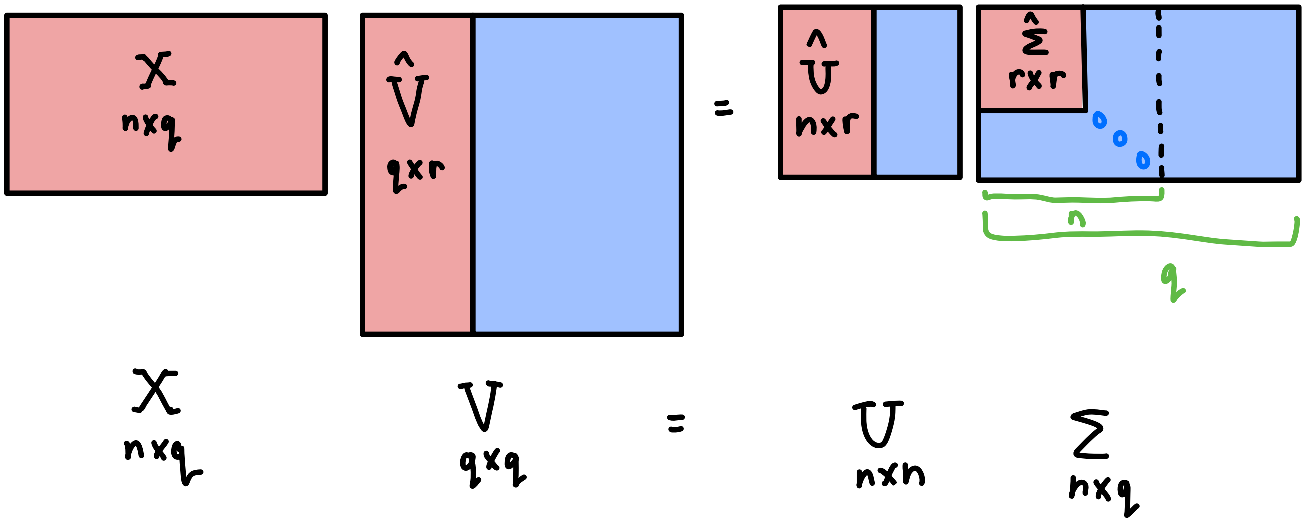


Looking at the blue rectangle in V (corresponding to the columns in V), we can see how some columns are forced into the null space. Note here that vectors in the null space mapped to the zero vector (a lack of that dimension), they are not equal to the zero vector after being multiplied by some constant.

Like in case 1, those $\vec{v}_i$'s are orthonormal to the other $\vec{v}_i$'s outside of the nullspace after this linear transformation - we are simply not interested in them (although they might be used in algorithms and behind the scenes of our work).

we can also see this in the $\Sigma_{n\times q}$ matrix. In case 2, it is inevitable that some of these columns are $\vec{0}$'s.

case 2B: with $r < n$ (rank deficient)



Again, we drop additional rows in V because of rank deficiency. And that $\Sigma_{n\times q}$ will have $n-r$ zero entries along the diagonal!

CASE 3 : When $n=q$

case 3A: with $r=(n=q)$ (full rank)

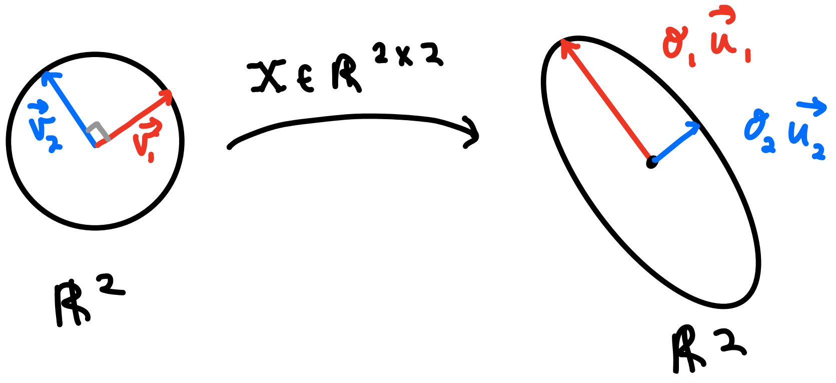
$$\boxed{\underset{\substack{(n\times q)=\\(q\times q)=\\(n\times n)}}{X}}\ \boxed{\underset{q\times q}{\hat{V}}} = \boxed{\underset{n\times n}{\hat{U}}}\ \boxed{\underset{\substack{(n\times n)=\\(q\times q)}}{\hat{\Sigma}}}$$

$$\underset{n\times q}{X}\quad \underbrace{\underset{q\times q}{V}}_{} = \underbrace{\underset{n\times n}{U}}_{}\quad \underset{n\times q}{\Sigma}$$

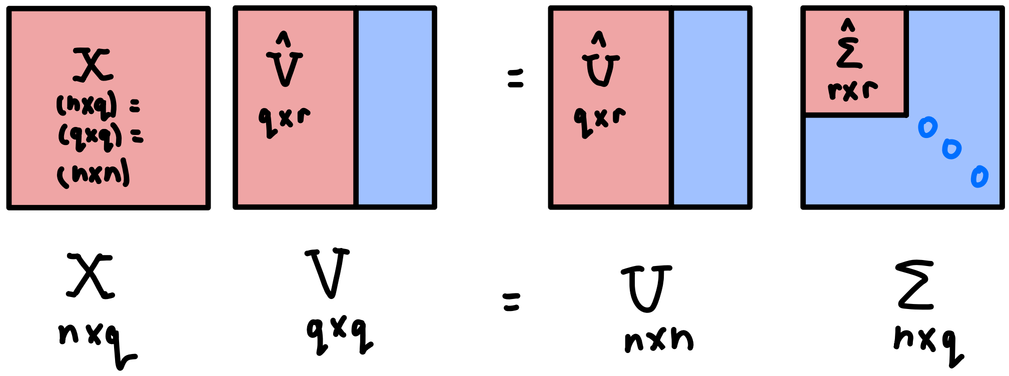
Notice in this case that $U=\hat{U}$ AND $V=\hat{V}$

VISUALIZATION : $X \in \mathbb{R}^{2\times 2}$

Statement: X maps the unit sphere in $\mathbb{R}^2$ to an ellipse in $\mathbb{R}^2$



case 3B: with $r<(n=q)$ (rank deficient)



section 2.4: Eigenvalue Decomposition of X^TX

In a moment, you will understand why we are interested in finding the eigenvalues and eigenvectors of X^TX. For now, consider the following:

Because every nxq matrix X has an SVD,

$$XV = U\Sigma \quad \Rightarrow X = U\Sigma V^{-1}$$

and because V is unitary

$$\Rightarrow X = U\Sigma V^T$$

Which implies that $X^T = (U\Sigma V^T)^T = V\Sigma^T U^T$.

Then,

$$\Rightarrow XX^T = U\Sigma\underbrace{V^T}_{}V\Sigma^TU^T \quad \&$$

$$X^TX = V\Sigma^T\underbrace{U^T}_{}U\Sigma V^T$$

$$\Rightarrow XX^T = U\Sigma V^{-1}V\Sigma^TU^T \quad \&$$

$$X^TX = V\Sigma^TU^{-1}U\Sigma V^T$$

$$\Rightarrow XX^T = U\Sigma\Sigma^TU^T$$

$$X^TX = V\Sigma^T\Sigma V^T$$

$$\Rightarrow XX^TU = U\Sigma\Sigma^T\underbrace{U^TU}_{} = U^{-1}U = I$$

$$X^TXV = V\Sigma^T\Sigma\underbrace{V^TV}_{} = V^{-1}V = I$$

$$\Rightarrow XX^TU = U\Sigma\Sigma^T$$

$$X^TXV = V\Sigma^T\Sigma$$

Therefore

$$(***)\begin{cases} \underset{(n\times q)}{X}\,\underset{(q\times n)}{X^T} = \underset{(n\times n)}{U}\,\underset{(n\times q)}{\Sigma}\,\underset{(q\times n)}{\Sigma^T}\,\underset{(n\times n)}{U^T} \\ \underset{(q\times n)}{X^T}\,\underset{(n\times q)}{X} = \underset{(q\times q)}{V}\,\underset{(q\times n)}{\Sigma^T}\,\underset{(n\times q)}{\Sigma}\,\underset{(q\times q)}{V^T} \end{cases}$$

And more over, because $\Sigma\Sigma^T$ and $\Sigma^T\Sigma$ are diagonal (and commute w/ U & V respectively)

$$XX^TU = U\Sigma\Sigma^T$$
$$X^TXV = V\Sigma^T\Sigma$$

$\Rightarrow$ This is a property of diagonal matrices, like $AB = BA$

$$\left.\begin{aligned} XX^TU &= \Sigma\Sigma^TU \\ X^TXV &= \Sigma^T\Sigma V \end{aligned}\right]$$

like $A\gamma_i = \lambda_i\gamma_i$, but in matrix form

And we have that

① For $i=1,\dots,n$, $\vec{u}_i$ are eigenvectors of $XX^T \in \mathbb{R}^{n\times n}$ with eigenvalues $\theta_1^2, \dots, \theta_n^2$

② For $i=1,\dots,q$, $\vec{v}_i$ are eigenvectors of $X^TX \in \mathbb{R}^{q\times q}$ with eigenvalues $\theta_1^2, \dots, \theta_q^2$

Consequently, algorithms that can obtain the eigenvalues (or estimates of the eigenvalues) of X^TX will yield the singular values (or the nonnegative square roots of the eigenvalues) as well as the vectors $\vec{v}_i$.

Moreover, we can obtain $\vec{u}_i$ through

$$X\vec{v}_i = \theta_i\vec{u}_i \quad \Rightarrow \vec{u}_i = \frac{X\vec{v}_i}{\theta_i}$$

HOWEVER!!!

If $q \geq n$,

$$\text{rank}(X) \leq \min(n, q)$$

$$\Rightarrow \text{rank}(X) \leq n$$

This can be proved - I will not prove this here but it is true for $X \in \mathbb{R}^{n \times q}$ (not sure about complex-valued matrices)

And because $\text{nullspace}(X^T X) \subseteq \text{nullspace}(X)$

$$\Rightarrow \text{rank}(X^T X) \leq n$$

But $X^T X \in \mathbb{R}^{q \times q}$ $\therefore$ it MUST have eigenvalues equal to 0 (specifically, at least $q-n$ of them).

WHICH MEANS THAT IT IS ILL-CONDITIONED FOR STANDARD EIGENVALUE ALGORITHMS (SEE POINT #4 IN INTRO)

We will illustrate in the next section how this relates to PCA ...

section 3: How SVD connects to PCA

PCA is a special case of SVD when the columns are centered about their means.

Let X be any $n \times q$ matrix of data. It's $q \times q$ covariance matrix is given by

$$\text{cov}(X) \overset{\text{def}}{=} \mathbb{E}\left[(X - \mathbb{E}(X))^T (X - \mathbb{E}(X)) \right]$$

And the sample covariance is given by

$$S_X \overset{\text{def}}{=} \frac{1}{n-1} (X - \bar{X})^T (X - \bar{X})$$

If our data is centered (moreover, if it is standardized)

$$\mathbb{E}(\underset{\underset{(n\times q)}{\uparrow}}{X}) = \begin{array}{cc} & q \\ n & \begin{bmatrix} \vec{0} & \cdots & \vec{0} \end{bmatrix} \end{array}$$

★ ie, for $j = 1, \dots, q$ we fix $\frac{1}{n} \sum_{i=1}^{n} x_{i,j} \overset{\text{set}}{=} 0$

↑ this is a column mean

∴ The sample covariance matrix of a centered matrix X_c is given by

$$S_{X_c} = \frac{X_c^T X_c}{n-1}$$

By (***), we know that if the SVD of $X_c = U\Sigma V^T$, then

$$S_{X_c} = \frac{X_c^T X_c}{n-1} = \frac{V\Sigma^T\Sigma V^T}{n-1}$$

And because

① The loadings of the PC's are the eigenvectors of the covariance matrix (or correlation matrix, depending on whether you centered or centered & scaled the data)

$\Rightarrow$ The loadings are given by $\underset{(q\times q)}{V} = \begin{bmatrix} | & & | \\ \vec{v}_1 & \cdots & \vec{v}_q \\ | & & | \end{bmatrix}$.

Then,

② The eigenvalues of S_{X_c} are given by the matrix

$$\Lambda = \frac{1}{n-1}\Sigma^T\Sigma \quad \text{sometimes written as} \quad \frac{\Sigma^2}{n-1}$$

where each $\lambda_i = \frac{\sigma_i^2}{n-1}$ explains λ_i% of the variability in X_c

And,

③ The PC scores are given by

$$\underset{(n\times q)}{X_{pc}} = \underset{(n\times q)}{X_c}\ \underset{(q\times q)}{V} = \underset{(n\times n)}{U}\ \underset{(n\times q)}{\Lambda}$$

and more often by

★ $$\underset{(n\times r)}{X_{pc}} = \underset{(n\times q)}{X_c}\ \underset{(q\times r)}{\hat{V}} = \underset{(n\times r)}{\hat{U}}\ \underset{(r\times r)}{\hat{\Lambda}}$$ ← rank reduced form

[MORE ON THIS IN THE NEXT SECTION]

And recall that in PCA

*if we use the covariance matrix, most algorithms first center the data, then the

WHEN WE SCALE THE DATA (center it **AND** divide by standard deviation)

Then $S_{X_s} = R_{X_s}$

In other words, the correlation matrix is equal to the covariance matrix when we scale X. Thus, PCA proceeds in the same way.

The General Procedure:

STEP 1: center (or scale) X

STEP 2: Find the SVD of X_c (or X_s)

STEP 3:

- ★ Loadings : V
- ★ Eigenvalues : $\Lambda = \frac{1}{n-1}\Sigma^T\Sigma$
- ★ PC scores : $X_{pc} = \underbrace{X_c V} = \underbrace{U \Lambda}$

Recall that (most, if not all) SVD algorithms find the $\vec{v}_i$'s then obtain $\vec{u}_i$ through $\vec{u}_i = \frac{X_c \vec{v}_i}{\sigma_i}$

In the next section, we will connect how a condition # relates to rank deficiency through case 2 of $\mathcal{X}$'s dimensionality.

Section 3.1: Why are there only $n-1$ Loadings when $q >> n$?

When we "center" a dataset $\mathcal{X}$, we center it about the column means.

For $j = 1, \dots, q$

$$\underbrace{X_{\text{centered},\, j}}_{\text{The } j^{\text{th}} \text{ column in } X} = \begin{bmatrix} x_{1,j} - \frac{1}{n}\sum_{i=1}^{n} x_{i,j} \\ \vdots \\ x_{n,j} - \frac{1}{n}\sum_{i=1}^{n} x_{i,j} \end{bmatrix}$$

$$\text{rank}(\mathcal{X}_c) \le \min(n, q)$$

$$\le$$

Recall the following,

When $n > q$ (case I),

① $r = \text{rank}(\mathcal{X}_c) \le \min(n, q)$

$\Rightarrow r \le q$

② In the SVD of $\mathcal{X}_c$, when $r = q$ (full rank), $V = \hat{V}$

When $q > n$ (case II)

① $r = \text{rank}(X_c) \leq \min(n, q)$
$\Rightarrow r \leq n$

② In the SVD of X_c, when $r = n$ (full rank), $U = \hat{U}$ and $V_{q \times q} \neq \hat{V}_{q \times n}$. However, the vectors $\vec{v}_i$ inside $\hat{V}_{q \times n}$ in the full rank case have all nonzero eigenvalues

CLAIM: Data that is centered around the mean yields one row that is linearly dependent. In other words,

★ when $n > q$, centered data X_c's rank is not effected by this

★ when $q \geq n$, the matrix X_c is rank deficient

Simultaneously,

① Algorithms struggle to find eigenvalues and eigenvectors when X_c is rank deficient

② These algorithms that often work by finding $\hat{V}_{q \times r}$, and don't look for $V_{q \times q}$ ... (the complete SVD)

③ Graham-Schmidt is useful in finding the $q-n$ extra orthogonal vectors, but the rank deficiency inevitably makes it more challenging for them to find the last vector in V

④ If the algorithm works by "picking off" columns, sometimes the $\frac{1}{n-1}$ doesn't work for this last case as well

A side note is that some PC algorithms also use $\frac{1}{n}$ instead of $\frac{1}{n-1}$ as the coefficient in S. This yields a more conservative estimate of variances. This is not a standard, but some (usually older) PC functions do this as a way to counteract ④

Really, mathematicians care more about this than statisticians. If you are interested in how the U, $\hat{U}$, V, $\hat{V}$, Σ, and $\hat{\Sigma}$ matrices are obtained, there are many resources only that explain popular algorithms (and their respective computational complexities, like FLOPs). For statisticians, however, who work with big data, computational complexity can become very important to their work and can be useful knowledge when analyzing data or problem-solving.

THE BIG TAKEAWAY:

The big takeaway is that there are ALWAYS n rows in a PC score (whether $n > q$, $q \geq n$, or there is rank deficiency).

However, for algorithms that approximate the vectors in the SVD, there are only r loadings and r PC score vectors. Therefore when rank $< \min(n, q)$, r will differ from the row count when $q \geq n$.

On your own, you can try to show that a row is linearly dependent when the matrix is centered about column means... but that concludes this lesson!