Review:

(2) Mean and variance and covariance:

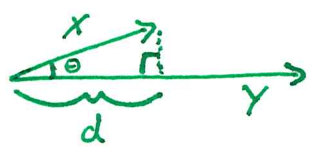
$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X^{(i)}$$

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}\left(X^{(i)} - \bar{X}\right)^2$$

$$Cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X^{(i)} - \bar{X})(Y^{(i)} - \bar{Y})$$

(4) Dot product, ~~projections~~ orthogonal projections, ~~orthogonal~~ orthonormal basis, and orthogonal matrix:

$$X \cdot Y = \sum_{i=1}^{p} X_i Y_i$$



Recall:

$$\cos(\theta) = \frac{X \cdot Y}{|X||Y|}$$

$$\cos(\theta) = \frac{d}{|X|}$$

$$d = \frac{X \cdot Y}{|Y|} = X \cdot \hat{Y}$$

$$V = \begin{bmatrix} | & | & \\ v_1 & v_2 & \cdots \\ | & | & \end{bmatrix}_{p \times p}$$

Let V be an orthogonal basis. That means the cols of V are all orthogonal to one another and have unit length.

<u>Note</u>:

$$V^T V = I$$

$\therefore V^T = V^{-1}$

$\therefore V V^T = I$

Put an orthonormal basis into a matrix to make an orthogonal matrix

PCA method:

(1) Our dataset:

n samples

p features

$X^{(1)}, X^{(2)}, \dots X^{(n)}$

$X^{(i)} \in \mathbb{R}^p$

(3) Step 1: Create the "centered design matrix" M.

$$M = \begin{bmatrix} \text{---} \; X^{(1)} - \bar{X} \\ \text{---} \; X^{(2)} - \bar{X} \\ \vdots \\ \text{---} \; X^{(n)} - \bar{X} \end{bmatrix}_{n \times p}$$

(5) Step 2: Compute the covariance Matrix

$$\Sigma = M^T M / n$$

(7) Step 3: Compute the eigenvectors/values of Σ.

evals, evects = eig $(M^T M)$

The evect w/ the largest eval is the direction of most variance in the data.

... second most...

(8) 2d example:

evect1 is the "first principal component"

all evects are orthogonal.

PCA derivation:

→ what will Σ look like?

... It is like there is a little covariance between all pairs, and some pair will have (possibly) <u>a lot</u> of covariance.

<u>Crazy idea</u>: Could we find a new orthogonal basis, project our data to it, and have no covariance?

Let's see...

(no covariance)

$$(MV)^T(MV) = \begin{bmatrix} \lambda_1 & & \emptyset \\ & \lambda_2 & \\ \emptyset & & \ddots \; \lambda_p \end{bmatrix}$$

$$(MV)^T(MV) = V^T M^T M V$$

$$V V^T M^T M V = V \begin{bmatrix} \lambda_1 & & \emptyset \\ & \lambda_2 & \\ \emptyset & & \ddots \; \lambda_p \end{bmatrix}$$

$$M^T M V = V \begin{bmatrix} \lambda_1 & \ddots & \emptyset \\ \emptyset & & \lambda_p \end{bmatrix}$$

$$\begin{bmatrix} M^T M \end{bmatrix} \begin{bmatrix} | & | & \\ v_1 & v_2 & \cdots \\ | & | & \end{bmatrix} = \begin{bmatrix} | & | & \\ v_1 & v_2 & \cdots \\ | & | & \end{bmatrix} \begin{bmatrix} \lambda_1 & \ddots & \emptyset \\ \emptyset & & \lambda_p \end{bmatrix}$$

consider only one column of result on each side

(6)

$$\begin{bmatrix} M^T M \end{bmatrix} \begin{bmatrix} | \\ v_i \\ | \end{bmatrix} = \begin{bmatrix} | \\ v_i \\ | \end{bmatrix} \lambda_i \rightarrow Ax = \lambda x$$

→ Eigenvectors!

(9) The relative size of the eigenvalues tell us the % explained variance of each eigenvector.

Let $\lambda_1, \lambda_2, \ldots, \lambda_p$ be the decreasing list of eigenvalues. also called the "loadings".

total power $= \sum_{i=1}^{p} \lambda_i$

~~tota~~ power of k eigenvetors $= \dfrac{\sum_{i=1}^{k} \lambda_k}{\sum_{i=1}^{p} \lambda_p}$

Typically, set k s.t. 90% of the variance is explained.

(10) Keep k eigenvectors $\longrightarrow$ let $U_{p \times k} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_k \\ | & | & & | \end{bmatrix}$ where u_i are the principal componets. i.e. $eig(M^T M)$

Edit: (I should have called this V.)

(11) Reduce the dims. by $R = MU$ i.e. $R_{n \times k} = M_{n \times p} U_{p \times k}$

↳ reduced to k dims.

(12) Reconstruct an estimate of M by "undoing" the dim reduction:

$$R = MU \rightarrow \tilde{M} = RU^{-1} = RU^T$$

Singular Value Decomposition (SVD)

(1) Every matrix M has a unique decomposition in the following form:

$$M = USV^T$$

where:

$M_{n \times p}$ is any matrix

$U_{n \times n}$ is an orthogonal matrix

$S_{n \times p}$ is a diagonal matrix with positive dec. real values on the diagonal
↳ these values are the "singular values" of M.

$V^T_{p \times p}$ is an orthogonal matrix

Some Properties:

The columns of U are the eigenvectors of MM^T

The columns of V are the eigenvectors of M^TM

If the rank of M is k, the SVD can look like this:

$$M_{n \times p} = U_{n \times k} S_{k \times k} V^T_{k \times p}$$

Note: $k \leq \min(n, p)$

(2) We can already see the SVD is computing the same thing as PCA from our description on the left: Both compute $\boxed{eig(M^TM)}$

Here's another way to see the same thing (that SVD and PCA are doing the same thing):

Recall: (from PCA)

$$\Lambda = \begin{bmatrix} \lambda_1 & & \emptyset \\ & \ddots & \\ \emptyset & & \lambda_p \end{bmatrix}$$

$(MV)^T MV = \Lambda$

$V^T M^T M V = \Lambda$

$M^T M V = V\Lambda$

$M^T M = V \Lambda V^T$

New: (in SVD world)

$M = USV^T$

$M^T M = (USV^T)^T USV^T$

$= VS^T U^T U S V^T$

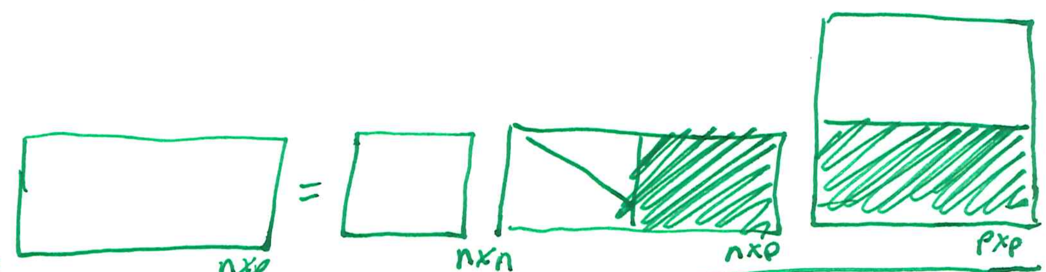
$= V S^T S V^T$

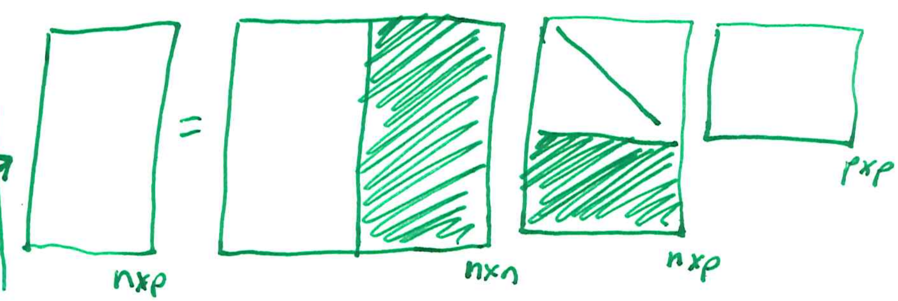
$M^T M = V S^2 V^T$

In SVD world, V is the matrix where the cols are $eig(M^TM)$

$\Lambda = S^2$

↳ the evals are the squared singular vals.





So... SVD and PCA do the same thing... but there are two ways SVD wins.

SVD FTW #1:

(3) When we calculated eigenfaces...

M is $n \times p$ where

$n = 105$

$p = ~~[illegible]~~ \; 320 \times 243 = 77{,}760$

$M^T M$ is $p \times p$...

... so $M^T M$ contains

6,046,617,600

real values ~~[illegible]~~ (floats in python)

... a float is 8 bytes, so

we need ~ 48 GB of RAM

to store $M^T M$.

We can shortcut this with SVD.

SVD allows ~~[illegible]~~ us to ~~calculate~~ compute

eig$(M^T M)$ without actually

~~calculating~~ computing $M^T M$.

Using "compact SVD": $M_{n \times p} = U_{n \times r} S_{r \times r} V^T_{r \times p}$

where $r \leq n$ (r = rank of M)

SVD FTW #2:

(4) SVD can reveal ~~[illegible]~~ "latent features" in your data. This can be used for "topic modeling" in datasets of user ratings.

... switch to IPython notebook

because we don't want

to be writing a ton of

real values on the board...

... we'll do a real example

of this (w/ real numbers)

in the notebook...

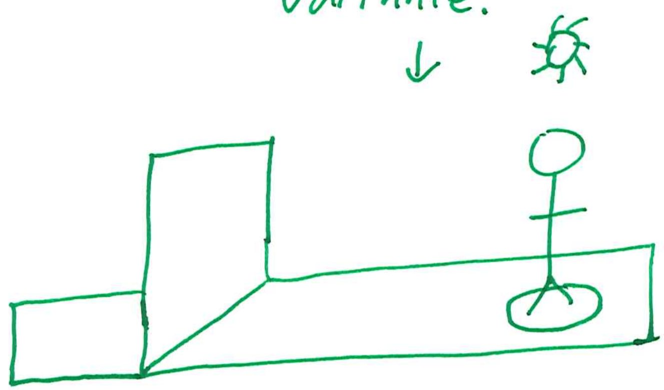
Takeaway: SVD is more computationally efficient than PCA when $n \ll p$.

PCA analogy to shadows cast by my body!

Projection from 3d to 2d via the sun.

↳ __Not__ in the direction of my body's most variance.

↓



In the direction of my body's most variance.

↓