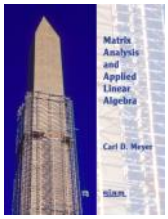
Lecture 12: Projections

Admin:



5.13

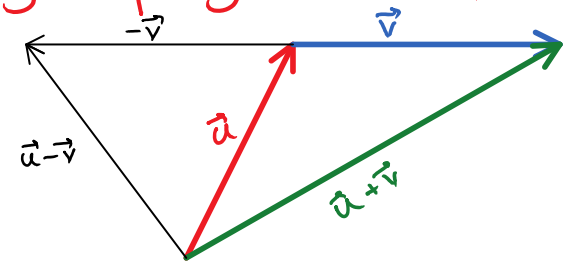
3.1-3.3

Recall:

Inner product: $\vec{u} \cdot \vec{v} = \sum_i u_i^* v_i$ (complex conjugate)

Length: $\|\vec{u}\| = \sqrt{\vec{u} \cdot \vec{u}}$

Triangle inequality: $\|\vec{u} + \vec{v}\| \leq \|\vec{u}\| + \|\vec{v}\|$

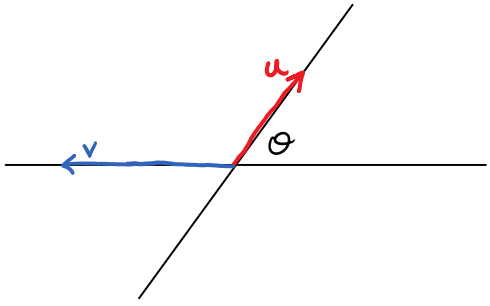


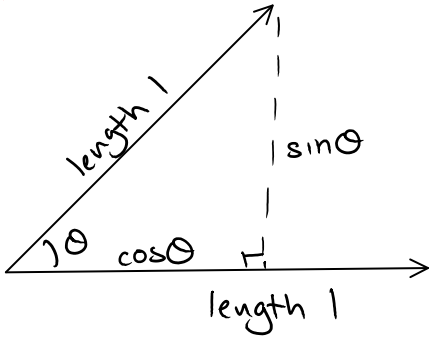
Corollary: $\|u\| \leq \|v\| + \|u - v\|$

$\Rightarrow \|u - v\| \geq \|u\| - \|v\|$

Angles: angle between lines Span(u) and Span(v)

$$\cos\theta = \frac{|\vec{u} \cdot \vec{v}|}{\|u\| \cdot \|v\|}$$





Cauchy-Schwarz inequality: $|\vec{u} \cdot \vec{v}| \leq \|\vec{u}\| \cdot \|\vec{v}\|$

(with equality iff collinear)

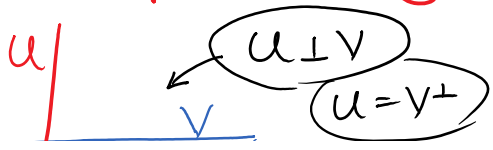
Example: $\sum_{i=1}^{n} u_i = (1, 1, \ldots, 1) \cdot \vec{u}$

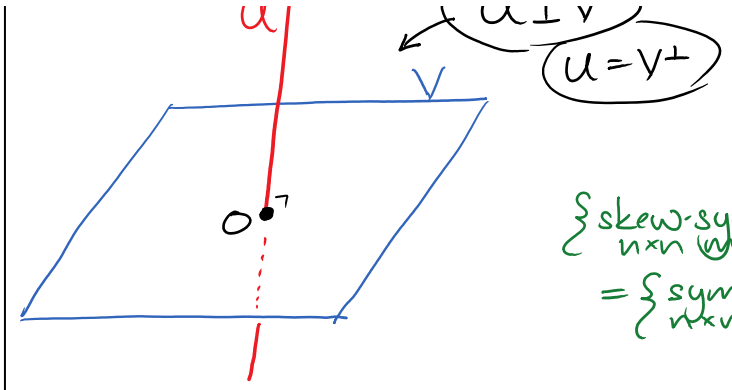
$\leq \|(1, \ldots, 1)\| \cdot \|\vec{u}\|$

$= \sqrt{n} \cdot \|\vec{u}\|$

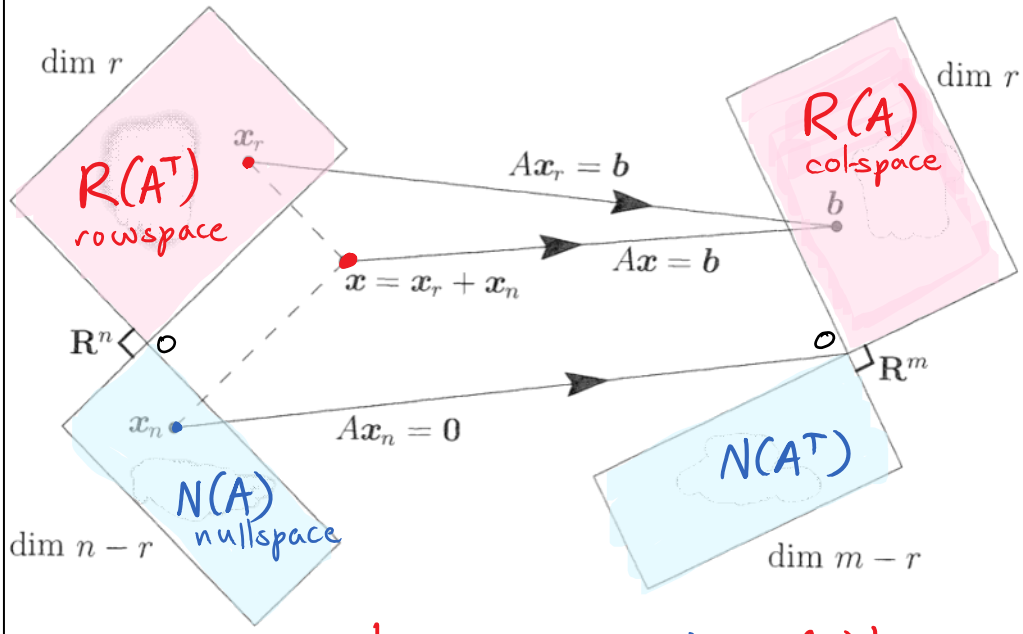
Orthogonal vectors: $\vec{u} \perp \vec{v} \iff \vec{u} \cdot \vec{v} = 0$

Orthogonal subspaces $\neq$ orthogonal complements:





$\{\text{skew-symmetric } n\times n \text{ matrices}\}^\perp = \{\text{symmetric } n\times n \text{ matrices}\}$



$N(A) = R(A^T)^\perp$ $\quad$ $N(A^T) = R(A)^\perp$

Consequences:

* $\{x \mid Ax = b\}$ is an $(n-r)$-dim affine space.
* Any subspace $\mathcal{U}$ can be described either
  - by a basis for $\mathcal{U}$ ($\dim \mathcal{U}$ vectors), or
  - by a basis for $\mathcal{U}^\perp$ ($n - \dim \mathcal{U}$ vectors)

Today: Projections and orthogonal bases

PROJECTIONS

Motivation: Principal component analysis (PCA)

High-dimensional data set → fewer dimensions

High-dimensional data set → fewer dimensions

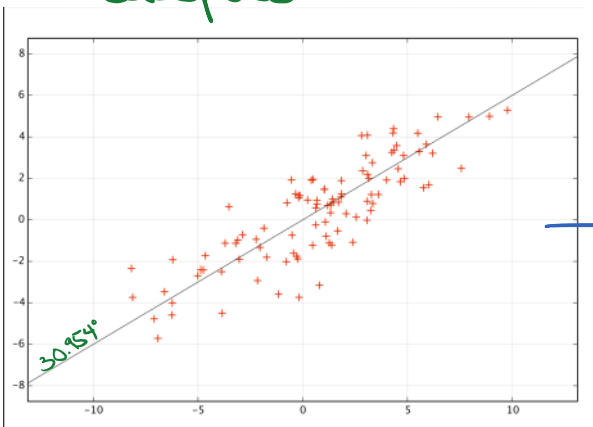
Example:

generate noisy data:

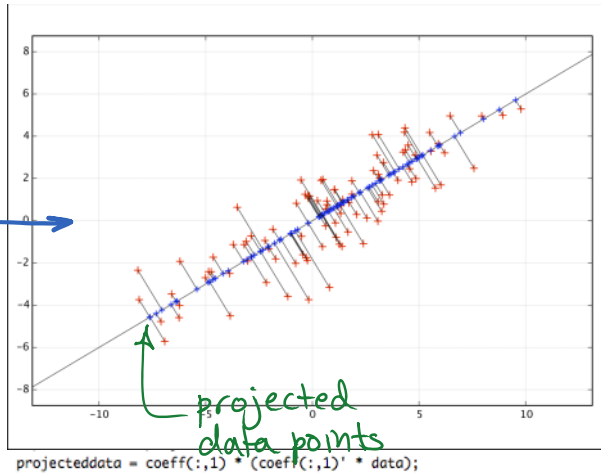
```
n = 100; largerstddev = 5; rotationangle = 30;
c = cosd(rotationangle); s = sind(rotationangle);
data = [c,-s; s,c] * [largerstddev,0; 0,1] * randn(2, n);
```

① Find best-fitting low-dim subspace

② Project data to subspace




```
# run PCA to extract best-fitting line
[coeff,score] = princomp(data');
slope = coeff(2,1) / coeff(1,1);
fitangle = atand(slope)
```




```
projecteddata = coeff(:,1) * (coeff(:,1)' * data);
```

Applications: Machine learning, statistics, data analysis, compression,...

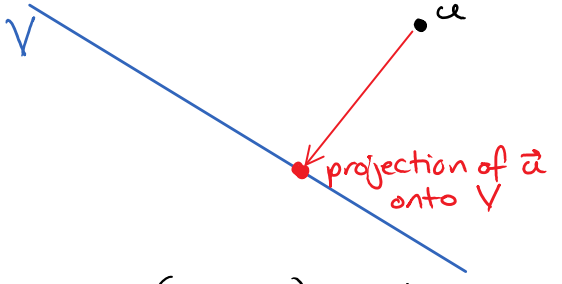
We'll cover PCA when we get to the singular-value decomp.

Problem: How can we make all these PROJECTIONS?

Definition: Let V be a subspace of $\mathbb{R}^n$ (or $\mathbb{C}^n$).

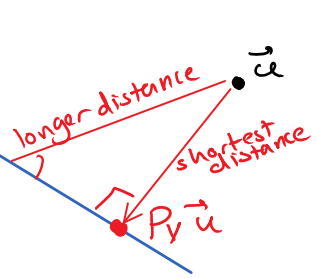
The orthogonal projection onto V maps

any point $u \in \mathbb{R}^n$ ⟼ the closest point in V



Observe: $(\vec{b} - \vec{v}) \perp V$ distance $\vec{u}$

Observe: 1. $(P_V\vec{u} - \vec{u}) \perp V$.



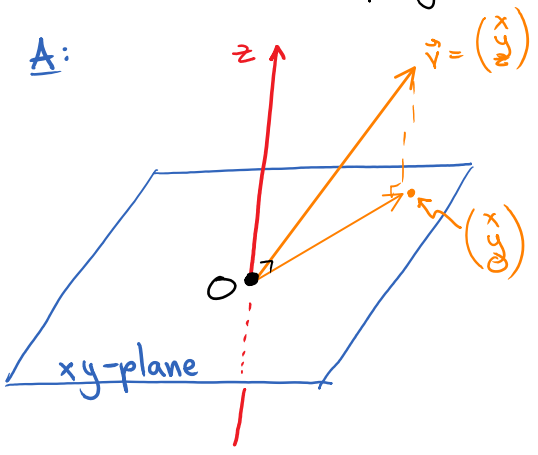
2. If $u \in V$, $P_V u = u$.
3. $P_V^2 = P_V$
   (since $P_V\vec{u} \in V$, $P_V(P_V\vec{u}) = P_V\vec{u}$)
4. Projections are linear transformations
   ($P_V(\alpha\vec{u}) = \alpha P_V\vec{u}$, $P_V(\vec{u}+\vec{w}) = P_V\vec{u} + P_V\vec{w}$)
5. $I - P_V = P_{V^\perp}$ projection onto $V^\perp$
   where $V^\perp = \{$all vectors $\perp$ to $V\}$, the "orthogonal complement" of V. Recall $\dim(V^\perp) = n - \dim(V)$.

→ We'll prove these properties later!

Intuition: Projection to a coordinate subspace

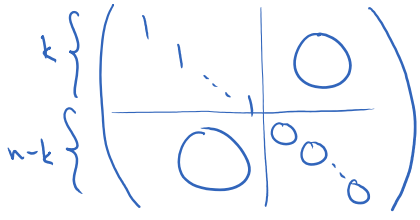
Q: What is the projection from $\mathbb{R}^3$ to the xy-plane?

A:



The matrix $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ discards the z-coordinate of any point.

More generally



projects onto the subspace $\text{Span}\{\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_k\}$ in $\mathbb{R}^n$.

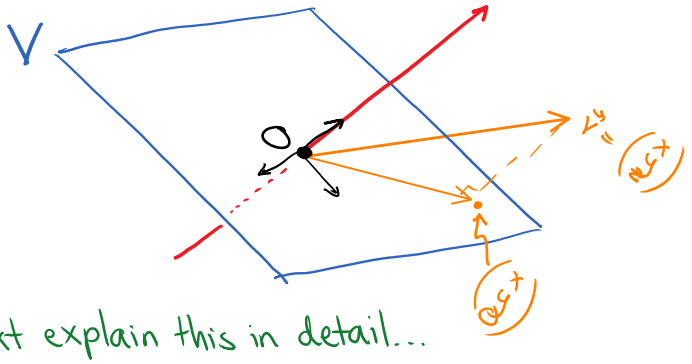
More generally:

Projecting onto an ⟷

- Change basis to orthogonal coordinate system starting with basis for V

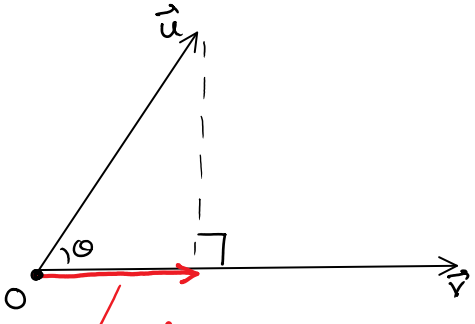
Projecting onto an **arbitrary** subspace V ⟷

- Change basis to orthogonal coordinate system starting with basis for V
- Discard the coordinates ⊥ to V



We'll next explain this in detail...

Example: Projection onto a line



$\vec{u}$ projection of onto the line Span($\vec{v}$) $= (\vec{v}\cdot\vec{u})\vec{v}$ if $\|\vec{v}\| = 1$

Lemma: The projection of $\vec{u}$ onto the line Span($\vec{v}$) is

(★) $(\vec{v}\cdot\vec{u})\vec{v}$ if $\|\vec{v}\|=1$ (★)

$\frac{\vec{v}\vec{v}^T}{\|\vec{v}\|^2}\vec{u} = \frac{(\vec{v}\cdot\vec{u})}{\|\vec{v}\|^2}\vec{v}$ in general

The matrix that projects onto Span($\vec{v}$) is

$$\frac{\vec{v}\,\vec{v}^T}{\|\vec{v}\|^2}$$

↑ an $n\times n$ matrix if $\vec{v}$ is $n\times 1$

– since $(\vec{v}\vec{v}^T)\vec{u} = \vec{v}(\vec{v}^T\vec{u}) = \vec{v}(\vec{v}\cdot\vec{u})$ ✓
$\|\vec{u}\|\cos\theta$

Example: $\mathbb{R}^3$:

$\|\vec{u}\|\cos\theta$

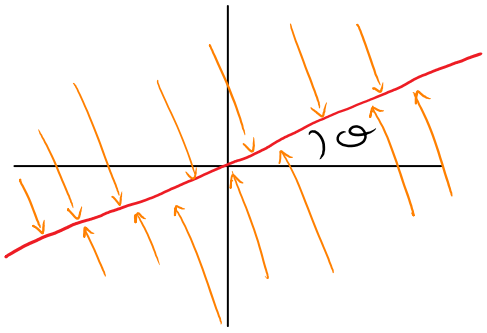
Example: $\mathbb{R}^3$:

$$\vec{e}_x = \begin{pmatrix}1\\0\\0\end{pmatrix},\ \vec{e}_y = \begin{pmatrix}0\\1\\0\end{pmatrix},\ \vec{e}_z = \begin{pmatrix}0\\0\\1\end{pmatrix}$$

$$\vec{e}_y\vec{e}_y^T = \begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}$$

$$\vec{e}_y\vec{e}_y^T\begin{pmatrix}\alpha\\\beta\\\gamma\end{pmatrix} = \begin{pmatrix}0\\\beta\\0\end{pmatrix}$$

Example: Projection onto the line at angle θ



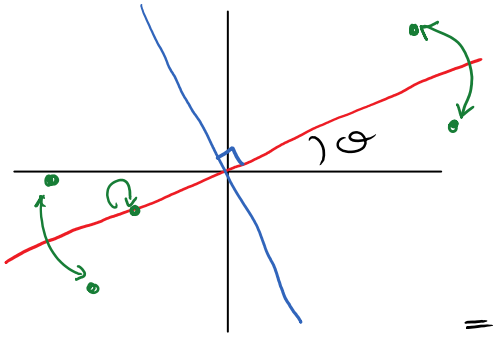
$$\vec{v} = \begin{pmatrix}\cos\theta\\\sin\theta\end{pmatrix}$$

$$\begin{pmatrix}\cos\theta\\\sin\theta\end{pmatrix}(\cos\theta, \sin\theta)$$

$$= \begin{pmatrix}\cos^2\theta & \cos\theta\sin\theta\\ \sin\theta\cos\theta & \sin^2\theta\end{pmatrix}$$

$$P_\theta''$$

Example: Reflection about the line at angle θ



$$P_\theta - (I - P_\theta)$$

P_θ: leaves vector in red space unchanged

$-(I-P_\theta)$: puts a minus sign on vectors in the orthogonal blue line

$$= 2P_\theta - I = \begin{pmatrix}2c^2-1 & 2cs\\ 2cs & 2s^2-1\end{pmatrix}$$

$$= \begin{pmatrix}\cos 2\theta & \sin 2\theta\\ \sin 2\theta & -\cos 2\theta\end{pmatrix}$$

Example: Projection onto $(1,1,\ldots,1)$

$$P = \frac{1}{\sqrt{n}}\begin{pmatrix}1\\1\\\vdots\\1\end{pmatrix}\cdot\frac{1}{\sqrt{n}}(1\ 1\ \cdots\ 1) = \frac{1}{n}\begin{pmatrix}1&1&\cdots&1\\1&1&\cdots&1\\\vdots&\vdots&\ddots&\vdots\\1&1&\cdots&1\end{pmatrix}$$

Observe: $P_v = (\text{avg. of } v\text{'s coords})\cdot\begin{pmatrix}1\\1\\\vdots\\1\end{pmatrix}$

Example: Projection to $(1,1,\ldots,1)^\perp = R\left(\begin{pmatrix}-2&1&&&\\1&-2&1&&\\&1&-2&1&\\&&1&\ddots&\\&&&&1\end{pmatrix}\right)$.

$= I - P$

Exercise: What is the projection of

Exercise: What is the projection of

$$\vec{u} = (2, -2, 3)$$

onto the line

$$L = \left\{ (x,y,z) \;\middle|\; \begin{array}{l} x + 2y + 3z = 0 \\ x - y + 2z = 0 \end{array} \right\} \quad ?$$

Answer:

① First find a basis for L.

L is a line because it is the intersection of two 2D planes in $\mathbb{R}^3$; it is the nullspace of the matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & -1 & 2 \end{pmatrix}.$$

Use Gaussian elimination to find $N(A)$:

$$A \to \begin{pmatrix} 1 & 2 & 3 \\ 0 & -3 & -1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 7/3 \\ 0 & 3 & 1 \end{pmatrix}$$

$$\Rightarrow x = -\tfrac{7}{3} \cdot z, \quad y = -\tfrac{1}{3} z$$

$$\Rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = -\tfrac{1}{3} \begin{pmatrix} 7 \\ 1 \\ -3 \end{pmatrix} z \quad \text{is the general solution to } A \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$$

$$\Rightarrow N(A) = \text{Span}\{(7, 1, -3)\}$$

(Check: Yes, $A \begin{pmatrix} 7 \\ 1 \\ -3 \end{pmatrix} = 0$.)

② Normalize it: $\|(7,1,-3)\|^2 = 59$

$$\Rightarrow \vec{v} = \frac{1}{\sqrt{59}}(7,1,-3) \text{ is a unit vector}$$

③ Project

$$\begin{aligned} P_L \vec{u} &= (\vec{v}\,\vec{v}^T)\vec{u} \\ &= \vec{v}(\vec{v}^T\vec{u}) \\ &= (\vec{v} \cdot \vec{u})\vec{v} \\ &= \frac{1}{\sqrt{59}}(7 \cdot 2 + 1 \cdot (-2) - 3 \cdot 3) \cdot \frac{1}{\sqrt{59}}(7,1,-3) \end{aligned}$$

$$\boxed{P_L \vec{u} = \frac{3}{59}(7,1,-3)}$$

④ Check the answer:

$\frac{3}{59}(7,1,-3) \in L$: ✓ since it is a multiple of $\vec{v}$, and yes $x + 2y - 3z = 0$, $x - y + 2z = 0$.

$(\vec{u} - P_L \vec{u}) \perp L$:

$$(2,-2,3) - \frac{3}{59}(7,1,-3) = \frac{1}{59}(97, -121, 186)$$

$(97, -121, 86) \cdot (7, 1, -3) = 0$ ✓

Exercise: What is the projection of

$\vec{u} = (4, 0, 3)$

onto the line

$$L = \left\{ (x,y,z) \;\middle|\; \begin{array}{l} x + 2y + 3z = 6 \\ x - y + 2z = 0 \end{array} \right\} \quad ?$$

Answer:

① Find a constructive formulation for L':

$$L' = \left(\text{any particular solution to } A\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \end{pmatrix} \right) + N(A)$$

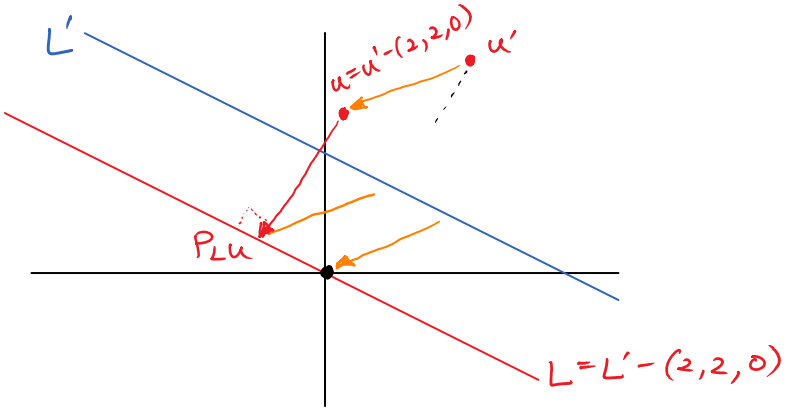
$$= (2, 2, 0) + \text{Span}((7, 1, -3))$$

General principle: To work with an **affine** subspace,

A. translate everything so it goes through 0

B. work there

C. translate back!



Since $L' - (2,2,0) = L$ (from above)

and $u' - (2,2,0) = u$,

the projection of u' onto L' is

$$(2,2,0) + \frac{3}{59}(7, 1, -3) = \frac{1}{59}(139, 121, -9)$$

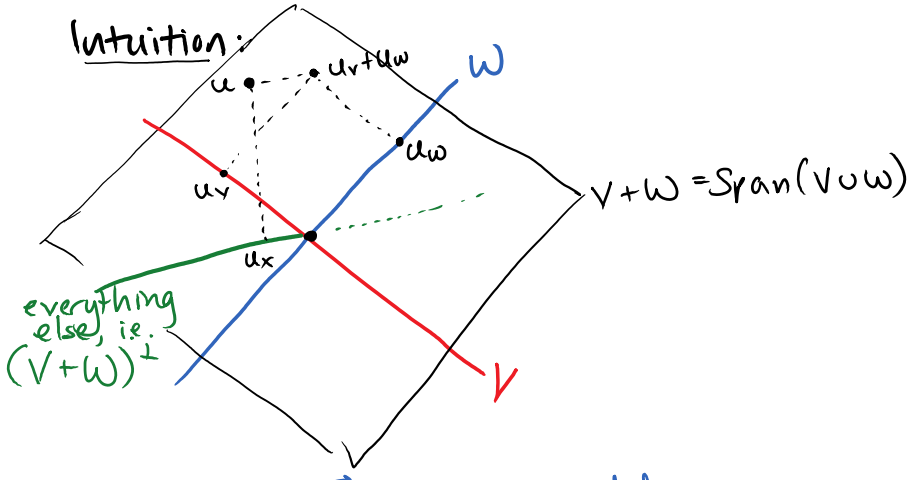
Problem: How can we construct the projector? onto higher-dim. subspaces

Key property: If $V \perp W$,

$$P_V + P_W = P_{V+W}$$

$$P_V + P_W = P_{V+W}$$

Intuition:



any vector $\vec{u}$ can be expanded as

$$\vec{u} = \vec{u}_V + \vec{u}_W + \vec{u}_X$$

where $\vec{u}_V \in V$, $\vec{u}_W \in W$, $\vec{u}_X \in X$

$P_V \vec{u}$ $P_W \vec{u}$ $P_X \vec{u}$

$$P_{V+W}\vec{u} = \vec{u}_V + \vec{u}_W$$

Examples & counterexamples:

$$P_{e_1} = e_1 e_1^T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

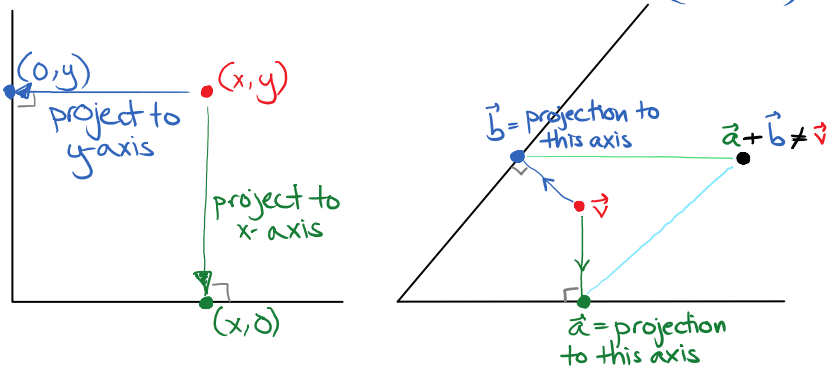
$$P_{e_2} = e_2 e_2^T = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$P_{e_1} + P_{e_2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = P_{x\text{-}y \text{ plane}}$$

This does not work if V is not $\perp$ to W:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \cos^2\theta & \cos\theta\sin\theta \\ \cos\theta\sin\theta & \sin^2\theta \end{pmatrix} = ? \text{ not a projection}$$

$(?^2 \neq ?)$



It only works because $e_1 \perp e_2$!

$(c, s) = (\cos\theta, \sin\theta)$

$$\begin{pmatrix} c \\ s \end{pmatrix}\begin{pmatrix} c & s \end{pmatrix} + \begin{pmatrix} s \\ -c \end{pmatrix}\begin{pmatrix} s & -c \end{pmatrix}$$

$$= \begin{pmatrix} c^2 & cs \\ cs & s^2 \end{pmatrix} + \begin{pmatrix} s^2 & -cs \\ -cs & c^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \checkmark$$

$\Rightarrow$ To project onto V, it is enough to have a basis of pairwise orthogonal vectors for V...

FACT: For a subspace $V \subseteq \mathbb{R}^n$ with orthonormal basis $\{\vec{v}_1, \dots, \vec{v}_k\}$, orthogonal projection onto V

$$P_V = \sum_{j=1}^{k} \underbrace{\vec{v}_j \vec{v}_j^T}_{n \times n \text{ matrix}}$$

Proof: $P_V = P_{\text{span}\{v_1, \dots, v_{k-1}\}} + P_{\text{span}\{v_k\}}$

$P_{\text{span}\{v_k\}} = v_k v_k^T$

Claim: For subspaces $V, V^\perp \subseteq \mathbb{R}^n$,

$$P_V + P_{V^\perp} = I$$

Proof:
Since V is orthogonal to $V^\perp$,

$$P_V + P_{V^\perp} = P_{V+V^\perp} = P_{\mathbb{R}^n} = I. \quad \square$$

Example: This can save a lot of time!

Let $A = \begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & & \\ 1 & & & & \end{pmatrix}_{n \times n}$, and $\vec{v} \in \mathbb{R}^n$.

Compute $P_{R(A)}\vec{v}$.

Answer 1: Recall that $\text{Rank}(A) = n - 1$, and

$$R(A) = \{\vec{x} \in \mathbb{R}^n \mid x_1 + \cdots + x_n = 0\}$$
$$= N(\begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}_{n \times 1})$$
$$= \text{Span}\{\vec{e}_1 - \vec{e}_2, \vec{e}_2 - \vec{e}_3, \vec{e}_3 - \vec{e}_4, \dots, \vec{e}_{n-1} - \vec{e}_n\}$$

make these orthonormal (hard!)

$$P_{R(A)} = \sum_{j=1}^{n-1} \vec{v}_j \vec{v}_j^T \quad \vec{v}_1, \ldots, \vec{v}_{n-1}$$

Better answer:

Let $\vec{1} = (1, 1, \ldots, 1) \in \mathbb{R}^n$

$R(A) = \text{Span}(\vec{1})^\perp$

$$\Rightarrow P_{R(A)} = I - P_{\vec{1}} = I - \frac{1}{n}\begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix}(1 \cdots 1)$$

In general, if $\dim V$ is close to n, then using $P_V = I - P_{V^\perp}$ will often simplify your calculations.

Recipe: How to make projections in Matlab

Problem: Given vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$, how do we project onto their span?

1. $A = \begin{pmatrix} | & | & & | \\ \vec{v}_1 & \vec{v}_2 & \cdots & \vec{v}_n \\ | & | & & | \end{pmatrix}$
2. $B = \text{orth}(A)$ in Matlab
3. projection matrix $= BB^T$

in Python:
scipy.linalg.orth()
numpy.linalg.qr()

Why? $B = \begin{pmatrix} | & | & & | \\ \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \\ | & | & & | \end{pmatrix}$

↑ orthonormal
$\text{Span}(\{\vec{u}_j\}) = \text{Span}(\{\vec{v}_j\})$

$$= \sum_j \vec{u}_j \vec{e}_j^T$$

$$\Rightarrow B \cdot B^T = \sum_{j,k} \vec{u}_j (\vec{e}_j^T \vec{e}_k) \vec{u}_k^T$$

$\vec{e}_j^T \vec{e}_k = \begin{cases} 0 & \text{if } j \neq k \\ 1 & \text{if } j = k \end{cases}$

$$= \sum_j \vec{u}_j \vec{u}_j^T = P_{\text{Span}(\{\vec{u}_j\})} \quad \checkmark$$

Exercise:
Compute the projection of $\vec{e}_1$ onto a random

100-dimensional subspace V of $\mathbb{R}^{10000}$.

Also: What is the expected squared length of the projection?

Answer:

```
n = 10000;
d = 100;
e1 = zeros(n,1); e1(1) = 1;
A = randn(n, d);
B = orth(A);
```

~~projection = B * B' * e1;~~ ← don't do this

neither in Matlab nor in hand calculations

```
projection = B * (B' * e1);
```

```
>> P = B * B';
>> whos P
```

needs n^2 memory

Name	Size	Bytes
P	10000x10000	800000000

```
>> whos B
```

needs $n \times d$ memory

Name	Size	Bytes
B	10000x100	8000000

```
>> format long
>> norm(projection)^2

ans =

   0.009994012962916
```

$\approx .01 = \frac{d}{n}$ Why?

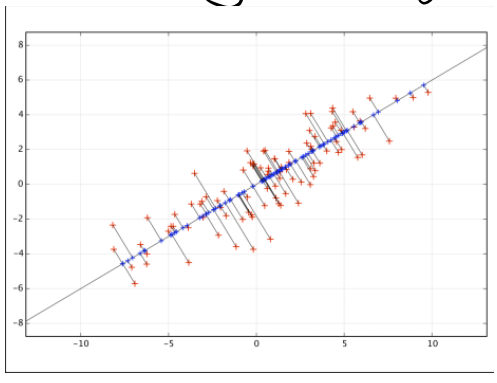
Gram-Schmidt orthogonalization

(see the other notes)

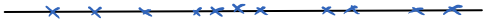
More examples of projections

① Find the shortest $\vec{x}$ solving $A\vec{x} = \vec{b}$

② Working with projected data

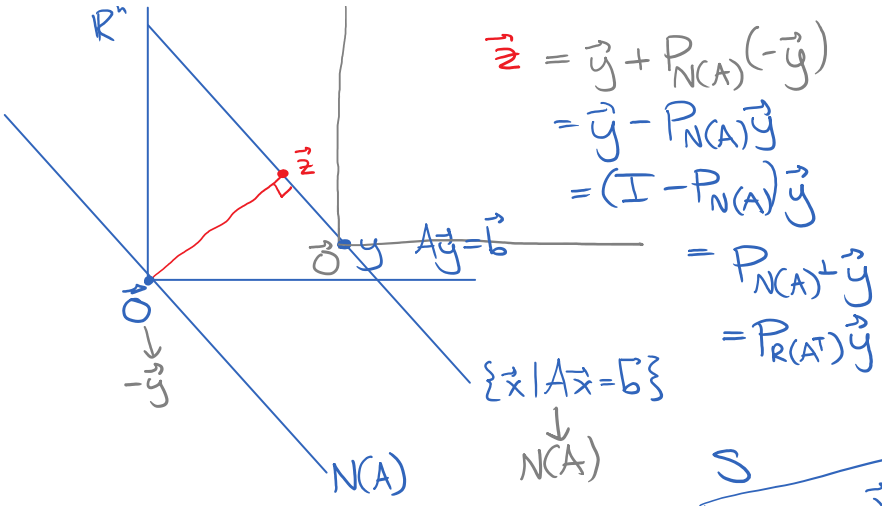


vs.



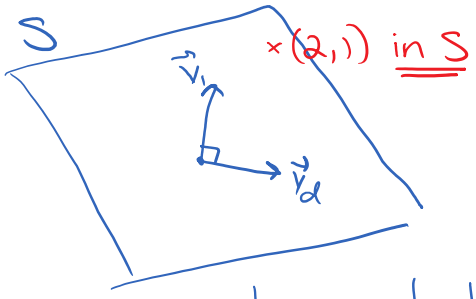
① Find the shortest $\vec{x}$ solving $A\vec{x} = \vec{b}$

$\vec{z} = \vec{y} + P_{N(A)}(-\vec{y})$

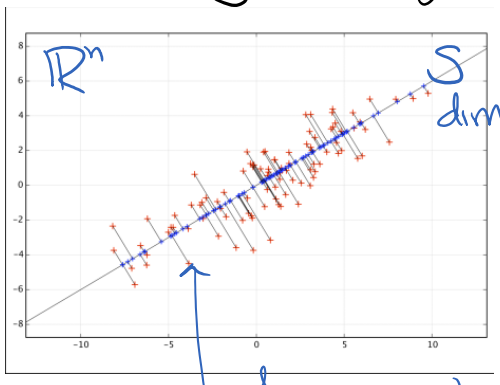


$$\vec{z} = \vec{y} + P_{N(A)}(-\vec{y})$$
$$= \vec{y} - P_{N(A)}\vec{y}$$
$$= (I - P_{N(A)})\vec{y}$$
$$= P_{N(A)^\perp}\vec{y}$$
$$= P_{R(A^T)}\vec{y}$$

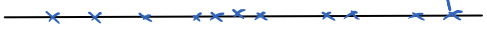
(Matlab example)



② Working with projected data



vs.



keep only the d coords within S

$$P_S\vec{x} = \sum_{j=1}^{d} \vec{v}_j(\vec{v}_j^T\vec{x})$$

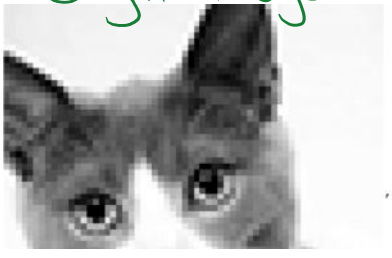
if $\vec{v}_1,\ldots,\vec{v}_d \in \mathbb{R}^n$ are an orthonormal basis for S

$$Q_S\vec{x} = \begin{pmatrix} \vec{v}_1\cdot\vec{x} \\ \vec{v}_2\cdot\vec{x} \\ \vdots \\ \vec{v}_d\cdot\vec{x} \end{pmatrix} = \sum_{j=1}^{d} \vec{e}_j(\vec{v}_j^T\vec{x})$$
$$= \left(\sum_{j=1}^{d} \vec{e}_j\vec{v}_j^T\right)\vec{x}$$
$$= \begin{pmatrix} \text{---}\ \vec{v}_1^T\ \text{---} \\ \text{---}\ \vec{v}_2^T\ \text{---} \\ \vdots \end{pmatrix}$$

(Matlab example)

Example: Image completion

Original image



Erased 40% of the coordinates

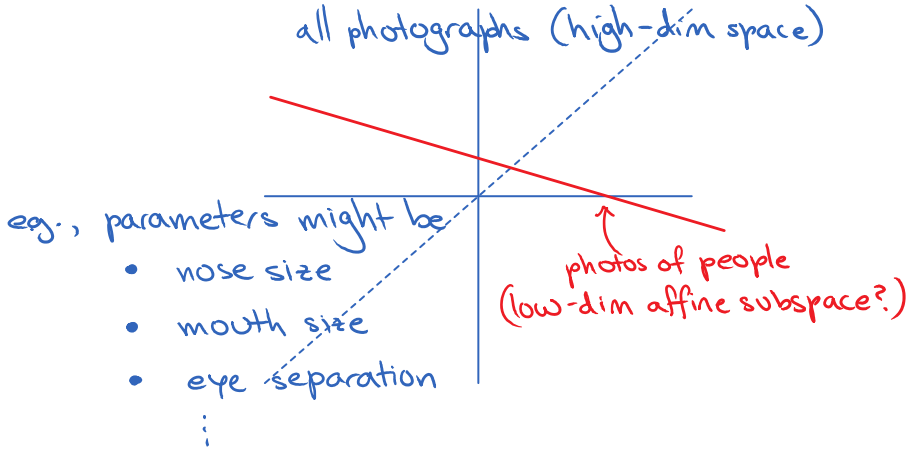
Solved to minimize $\|\vec{x}_{Haar}\|^2$ with constrained coordinates

Experiment: How many degrees of freedom are there in a face?

Hypothesis: Not many! (?)

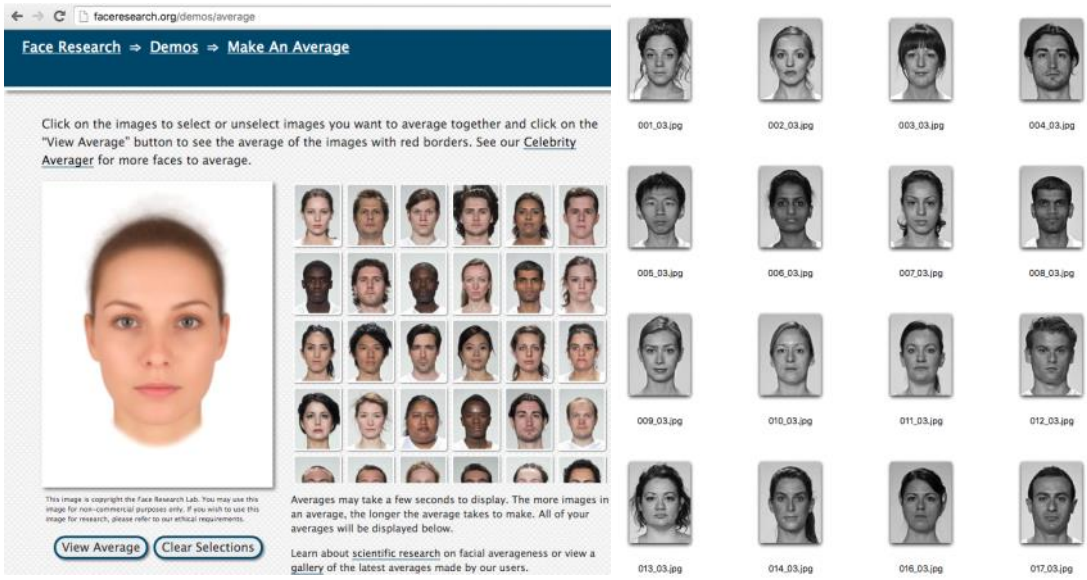
Idea:

photograph $\in \mathbb{R}^{\text{number of pixels}}$



Approach:

① Scrape 100+ photos from http://faceresearch.org/
Resize and convert to grayscale (for simplicity)



(2) Load images into Matlab as vectors

```
files = {'001_03.jpg', '002_03.jpg', '003_03.jpg', '004_03.jpg', '005_03.jpg', '006_03.jpg', '007_03.jpg',
'008_03.jpg', '009_03.jpg', '010_03.jpg', '011_03.jpg', '012_03.jpg', '013_03.jpg', '014_03.jpg', '016_03.jpg',
'017_03.jpg', '018_03.jpg', '019_03.jpg', '020_03.jpg', '021_03.jpg', '022_03.jpg', '024_03.jpg', '025_03.jpg',
'026_03.jpg', '027_03.jpg', '029_03.jpg', '030_03.jpg', '031_03.jpg', '032_03.jpg', '033_03.jpg', '034_03.jpg',
'036_03.jpg', '037_03.jpg', '038_03.jpg', '039_03.jpg', '041_03.jpg', '042_03.jpg', '043_03.jpg', '044_03.jpg',
'045_03.jpg', '061_03.jpg', '062_03.jpg', '063_03.jpg', '064_03.jpg', '066_03.jpg', '067_03.jpg', '068_03.jpg',
'069_03.jpg', '070_03.jpg', '081_03.jpg', '082_03.jpg', '083_03.jpg', '086_03.jpg', '087_03.jpg', '088_03.jpg',
'090_03.jpg', '091_03.jpg', '092_03.jpg', '094_03.jpg', '096_03.jpg', '097_03.jpg', '099_03.jpg', '100_03.jpg',
'101_03.jpg', '102_03.jpg', '103_03.jpg', '104_03.jpg', '105_03.jpg', '107_03.jpg', '108_03.jpg', '111_03.jpg',
'112_03.jpg', '113_03.jpg', '114_03.jpg', '115_03.jpg', '117_03.jpg', '118_03.jpg', '119_03.jpg', '120_03.jpg',
'121_03.jpg', '122_03.jpg', '123_03.jpg', '124_03.jpg', '125_03.jpg', '126_03.jpg', '127_03.jpg', '128_03.jpg',
'129_03.jpg', '130_03.jpg', '131_03.jpg', '132_03.jpg', '134_03.jpg', '135_03.jpg', '136_03.jpg', '137_03.jpg',
'138_03.jpg', '139_03.jpg', '140_03.jpg', '141_03.jpg', '142_03.jpg', '143_03.jpg', '144_03.jpg', '172_03.jpg',
'173_03.jpg'};

n = length(files);                  % n = 103
imagesize = size(imread(files{1}))  % 100 x 75
m = imagesize(1) * imagesize(2)     % m = 7500

images = zeros(m,n);
for i = 1:n
  image = imread(files{i});
```

flattens 100×75 array into 7500×1 column vector

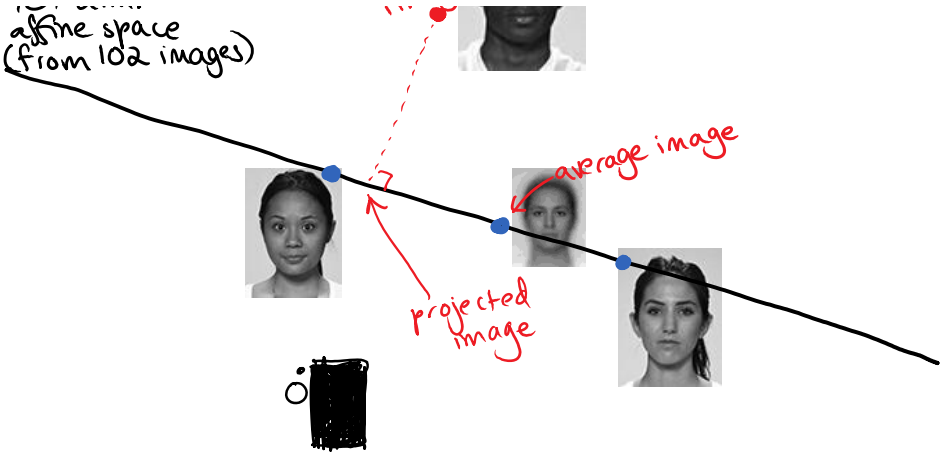
(3) Remove a random image

```
% pick a random image to remove
randomindex = randi(n);
randomimage = images(:,randomindex);  % select and remove that image
images(:,[randomindex]) = [];
n = n - 1;                            %  and decrement n
imwrite(reshape(randomimage, imagesize(1), imagesize(2))/255, 'randomimage.png');
```

(4) Project it onto the affine span of the others

101-dim. affine space (from 102 images)

original image




```
% recenter about the mean image
meanimage = sum(images,2) / n;  % sum(A,2) adds up the columns of A (i.e., the second dimension)
for i = 1:n
  images(:,i) = images(:,i) - meanimage;
end

O = orth(images);   % size(O) is 7500x102
projectedimage = meanimage + O * (O' * (randomimage-meanimage));
imwrite(reshape(projectedimage, imagesize(1), imagesize(2))/255, 'projectedimage.png');
```

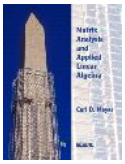
Idea: If faces form a low-dimensional affine subspace, then the projection should be close to the original image.

Result:

Original image

Projected image

Lecture 12: Gram-Schmidt orthogonalization

Reading:  3.4  5.5

Recall:

Orthogonal basis → pairwise orthogonal vectors

Orthonormal basis → orthogonal, length-one vectors

MORAL: Orthonormal bases behave just like the standard basis.

eg., for $\vec{u} = \sum_j \alpha_j \vec{v}_j$, $\vec{v} = \sum_j \beta_j \vec{v}_j$,

$$\vec{u} \cdot \vec{v} = \sum_j \alpha_j^* \beta_j$$

FACT: For a subspace $U \subseteq \mathbb{R}^n$ with orthonormal basis $\{\vec{u}_1, \ldots, \vec{u}_k\}$, orthogonal projection onto U

$$P_U = \sum_{j=1}^{k} \vec{u}_j \vec{u}_j^T$$

Example: (coordinate expansion)

$$\vec{v} = P_{\mathbb{R}^n} \vec{v}$$

$$= \sum_{j=1}^{n} \vec{u}_j \vec{u}_j^T \vec{v}$$

($\vec{u}_j^T \vec{v} = \vec{u}_j \cdot \vec{v}$)

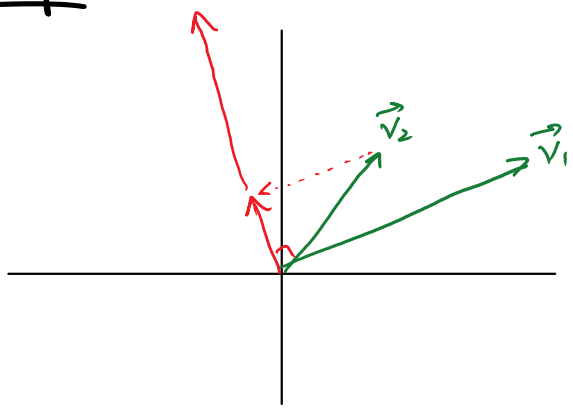
$$\boxed{\vec{v} = \sum_j (\vec{u}_j \cdot \vec{v}) \vec{u}_j}$$

Today:

How to get an orthonormal basis

Example:

Example:



General problem: Let $S = \{\vec{v}_1, \ldots, \vec{v}_k\}$ be a set of vectors in $\mathbb{R}^n$ or $\mathbb{C}^n$.

Goal: Find an orthonormal basis for Span(S).

Step One: Find any basis for Span(S).
(because the vectors might be linearly dependent)

Let $A = \begin{pmatrix} \vec{v}_1^T \\ \vec{v}_2^T \\ \vdots \\ \vec{v}_k^T \end{pmatrix}$. $\text{Span}(S) = \underset{\text{rowspace}}{R(A^T)}$.

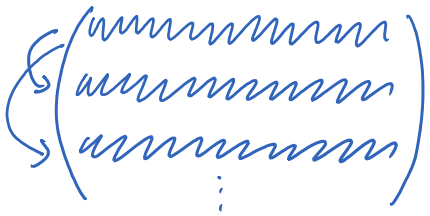
Gaussian elimination on A leaves a basis.

Step Two: Adjust the basis to be orthonormal.

Assume S is a basis (ie., lin. indep.).

$\Rightarrow \dim(\text{Span}(S)) = k$

Gaussian elimination



keep going

Same idea! $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k\} \longrightarrow \{\vec{v}_1, P^{\perp}_{v_1} v_2, \ldots, P^{\perp}_{v_1, \ldots, v_{k-1}} v_k\}$ keep

Same idea! $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k\} \longrightarrow \{\vec{v}_1, P^{\perp}_{v_1}v_2, \ldots, P^{\perp}_{v_1}v_k\} \xrightarrow{\text{keep going}}$

First fix $\vec{v}_2, \ldots, \vec{v}_k$ to be $\perp$ to $\vec{v}_1$

Then fix vectors 3-k to be $\perp$ to vector 2

⋮ etc.

Gram-Schmidt procedure (inputs $\vec{v}_1, \ldots, \vec{v}_k$)

1. Project $\vec{v}_2, \ldots, \vec{v}_k$ to be orthogonal to $\vec{v}_1$, using $P^{\perp}_{v_1} = I - P_{v_1} = I - \frac{\vec{v}_1 \vec{v}_1^T}{\|v_1\|^2}$.
2. Now $P^{\perp}_{v_1}v_2, \ldots, P^{\perp}_{v_1}v_k$ span a $(k-1)$-dim subspace of Span(S), orthogonal to $\vec{v}_1$.
   Recurse to find an orthogonal basis for it.
   ie. $P^{\perp}_{v_1}v_3 \longmapsto P^{\perp}_{P^{\perp}_{v_1}v_2}(P^{\perp}_{v_1}v_3)$, etc.
3. Renormalize the vectors (divide by their lengths)

Example: Find an orthonormal basis for the span of

$(1,0,0,-1)$, $(1,2,0,-1)$, $(3,1,1,-1)$.

Answer: v_1 = v_1, v_2 = v_2, v_3 = v_3

$v_1 \xrightarrow{\text{normalize}} v_1' = \frac{v_1}{\|v_1\|} = \frac{1}{\sqrt{2}}(1,0,0,-1)$

$v_2 \xrightarrow{\perp v_1} v_2 - (v_1' \cdot v_2)v_1' = (1,2,0,-1) - (1,0,0,-1)$

$= (0,2,0,0)$

$\xrightarrow{\text{normalize}} v_2' = (0,1,0,0)$

$v_3 \xrightarrow{\perp v_1} v_3 - (v_1' \cdot v_3)v_1' = (3,1,1,-1) - \frac{1}{2}(v_1 \cdot v_3)v_1$

$= (3,1,1,-1) - 2(1,0,0,-1)$

$v_3' = (1,1,1,1)$

$\xrightarrow{\perp v_2'} v_3' - (v_2' \cdot v_3')v_2'$

$= (1, 0, 1, 1)$

normalize $\rightarrow \frac{1}{\sqrt{3}}(1, 0, 1, 1)$

$\Rightarrow \left\{ \frac{1}{\sqrt{2}}(1,0,0,-1), (0,1,0,0), \frac{1}{\sqrt{3}}(1,0,1,1) \right\}$

Sanity check: Why $v_2 \longrightarrow v_2 - \frac{(v_1 \cdot v_2)}{\|v_1\|^2} v_1$?

1. $v_1 \cdot \left(v_2 - \frac{(v_1 \cdot v_2)}{\|v_1\|^2} v_1 \right) = v_1 \cdot v_2 - \frac{(v_1 \cdot v_2)}{\|v_1\|^2}(v_1 \cdot v_1)$

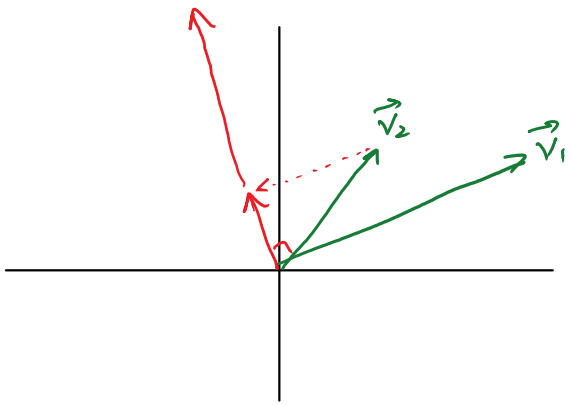
$= 0$

result is $\perp$ to v_1 ✓

2. $\text{Span}\{v_1, v_2\} = \text{Span}\{v_1, v_2 - \alpha v_1\}$ ✓

linear combination leave spanned space alone

(same as Gaussian Elim. preserves rowspace)



Observe: Two natural orders for Gram-Schmidt:

① For vector i from 1 to $k-1$:
- fix all subsequent vectors to be $\perp$ to vector i

② For vector i from 2 to k:
- project vector i to be $\perp$ to all preceding vectors

these are equivalent! (but method ① is more numerically stable)

Exercise: In Matlab/Octave/Mathematica, perform Gram-Schmidt on 5 random vectors in $\mathbb{R}^{10}$. Verify that the order doesn't matter. (Or does it?)

```
n = 10;
d = 5;
% choose d random vectors in R^n, with normally distributed coordinates
vectors = randn(n, d)
```

(2)
```
A = vectors;
for i = 1:d
 for j = 1:i-1
  A(:,i) = A(:,i) - (A(:,j)'*A(:,i)) * A(:,j);
 end
 A(:,i) = A(:,i) / sqrt(A(:,i)'*A(:,i));
end
```
} project ith col. to be ⊥ to previous columns

(1)
```
B = vectors;
for i = 1:d
 B(:,i) = B(:,i) / sqrt(B(:,i)'*B(:,i));
 for j = i+1:d
  B(:,j) = B(:,j) - (B(:,i)'*B(:,j)) * B(:,i);
 end
end
```
project subsequent columns to be ⊥ to column i

Check the answer:

```
A'*A
```

```
ans =

   1.0000e+00   4.8572e-17   3.1225e-17  -9.7145e-17   8.3267e-17
   4.8572e-17   1.0000e+00   4.8572e-17  -7.6328e-17  -5.5511e-17
   3.1225e-17   4.8572e-17   1.0000e+00   7.8063e-17  -1.3878e-17
  -9.7145e-17  -7.6328e-17   7.8063e-17   1.0000e+00   3.4694e-17
   8.3267e-17  -5.5511e-17  -1.3878e-17   3.4694e-17   1.0000e+00
```

```
sum(sum(abs(A-B)))
```
→ 0 ✓

Observe: On m vectors in $\mathbb{R}^n$, running time of G-S. is $\boxed{O(m^2 n)}$ ($=O(n^3)$ if $m=n$)

Question: What happens if we don't start with a linearly independent set of vectors?

Answer: It still works! Just some vectors will be zeroed out.
$\Rightarrow$ No need to run Gaussian elimination first.

Example: Gram-Schmidt on

$$\{(1,0), (1,-2), (0,1)\}$$

project Y to $(1,0)$:

$(1,-2) \longrightarrow (1,-2) - ((1,0)\cdot(1,-2))(1,0)$
$= (0,-2) \longrightarrow (0,-1)$ renormalizing

$(0,1) \longrightarrow (0,1) - ((1,0)\cdot(0,1))(1,0)$
$= (0,1)$

project Y to $(0,-1)$:

$(0,1) \longrightarrow (0,1) - ((0,-1)\cdot(0,1))(0,-1)$

project $\vec{v}$ to $(0,-1)$:

$(0,1) \longrightarrow (0,1) - \underbrace{((0,-1)\cdot(0,1))}_{-1}(0,-1)$

$= (0,0)$

$\Rightarrow$ $\boxed{\{(1,0),(0,-1)\}}$ an orthonormal basis

(just be careful about dividing by 0!!)

Recall: LU-decomposition

$$A = \begin{pmatrix}\text{permutation}\\ \text{of}\\ \text{rows}\end{pmatrix} \cdot \begin{pmatrix}\text{lower triangular}\end{pmatrix} \cdot \begin{pmatrix}\text{upper triangular}\\ U\end{pmatrix}$$

(inverse) history of G. elim.

result of Gaussian elimination

Gaussian elimination $O(n^3)$ steps to solve n equations in n unknowns

- But if you store the LU decomposition, then further equations can each be solved in $O(n^2)$ steps by back-substitution (for L and for U).

Main idea: Solving a lower or upper triangular system of equations is much faster than solving a general system: $O(n^2)$ versus $O(n^3)$.

Similarly...

It is easy to solve $Ax = b$

when the columns of A are orthonormal!

$$\left(\begin{array}{c|c|c|c} & & & \\ \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \\ & & & \end{array}\right)\begin{pmatrix}\\ \vec{x} \\ \\ \end{pmatrix} = \begin{pmatrix}\\ \vec{b} \\ \\ \end{pmatrix}$$

$$\Leftrightarrow x_1\vec{u}_1 + x_2\vec{u}_2 + \cdots + x_n\vec{u}_n = \vec{b}$$

$\Rightarrow x_1 = \vec{a}_1 \cdot \vec{b}$, $x_2 = \vec{a}_2 \cdot \vec{b}$, ... $O(n^2)$ steps

QR decomposition: Any $m \times n$ matrix A with linearly independent columns can be factored as

$$A_{m \times n} = \underset{m}{\underset{n}{Q}} \cdot \underset{n}{\underset{n}{R}}$$

Q: orthonormal columns spanning $R(A)$

R: upper bar history of Gram-Schmidt process

This gives another way of amortizing the cost of solving

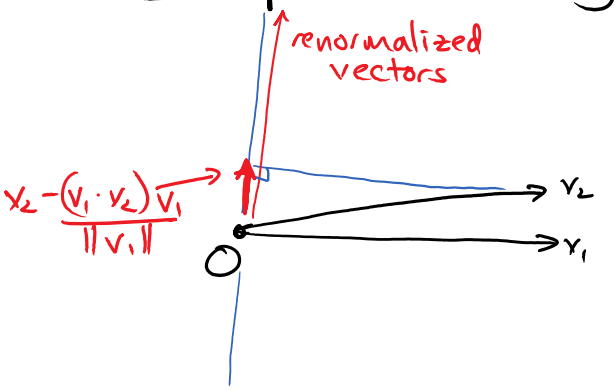
$Ax = b_1$, $Ax = b_2$, $Ax = b_3$, $Ax = b_4$, ...

Run G.S. once $O(n^3)$, then $O(n^2)$ for each equation.

DON'T DO THIS!

Gram-Schmidt gets ugly fast.

Example: Instability



If a vector v_j is close to $\text{Span}\{v_1, \ldots, v_{j-1}\}$, then renormalization will blow up the length a lot, amplifying errors!

(QR decomposition can still be useful (see example 5.5.3 for using it to find x minimizing $\|Ax - b\|$). Numerically, the "Householder method" is slightly better than naive Gram-Schmidt; that's what Matlab's qr(·) function uses.)