Lecture 12: Gram-Schmidt orthogonalization (class)

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 $\mathcal{U} \subseteq \mathbb{R}^n$ with orthonormal basis $\{\vec{u}_1, \ldots, \vec{u}_k\}$, orthogonal projection onto $\mathcal{U}$

$$P_{\mathcal{U}} = \sum_{j=1}^{k} \vec{u}_j \vec{u}_j^T$$

If $\{u_1, \ldots, u_k\}$ is o.n.

$\mathcal{U} = \mathrm{Span}\{u_1, \ldots, u_k\}$

any $\vec{v} \in \mathcal{U}$ can be written

$$\vec{v} = \sum_{j=1}^{k} (u_j \cdot v) \vec{u}_j$$

any $\vec{v}$,

$$\mathrm{Proj}_{\mathcal{U}} \vec{v} = \sum_{j=1}^{k} (u_j \cdot v) \vec{u}_j = \left(\sum_{j=1}^{k} u_j \vec{u}_j^T \right) \vec{v}$$

$$\Rightarrow \mathrm{Proj}_{\mathcal{U}} = \sum_{j=1}^{k} u_j \vec{u}_j^T$$

Example: (coordinate expansion)

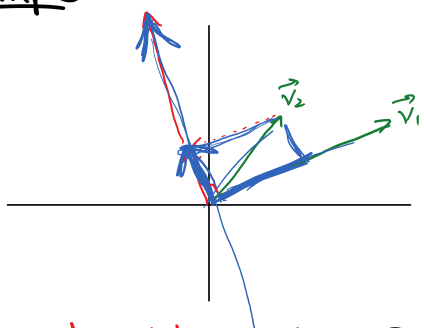
$$\vec{v} = P_{\mathbb{R}^n} \vec{v} = \sum_{j=1}^{n} \vec{u}_j \underbrace{\vec{u}_j^T \vec{v}}_{\vec{u}_j \cdot \vec{v}}$$

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

Today:

HOW TO GET AN ORTHONORMAL BASIS

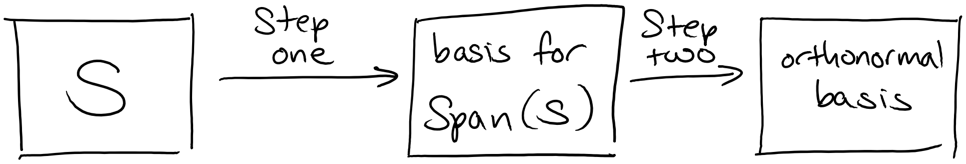
Example:



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).

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

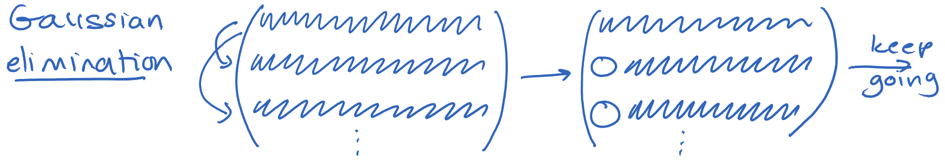


Step One: Find *any* basis for Span(S).
(because the vectors might be linearly dependent)

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

Gaussian elimination on A leaves a basis.

Step Two: Adjust the basis to be orthonormal.
Assume S is a basis (ie, lin. indep.)
$\Rightarrow \dim(\mathrm{Span}(S)) = k$



Same idea! $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_k\} \longrightarrow$

```
Gram-Schmidt1($\vec{v}_1, \dots, \vec{v}_k$)
    for $i = 1$ to $k$
        $v_i \leftarrow v_i / \|v_i\|$
        for $j = i+1$ to $k$
            $v_j \leftarrow v_j - (v_i \cdot v_j)\hat{v}_i$
```

overall time $O(k^2 \cdot \text{dimension } n)$

project v_j $\perp$ to v_i

```
Gram-Schmidt2($\vec{v}_1, \dots, \vec{v}_k$)
    for $j = 1$ to $k$
        for $i = 1$ to $j-1$
            $v_j \leftarrow v_j - (v_i \cdot v_j)\hat{v}_i$
        $v_j \leftarrow v_j / \|v_j\|$
```

project v_j $\perp$ to v_i

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

1. Project $\vec{v}_2, \dots, \vec{v}_k$ to be orthogonal to $\vec{v}_1$, using $P_{v_1}^{\perp} = I - P_{v_1} = I - \frac{\vec{v}_1 \vec{v}_1^T}{\|v_1\|^2}$.

2. Now $P_{v_1}^{\perp} v_2, \dots, P_{v_1}^{\perp} v_k$ span a $(k-1)$-dim subspace of $\mathrm{Span}(S)$, orthogonal to $\vec{v}_1$.
   Recurse to find an orthogonal basis for it.
   ie. $P_{v_1}^{\perp} v_3 \longmapsto P_{P_{v_1}^{\perp} v_2}^{\perp}(P_{v_1}^{\perp} v_3)$, etc.

3. Renormalize the vectors (divide by their lengths)

Example: Find an orthonormal basis for the span of

$$\underset{\overset{\parallel}{v_1 = e_1 - e_4}}{(1,0,0,-1)}, \quad \underset{\overset{\parallel}{v_2}}{(1,2,0,-1)}, \quad \underset{\overset{\parallel}{v_3}}{(3,1,1,-1)}.$$

projected $\perp$ to $v_2 = e_2$

Answer:

$$\vec{v}_1' = \frac{1}{\sqrt{2}}(1,0,0,-1)$$

$$= \frac{e_1 - e_4}{\sqrt{2}} \qquad e_2 \qquad e_3$$

$$\frac{e_1 + e_4}{\sqrt{2}}$$

$$v_2' = v_2 - (v_1' \cdot v_2)v_1'$$

$$= v_2 - \frac{1}{2}\left[\begin{pmatrix}1\\0\\0\\-1\end{pmatrix} \cdot \begin{pmatrix}1\\2\\0\\-1\end{pmatrix}\right]\begin{pmatrix}1\\0\\0\\-1\end{pmatrix}$$

$$= v_2 - v_1$$

$$= \begin{pmatrix}0\\2\\0\\0\end{pmatrix}$$

$$v_2'' = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$v_3' = (3,0,1,-1)$$

$$v_3'' = v_3' - \frac{1}{2}\underset{\overset{\parallel}{4}}{\left[\begin{pmatrix}1\\0\\0\\-1\end{pmatrix} \cdot \begin{pmatrix}3\\0\\1\\-1\end{pmatrix}\right]}\begin{pmatrix}1\\0\\0\\-1\end{pmatrix}$$

$$= \begin{pmatrix}3\\0\\1\\-1\end{pmatrix} - \begin{pmatrix}2\\0\\0\\-2\end{pmatrix} = \begin{pmatrix}1\\0\\1\\1\end{pmatrix}$$

$$v_3''' = \frac{1}{\sqrt{3}}\begin{pmatrix}1\\0\\1\\1\end{pmatrix}$$

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

Sanity check: Why $v_2 \longrightarrow v_2 - \frac{(v_1 \cdot v_2)v_1}{\|v_1\|^2} = \left(I - \overset{P_{\mathrm{span}\{v_1\}}}{\overset{\parallel}{\frac{v_1 v_1^T}{\|v_1\|^2}}}\right) v_2$

$\parallel$

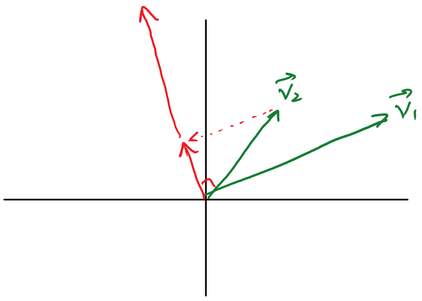
$P_{\mathrm{Span}\{v_1\}^\perp}$

$$v_1 \cdot \left(v_2 - \frac{(v_1 \cdot v_2)\, v_1}{\|v_1\|^2} \right) = v_1 \cdot v_2 - \frac{(v_1 \cdot v_2)}{\cancel{\|v_1\|^2}} \cancel{v_1 \cdot v_1} = 0$$

result is $\perp$ to x_1 ✓

2. $\mathrm{Span}\{x_1, v_2\} = \mathrm{Span}\{x_1, v_2 - \alpha v_1\}$ ✓

linear combination leave spanned space alone (same as Gaussian Elim. preserxes 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 $\perp$ 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 $\perp$ to column i

Check the answer:

```
A'*A
```

```
sum(sum(abs(A-B)))
```

$\rightarrow 0$ ✓

```
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
```

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 ⊥ to $(1,0)$

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

$$= (0,-2) \longrightarrow (0,-1) \quad \text{renormalizing}$$

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

$$= (0,1)$$

project ⊥ to $(0,-1)$

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

$$= (0,0)$$

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

(just be careful about dividing by 0!!)

Recall: LU-decomposition

$$A = \begin{pmatrix}\text{permutation}\\ \text{of}\\ \text{rows}\end{pmatrix} \cdot \underset{\substack{\text{(inverse) history}\\ \text{of G. elim.}}}{\begin{pmatrix} L \\ \text{lower triangular}\end{pmatrix}} \cdot \underset{\substack{\text{result of}\\ \text{Gaussian elimination}}}{\begin{pmatrix}\text{upper }\Delta\text{ar} \\ U\end{pmatrix}}$$

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!

$$\begin{pmatrix} | & | & & | \\ \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} \\ \vec{x} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \vec{b} \\ \\ \end{pmatrix}$$

"Trick"
dot both sides with $\vec{u}_j$

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

$$\Rightarrow x_1 = \vec{u}_1 \cdot \vec{b},\ x_2 = \vec{u}_2 \cdot \vec{b},\ \ldots \qquad O(n^2) \text{ steps}$$

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

$$A_{m\times n} = {}_m\underbrace{\overset{n}{\begin{pmatrix} | & | & | & | \\ & Q & & \\ | & | & | & | \end{pmatrix}}}_{\text{orthonormal columns spanning } R(A)} \cdot {}_n\underbrace{\overset{n}{\begin{pmatrix} & & R \\ & & \\ & & \end{pmatrix}}}_{\text{upper }\Delta\text{ar history of Gram-Schmidt process}}$$

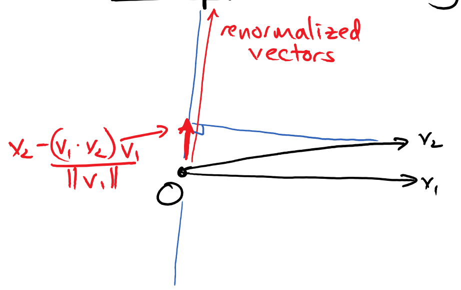
This gives another way of amortizing the cost of solving

$$Ax = b_1,\ Ax = b_2,\ Ax = b_3,\ Ax = b_4, \ldots$$

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 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($\cdot$) function uses.)