Overview

- Linear projection onto a subspace
- QR factorization
- Stabilized Gram-Schmidt algorithm

Science = Data Reduction

- Science goal: describe regularity in data, e.g., $a^2 + b^2 = c^2$, instead of

		(7, 24, 25)
	(5, 12, 13)	(55, 48, 73)
		(45, 28, 53)
		(39, 80, 89)
(3, 4, 5)	(21, 20, 29)	(119, 120, 169).
		(77, 36, 85)
		(33, 56, 65)
	(15, 8, 17)	(65, 72, 97)
		(35, 12, 37)

- Linear data reduction: projection from $\mathbb{C}^m$ to $\mathbb{C}^n$, $n < m$

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{P}\boldsymbol{v}, \boldsymbol{A} \in \mathbb{C}^{m \times n}, \boldsymbol{P} \in \mathbb{C}^{m \times m}$$

$\boldsymbol{x}$ coordinates of $\boldsymbol{b}$ in basis $\boldsymbol{A}$, $\mathrm{rank}(\boldsymbol{A}) = n$. Let $\boldsymbol{b}$ be the linear projection onto $C(\boldsymbol{A}) = C(\boldsymbol{P})$ of $\boldsymbol{v}$, $\boldsymbol{b} = \boldsymbol{P}\boldsymbol{v}$

- A *projector* is idempotent: $\boldsymbol{P}^2 = \boldsymbol{P}$, $\boldsymbol{P} \in \mathbb{C}^{m \times m}$

- $\boldsymbol{P}$ a projector, $\boldsymbol{I} - \boldsymbol{P}$ is the *complementary projector*

$$C(\boldsymbol{I}-\boldsymbol{P}) = N(\boldsymbol{P}),\ \ C(\boldsymbol{P}) = N(\boldsymbol{I}-\boldsymbol{P}),\ \ C(\boldsymbol{P}) \cap N(\boldsymbol{P}) = \{\boldsymbol{0}\}$$

- $\boldsymbol{P}$ is an *orthogonal projector* if $C(\boldsymbol{P}) \perp N(\boldsymbol{P})$, but $\boldsymbol{P}$ is not necessarily an orthogonal matrix
- $\boldsymbol{P}$ is an orthogonal projector iff $\boldsymbol{P} = \boldsymbol{P}^*$

- Consider $\hat{\boldsymbol{Q}} \in \mathbb{C}^{m \times n}$ with orthonormal columns
- $\boldsymbol{P} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{Q}}^*$ is an orthogonal projector and $\boldsymbol{P}$ is orthogonal
- Compare with the much more complicated expression for general $\boldsymbol{A} \in \mathbb{C}^{m \times n}$

$$\boldsymbol{P} = \boldsymbol{A}(\boldsymbol{A}^*\boldsymbol{A})^{-1}\boldsymbol{A}^*$$

  to appreciate the benefits of orthonormal bases, $C(\boldsymbol{A}) = C(\hat{\boldsymbol{Q}})$
- Seek orthonormal basis for $C(\boldsymbol{A})$

$$\boldsymbol{A} = [\ \boldsymbol{a}_1 \ \ \boldsymbol{a}_2 \ \ \dots \ \ \boldsymbol{a}_n\ ] = \hat{\boldsymbol{Q}}\,\boldsymbol{R} = [\ \boldsymbol{q}_1 \ \ \boldsymbol{q}_2 \ \ \dots \ \ \boldsymbol{q}_n\ ] \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & r_{nn} \end{bmatrix}$$

Algorithm (Gram-Schmidt)

Given n vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$
for $j = 1$ to n
  $\boldsymbol{v} = \boldsymbol{a}_j$
  for $i = 1$ to $j - 1$
    $r_{ij} = \boldsymbol{q}_i^* \boldsymbol{a}_j$
    $\boldsymbol{v} = \boldsymbol{v} - r_{ij}\boldsymbol{q}_i$
  end
  $r_{jj} = \|\boldsymbol{v}\|_2$
  if $r_{jj} < \epsilon$ break;
  $\boldsymbol{q}_j = \boldsymbol{q}_j / r_{jj}$
end
return $\boldsymbol{Q}, \boldsymbol{R}$

```
octave> function [Q,R] = gs(A)
          [m,n]=size(A); Q=A; R=eye(n);
          for j=1:n
            v=A(:,j);
            for i=1:j-1
              R(i,j)=Q(:,i)'*A(:,j);
              v=v-R(i,j)*Q(:,i);
            end;
            R(j,j)=norm(v);
            if (R(j,j)<eps) break; endif;
            Q(:,j)=Q(:,j)/R(j,j);
          end;
        end
octave]
```

Algorithm (Gram-Schmidt)

Given n vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$
Initialize $\boldsymbol{q}_1 = \boldsymbol{a}_1$,..,
$\boldsymbol{q}_n = \boldsymbol{a}_n$,
$\boldsymbol{R} = \boldsymbol{I}_n$
for $i = 1$ to n
  $r_{ii} = (\boldsymbol{q}_i^T \boldsymbol{q}_i)^{1/2}$
  if $r_{ii} < \epsilon$ break;
  $\boldsymbol{q}_i = \boldsymbol{q}_i / r_{ii}$
  for $j = i+1$ to n
    $r_{ij} = \boldsymbol{q}_i^T \boldsymbol{a}_j$
    $\boldsymbol{q}_j = \boldsymbol{q}_j - r_{ij}\boldsymbol{q}_i$
  end
end
return $\boldsymbol{Q}, \boldsymbol{R}$

```
octave> function [Q,R] = mgs(A)
          [m,n]=size(A); Q=A; R=eye(n);
          for i=1:n
            R(i,i)=norm(Q(:,i));
            if (R(i,i)<eps) break; endif;
            Q(:,i)=Q(:,i)/R(i,i);
            for j=i+1:n
              R(i,j)=Q(:,i)'*A(:,j);
              Q(:,j)=Q(:,j)-R(i,j)*Q(:,i);
            end;
          end;
        end
octave]
```