Orthonormal Vectors

- A set $S$ of nonzero vectors are *orthonormal* if, for every $x$ and $y$ in $S$, we have $\text{dot}(x,y)=0$ (orthogonality) and for every $x$ in $S$ we have $\|x\|_2=1$ (length is 1).
The QR Algorithm

• The QR algorithm for finding eigenvalues is based on the QR factorisation we learnt in the least squares part of the course

• Recall the QR factorization represents a matrix $A$ as:
  \[ A = QR \]
  where $Q$ is a matrix whose columns are orthonormal, and $R$ is an upper triangular matrix.

• Recall that $Q^H Q = I$ and $Q^{-1} = Q^H$.

• $Q$ is termed a unitary matrix.
QR Algorithm without Shifts

\[ A_0 = A \]

for \( k = 1, 2, \ldots \)

\[ Q_k R_k = A_k \]

\[ A_{k+1} = R_k Q_k \]

end

Since:

\[ A_{k+1} = R_k Q_k = Q_k^{-1} A_k Q_k \]

then \( A_k \) and \( A_{k+1} \) are similar and so have the same eigenvalues.

\( A_{k+1} \) tends to an upper triangular matrix with the same eigenvalues as \( A \). These eigenvalues lie along the main diagonal of \( A_{k+1} \).
QR Algorithm with Shift

\[
A_0 = A \\
\text{for } k=1,2,\ldots \\
s = A_k(n,n) \\
Q_kR_k = A_k - sI \\
A_{k+1} = R_kQ_k + sI \\
\text{end}
\]

Since:
\[
A_{k+1} = R_kQ_k + sI \\
= Q_k^{-1}(A_k - sI)Q_k + sI \\
= Q_k^{-1}A_kQ_k
\]

so once again \( A_k \) and \( A_{k+1} \) are similar and so have the same eigenvalues.

The shift operation subtracts \( s \) from each eigenvalue of \( A \), and speeds up convergence.
MATLAB Code for QR Algorithm

• Let A be an \( n \times n \) matrix

\[
\begin{align*}
  n &= \text{size}(A,1); \\
  I &= \text{eye}(n,n); \\
  s &= A(n,n); \quad [Q,R] = \text{qr}(A-s*I); \quad A = R*Q+s*I
\end{align*}
\]

• Use the up arrow key in MATLAB to iterate or put a loop round the last line.
Deflation

• The eigenvalue at $A(n,n)$ will converge first.
• Then we set $s=A(n-1,n-1)$ and continue the iteration until the eigenvalue at $A(n-1,n-1)$ converges.
• Then set $s=A(n-2,n-2)$ and continue the iteration until the eigenvalue at $A(n-2,n-2)$ converges, and so on.
• This process is called *deflation*.
Computational Methods
CMSC/AMSC/MAPL 460

EigenValue decomposition
Singular Value Decomposition

Ramani Duraiswami,
Dept. of Computer Science
Eigenvalue sensitivity

\[ A = XX^{-1} \]

\[ \Lambda = X^{-1}AX \]

\[ \Lambda + \delta \Lambda = X^{-1}(A + \delta A)X \]

\[ \delta \Lambda = X^{-1}\delta AX \]

\[ ||\delta \Lambda|| \leq ||X^{-1}||||X||||\delta A|| = \kappa(X)||\delta A|| \]

The sensitivity of the eigenvalues is estimated by the condition number of the matrix of eigenvectors.
The SVD

• **Definition:** Every matrix $A$ of dimensions $m \geq n$ can be decomposed as

$$A = U \Sigma V^*$$

• where
  - $U$ has dimension $m \times m$ and $U^*U = I$,
  - $\Sigma$ has dimension $m \times n$,
    the only nonzeros are on the main diagonal, and they are nonnegative real numbers $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$,
  - $V$ has dimension $n \times n$ and $V^*V = I$. 
Relation with the Eigenvalue Decomposition

• Let $A = U \Sigma V^*$. Then

$$A^* A = (U \Sigma V^*)^* U \Sigma V^*$$

$$= V \Sigma^* U^* U \Sigma V^* = V \Sigma^2 V^*$$

• This tells us that the singular value decomposition of $A$ is related to the Eigenvalue decomposition of $A^* A$

• Recall eigen value decomposition $A = (X \Lambda X^*)$
  – So $V$ which contains the right singular vectors of $A$ has the right eigenvectors of $A^* A$
  - $\Sigma^2$ are the eigenvalues of $A^* A$
  - The **singular values** $\sigma_i$ of $A$ are the square roots of the **eigenvalues** of $A^* A$. 
Relation with the Eigenvalue Decomposition (2)

• Let \( A = U \Sigma V \). Then

\[
A A^* = (U \Sigma V^*) (U \Sigma V^*)^*
= U \Sigma V^* V \Sigma^* U^* = U \Sigma^2 U^*
\]

• This tells us that the singular value decomposition of \( A \) is related to the Eigenvalue decomposition of \( AA^* \)

• Recall eigen value decomposition \( A = (X \Lambda X^*) \)
  
  – So \( U \) contains the the **left singular vectors** of \( A \), which are also the left eigenvectors of \( AA^* \)

  □ \( \Sigma^2 \) are the eigenvalues of \( AA^* \) and the **singular values** \( \sigma_i \) of \( A \) are the square roots of the **eigenvalues** of \( AA^* \)
Economy-sized SVD

\[
A = \begin{bmatrix} U & \Sigma & V' \end{bmatrix}
\]

\[
A = \begin{bmatrix} U & \Sigma & V' \end{bmatrix}
\]
Computing the SVD

• The algorithm is a variant on algorithms for computing eigendecompositions.
  – rather complicated, so better to use a high-quality existing code rather than writing your own.
• In Matlab: \([U,S,V] = svd(A)\)
• The cost is \(O(mn^2)\) when \(m \geq n\). The constant is of order 10.
Uses of the SVD

• Recall to solve least squares problems we could look at the normal equations \( (A^*Ax = A^*b) \)
  – So, SVD is closely related to solution of least-squares
  – Used for solving ill conditioned least-squares
• Used for creating low-rank approximations
• Both applications are related
SVD and reduced rank approximation

- \(Ax=b\) \(A\) is \(m\times n\), \(x\) is \(n\times l\) and \(b\) is \(m\times l\).
- \(A=USV^t\) where \(U\) is \(m\times m\), \(S\) is \(m\times n\) and \(V\) is \(n\times n\)
- \(USV^t x=b\). So \(SV^t x=U^t b\)
- If \(A\) has rank \(r\), then \(r\) singular values are significant
  \(V^tx=\text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)U^t b\)
  \(x=V\text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0)U^t b\)
  \[
x_r = \sum_{i=1}^r \frac{u_i^tb}{\sigma_i} v_i \quad \sigma_r > \epsilon, \quad \sigma_{r+1} \leq \epsilon
  \]
- We can truncate \(r\) at any value and achieve “reduced-rank” approximation to the matrix
- For ordered signular values, this gives the “best reduced rank approximation”
SVD and pseudo inverse

• Pseudoinverse $A^+=V \text{ diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0) U^t$
  
  – $A^+$ is a $n \times m$ matrix.
  
  – If rank $(A) = n$ then $A^+ = (A^tA)^{-1}A$
  
  – If $A$ is square $A^+ = A^{-1}$
Well posed problems

- Hadamard postulated that for a problem to be “well posed”
  1. Solution must exist
  2. It must be unique
  3. Small changes to input data should cause small changes to solution
- Many problems in science and computer vision result in “ill-posed” problems.
  - Numerically it is common to have condition 3 violated.
  - Recall from the SVD

\[
x = \sum_{i=1}^{n} \left( \frac{u_i^T b}{\sigma_i} \right) v_i \quad \sigma_r > \varepsilon, \quad \sigma_{r+1} \leq \varepsilon
\]

If the \( \sigma \) are close to zero small changes in the “data” vector \( b \) cause big changes in \( x \).
- Converting ill-posed problem to well-posed one is called \textit{regularization}. 
SVD and Regularization

• Pseudoinverse provides one means of regularization

• Another is to solve \((A + \varepsilon I)x = b\) 

\[ x = \sum_{i=1}^{n} \frac{\sigma_i}{\varepsilon + \sigma_i^2} (u_i^T b) v_i \]

• Solution of the regular problem requires minimizing of \(\|Ax - b\|^2\)

• Solving this modified problem corresponds to minimizing 

\[ \|Ax - b\|^2 + \varepsilon \|x\|^2 \]

• Philosophy – pay a “penalty” of \(O(\varepsilon)\) to ensure solution does not blow up.

• In practice we may know that the data has an uncertainty of a certain magnitude … so it makes sense to optimize with this constraint.

• Ill-posed problems are also called “ill-conditioned”