Holger Broman, August 19, 1999

Factorization

Why do we bother with the problem of how to factorize a matrix? Well, you have seen glimpses of answers already. One glimpse is from the Gaussian elimination scheme: when you solve a system of linear equations in a systematic way, a general matrix is transformed into a triangular one. A second viewpoint of the same problem is as follows. Given a full rank, square, matrix A, the solution to

$$A\mathbf{x} = \mathbf{b}$$

is

$$\mathbf{x} = A^{-1}\mathbf{b}.$$

However, A may be tough to invert, and a factorization

$$A = BC$$

can help if both B and C are simple to invert.

One general answer is thus: if you can split an unstructured matrix into factors that do have special structures, this factorization may be utterly useful.

Factorization is so important that we have chosen to collect four of the most interesting schemes in one 'chapter'. Here they are, all four:

- *LU* or *LDU* factorization.
  Stems from Gaussian elimination and results in (for square matrices A)

$$A = LU = LDU'$$

$$L = \begin{bmatrix} 1 & & & \\ x & \ddots & 0 & \\ \vdots & & \ddots & \\ x & \cdots & x & 1 \end{bmatrix} \qquad U = \begin{bmatrix} x & \cdots & \cdots & x \\ & \ddots & & \\ 0 & & \ddots & \\ & & & x \end{bmatrix}$$

$$D = \mathrm{diag}(d_1 \cdots d_N) \qquad U' = \begin{bmatrix} 1 & x & \cdots & x \\ & \ddots & & \vdots \\ 0 & & \ddots & x \\ & & & 1 \end{bmatrix}$$

- QR factorization.
  Stems from orthogonalization and results in

$$A = QR$$

$$Q^T Q = I, \text{ i.e. } Q \text{ has orthonormal columns,}$$

$$R = \begin{bmatrix} x & \cdots & \cdots & x \\ & \ddots & & \vdots \\ 0 & & \ddots & \vdots \\ & & & x \end{bmatrix}$$

  R is upper right triangular, and x stands for any possible non-zero entry.

- Eigenvalue decomposition.
  Only for square matrices with a basis of eigenvectors:

$$A = G^{-1} \Lambda \, G$$

$$G = (\mathbf{g}_1 \cdots \mathbf{g}_N)\,, \; \mathbf{g}_n \text{ is an eigenvector}$$

$$\Lambda = \mathrm{diag}\,(\lambda_1 \cdots \lambda_N)\,, \; \lambda_n \text{ is an eigenvalue}$$

- Singular Value Decomposition, SVD for short.
  The most general decomposition, need not square matrices A:

$$A = Q_1 \; \Sigma \; Q_2^T$$

$$Q_1 \;\; \text{orthogonal, square}$$

$$Q_2 \;\; \text{orthogonal, square}$$

$$\Sigma \;\; \text{diagonal}$$

LU factorization

One way of factorizing a (square) matrix A follows from solving a linear system of equations by Gaussian elimination. Let us start by an example:

$$\begin{cases} x + 2y = 4 \\ 3x + 4y = 10 \end{cases} \qquad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 10 \end{bmatrix}$$

$$\Longleftrightarrow$$

$$\begin{cases} x + 2y = 4 \\ -2y = -2 \end{cases} \qquad \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \end{bmatrix}$$

Now, search for a matrix L so that

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = L \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}.$$

The answer is

$$L = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}, \text{ so that}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}.$$

In general terms, a square matrix A can be factorized

$$A = LU,$$

where L is lower left triangular with unity on the main diagonal, and U is upper right triangular if the Gaussian elimination scheme works.

Here is an example where it works:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -11 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & 1 \end{bmatrix} = U$$

Here is an example where it does not work:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 4 \\ 5 & 6 & 7 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 1 \\ 0 & -4 & -8 \end{bmatrix}$$

Oops, no usable pivot element in the second row. This can easily be fixed by considering a row-shifted matrix – system of equations. Note that the equation

$$A\mathbf{x} = \mathbf{b}$$

$\Longleftrightarrow$

$$\begin{bmatrix} \text{row 1} \\ \text{row 2} \\ \text{row 3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$\Longleftrightarrow$

$$\begin{bmatrix} \text{row 1} \\ \text{row 3} \\ \text{row 2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_3 \\ b_2 \end{bmatrix},$$

so that we can factorize

$$A' = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 1 & 2 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & -4 & -8 \\ 0 & 0 & 1 \end{bmatrix} = U$$

Here is a second example where it does not work – or does it?

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -12 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & 0 \end{bmatrix},$$

oops, A is not full rank.

Note 1: The staircase pattern in the matrix above is called the echelon form of a matrix. We find, however, that

$$\begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 7 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

The global conclusion is that all (full rank) square matrices can be LU factorized, possibly after one or more row exchange operations. Row exchanges are accomplished by permutation matrices – se the leaflet named Permutation matrices. This means that there exists a permutation matrix P so that

$$AP = LU$$

———— o O o ————

Sometimes you desire unity on the main diagonal of U as well. Then you perform an LDU factorization so that

$$AP = LU = LDU',$$

where D is diagonal using the diagonal elements of U, and U' is upper right triangular with unity on the main diagonal.

Note 2: $\det A = \det D$ as $\det L = \det U' = 1$.

Please run the m-file lufac in Matlab.

```
% lufac.m
% This file demonstrates LU-factorization, equivalently
% Gaussian elimination.
clear

N=4;
A=randn(4,4);
b=randn(4,1);

% solve Ax=b
x=A\b;

% now do it by explicit row operations
A1=A;
b1=b;
pivfac1=A1(2,1)/A1(1,1); % Pivot is A(1,1)
A1(2,:)=A1(2,:)-pivfac1*A1(1,:);
b1(2)=b1(2)-pivfac1*b1(1);
% norm(x-A1\b1,'fro') % check that we have the same solution
pivfac2=A1(3,1)/A1(1,1);
A1(3,:)=A1(3,:)-pivfac2*A1(1,:);
b1(3)=b1(3)-pivfac2*b1(1);
% norm(x-A1\b1,'fro') % check that we have the same solution
pivfac3=A1(4,1)/A1(1,1);
A1(4,:)=A1(4,:)-pivfac3*A1(1,:);
b1(4)=b1(4)-pivfac3*b1(1);
% norm(x-A1\b1,'fro') % check that we have the same solution
pivfac4=A1(3,2)/A1(2,2); % Pivot is A(2,2)
A1(3,:)=A1(3,:)-pivfac4*A1(2,:);
b1(3)=b1(3)-pivfac4*b1(2);
% norm(x-A1\b1,'fro') % check that we have the same solution
pivfac5=A1(4,2)/A1(2,2);
A1(4,:)=A1(4,:)-pivfac5*A1(2,:);
b1(4)=b1(4)-pivfac5*b1(2);
% norm(x-A1\b1,'fro') % check that we have the same solution
pivfac6=A1(4,3)/A1(3,3); % Pivot is A(3,3)
A1(4,:)=A1(4,:)-pivfac6*A1(3,:);
b1(4)=b1(4)-pivfac6*b1(3);
norm(x-A1\b1,'fro') % check that we have the same solution

% now A1 is U by construction, what is L? remember A=L*U
L=A*inv(A1);
% however, we can do it without inverting any matrix
L=eye(N);
L(2,1)=pivfac1;
L(3,1)=pivfac2;
L(4,1)=pivfac3;
```

```
L(3,2)=pivfac4;
L(4,2)=pivfac5;
L(4,3)=pivfac6;
L, A1, norm(A-L*A1,'fro')
pause

% Finally let's do the LDU
v=diag(A1);
D=diag(v);
U=diag(1./v)*A1; % The inversion is now scalar.
L, D, U, norm(A-L*D*U,'fro')
```

QR factorization

Recall the procedure of orthogonalizing a set of vectors $\{\mathbf{v}_m\}_{m=1}^M$ and produce the orthonormal set of vectors $\{\mathbf{u}_m\}_{m=1}^M$ that span the same subspace.

As was shown, you could use a procedure named after Gram and Schmidt or you could use projections. Well, regardless of which we can pose the following problem. Stack the vectors in matrices

$$A = (\mathbf{v}_1 \cdots \mathbf{v}_M)\,,$$

and

$$Q = (\mathbf{u}_1 \cdots \mathbf{u}_M)\,.$$

Now, find a relation, in matrix terms, between these two matrices. The trick is to write the $\mathbf{v}$-vectors as linear combinations of the $\mathbf{u}$-vectors, or to use the following argument. By the Gram-Schmidt construction, $\{\mathbf{v}\}_1^k$ and $\{\mathbf{u}\}_1^k$ span the same subspace. Thus, $\mathbf{v}_k$ depends only on $\{\mathbf{u}_m\}_{m=1}^k$ and $\mathbf{u}_k$ only on $\{\mathbf{v}_m\}_{m=1}^k$. It follows

$$\begin{aligned}
\mathbf{v}_k = \sum_{m=1}^{k} r_m\, \mathbf{u}_m = (\mathbf{u}_1 \cdots \mathbf{u}_k) \begin{pmatrix} r_1 \\ \vdots \\ r_k \end{pmatrix} = \\
= (\mathbf{u}_1 \cdots \mathbf{u}_k\; \mathbf{u}_{k+1} \cdots \mathbf{u}_M) \begin{pmatrix} r_1 \\ \vdots \\ r_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \\
= Q \begin{pmatrix} r_1 \\ \vdots \\ r_k \\ 0 \end{pmatrix},
\end{aligned}$$

where Q has orthonormal columns, and thus $Q^T Q = I$. Thus,

$$A = (\mathbf{v}_1 \cdots \mathbf{v}_M) = Q \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ 0 & r_{22} & & \\ \vdots & & \ddots & \\ 0 & & & r_{MM} \end{pmatrix} = QR,$$

where R is square, upper right triangular. In conclusion:

> Any matrix A with linearly independent columns can be factorized into one matrix with orthonormal columns and an upper right triangular matrix that is invertible.

Exercise: Argue for the fact that R is invertible.

Why bother with QR factorization? One reason is that it is the preferred way to calculate the least squares solution of an overdetermined set of equations. Remember the equation

$$A\,\mathbf{x} = \mathbf{b},$$

where the full rank matrix A is taller than wide. The least squares solution was shown to be

$$\mathbf{x}_{LS} = \left(A^T A\right)^{-1} A^T\,\mathbf{b}.$$

However, $A^T A$ could be tough to invert or a poor choice from a numerical point of view due to the 'square' $A^T A$. Therefore, make a QR factorization of A:

$$\begin{aligned}
\mathbf{x}_{LS} &= \left(R^T Q^T\,Q\,R\right)^{-1} R^T Q^T\,\mathbf{b} = \\
&= \left(R^T R\right)^{-1} R^T Q^T\,\mathbf{b} = \\
&= R^{-1}\left(R^T\right)^{-1} R^T Q^T\,\mathbf{b} = R^{-1} Q^T\,\mathbf{b}.
\end{aligned}$$

This is much simpler and numerically sound as R is triangular.

Note 1: The QR factorization is not unique. Not only can you re-order the vectors $\{\mathbf{v}_m\}$ to produce different A matrices. For one given A matrix, of dimensions $N \times M$, $M \leq N$ as independent columns are required, you can separate the two cases

i) $\dim Q = N \times M, \quad \dim R = M \times M,$

ii) $\dim Q = N \times N, \quad \dim R = N \times M.$

The version above is case i).

———— o O o ————

QR factorization using Givens rotations – Remember the definition of the rotation matrix, and note the following properties.

$$Q = \begin{array}{cl}
\begin{bmatrix}
1 & & & & & & & & & & \\
& \ddots & & & & & & & & & \\
& & 1 & & & & & & & & \\
& & & c & & & & -s & & & \\
& & & & 1 & & & & & & \\
& & & & & \ddots & & & & & \\
& & & & & & 1 & & & & \\
& & & s & & & & c & & & \\
& & & & & & & & 1 & & \\
& & & & & & & & & \ddots & \\
& & & & & & & & & & 1
\end{bmatrix}
&
\begin{array}{l}
\\ \\ \\ \leftarrow \text{row } m \\ \\ \\ \\ \leftarrow \text{row } n \\ \\ \\ \\
\end{array}
\end{array}$$

- The multiplication QA leaves A unchanged in all but rows m and n.
- You can design Q to put a zero in one position at your choice in the product QA, for instance
$$[QA]_{nm} = s \cdot a_{mm} + c\, a_{nm},$$
choose the angle α to vary $s = \sin\alpha,\ c = \cos\alpha$. Equivalently, make the vector $(\sin\alpha\ \ \cos\alpha)^T$ perpendicular to the vector $(a_{mm}\ a_{nm})^T$.

Now, collect your vectors $\mathbf{v}_m$ in a matrix A. Let us illustrate in a 4×3 case ($N = 4,\ M = 3$).
$$A = \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}$$
Take Q_1 to produce
$$Q_1 A = \begin{bmatrix} x & x & x \\ 0 & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \qquad \text{(not all } x\text{:es are the same as before)}$$
Take Q_2 to produce
$$Q_2 Q_1 A = \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ x & x & x \end{bmatrix}$$
and
$$Q_3 Q_2 Q_1 A = \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix}$$
Now comes the trick, Q_4 does not destroy the zeros in the first column:
$$Q_4 Q_3 Q_2 Q_1 A = \begin{bmatrix} 1 & & & \\ & c & -s & \\ & s & c & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix} = \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & x & x \end{bmatrix}$$
The example is finished with Q_6:
$$Q_6 Q_5 Q_4 Q_3 Q_2 Q_1 A = \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \end{bmatrix}$$
Let
$$Q^T = Q_6 Q_5 Q_4 Q_3 Q_2 Q_1,$$

and note that

$$\left(Q^T\right)^{-1} = \left(Q^T\right)^T = Q,$$

as the rotation matrices have that property. We arrive at the QR factorization

$$A = QR.$$

This is the preferred algorithm for numerical reasons. You should use some smart algorithm when multiplying the rotation matrices.

Note 2 This scheme results in a Q of dimensions $N \times N$ and an R of dimensions $N \times M$. Note, however, that we may partition Q and R as follows:

$$Q = [Q_1 \quad Q_2], \quad R = [\begin{smallmatrix} R_1 \\ R_2 \end{smallmatrix}] = \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

so that

$$A = QR = [Q_1 \quad Q_2] \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1.$$

The Gram-Schmidt approach would have produced Q_1, R_1. The columns of Q_1 span the subspace of the vectors $\mathbf{v}_m$, or equivalently, the range space of A. The columns of Q_2 span the null space of A. The factorization $Q_1 R_1$ is called the "economy size" QR factorization.

Note 3 This is also the preferred way to calculate projection matrices (possibly unless $M << N$) as

$$\begin{aligned} P =& A(A^T A)^{-1} A^T = Q_1 R_1 \left(R_1^T Q_1^T Q_1 R_1\right)^{-1} R_1^T Q_1^T = \\ =& Q_1 R_1 \left(R_1^T R_1\right)^{-1} R_1^T Q_1^T = Q_1 Q_1^T \end{aligned}$$

———— o O o ————

Application example:

Given a data matrix X, filled with complex-valued entries, that has 50 rows and 1000 columns, you want to calculate an estimate of the covariance matrix $\hat{R}$:

$$\hat{R} \sim XX^H$$

The numerically proper way to do it is to calculate R only in

$$X^H = QR \qquad - \qquad \dim R = 50 \times 50$$

and note

$$\hat{R} \sim XX^H = (X^H)^H X^H = (R^H Q^H)(QR) = R^H R.$$

As a bonus, you get a factorized covariance matrix, which is good from a numerical point of view.

Note 4 A second application example is given in the leaflet on Least Squares.

Note 5 Another sound way to perform QR factorization is to use Householder reflections. It is left as an exercise to find the scheme by 'inverse engineering' of the code in the m-file housholder.

Please run the m-files ortho and householder in Matlab.

```
% ortho.m
% to study ways to create on-bases from any base.
clear

% generate 3 random vectors, the columns of A
A=rand(3,3);
v1=A(:,1); v2=A(:,2); v3=A(:,3);

% Let us first follow Gram and Schmidt
u1=v1/norm(v1);
u2=v2-(v2'*u1)*u1; u2=u2/norm(u2);
u3=v3-(v3'*u1)*u1-(v3'*u2)*u2; u3=u3/norm(u3);
Ugs=[u1 u2 u3];
% now, u1,u2,u3 make an orthonormal basis for the space
% equivalently, Ugs is an orthogonal matrix

% our second method is by means of projections
u1=v1;
P1=v1*inv(v1'*v1)*v1'; P1perp=eye(3)-P1; u2=P1perp*v2;
P2=[v1 v2]*inv([v1 v2]'*[v1 v2])*[v1 v2]'; P2perp=eye(3)-P2;
u3=P2perp*v3;

u1=u1/norm(u1); u2=u2/norm(u2); u3=u3/norm(u3);
Uproj=[u1 u2 u3];

% our third method is by Givens rotations
alpha1=-atan(A(2,1)/A(1,1));
c=cos(alpha1); s=sin(alpha1);
Rot1=[c -s 0 ; s c 0 ; 0 0 1]; A1=Rot1*A;
alpha2=-atan(A1(3,1)/A1(1,1));
c=cos(alpha2); s=sin(alpha2);
Rot2=[c 0 -s ; 0 1 0 ; s 0 c]; A2=Rot2*A1;
alpha3=-atan(A2(3,2)/A2(2,2));
c=cos(alpha3); s=sin(alpha3);
Rot3=[1 0 0 ; 0 c -s ; 0 s c]; A3=Rot3*A2;
Urot=Rot3*Rot2*Rot1;
% note that we now have a complete QR-factoriztion of A, A=Urot'*A3

% our fourth method is by householder reflections, see householder.m

% our fifth method is by the matlab command qr
[Uqr,R]=qr(A);
% now, A=Uqr*R
% Please, use the matlab command window to study the various matrices.
% For instance, try Urot, Urot'*Urot, eig(Urot), eig(Urot).*conj(eig(Urot)), etc
```

```
% Householder.m
% orthogonalization by means of reflections. The problem is to find the
% proper subspace in which to mirror.
clear

N=5;
A=randn(N,3);

% the first subspace is a line

v1=A(:,1);
h1=zeros(N,1);
h1(1)=sign(v1(1)); % The desired mirror image is norm(v1)*h1
s1=v1/norm(v1)+h1; % The angle of this vector is midway between v1 and desired image
s1=s1/norm(s1);
S1=2*s1*s1'-eye(N); % S=2P-I, P is s1*s1' as s1 is normalized so that s1'*s1=1
u1=S1*v1; % u1 is the desired vector with zeros in all positions but the first
A1=S1*A;

% To produce zeros in a second vector without
% destroying what is already done, we need to mirror
% in a plane that contains a unity vector along the first coordinate

v2=A1(:,2);
h2=zeros(N,1);
h2(2)=sign(v2(2)); % Desired image if v2(1) were zero
temp=v2; temp(1)=0; % To produce "v2(1)=0"
s1=temp/norm(temp)+h2; s1=s1/norm(s1);
s2=zeros(N,1); s2(1)=1;
S2=2*[s1 s2]*[s1 s2]'-eye(N); % As s1 and s2 are orthonormal by construction
u2=S2*v2;
A2=S2*A1;

% In the next step, we must mirror in a 3D space containing the first two coordinates.

v3=A2(:,3);
h3=zeros(N,1);
h3(3)=sign(v3(3));
temp=v3; temp(1)=0; temp(2)=0;
s1=temp/norm(temp)+h3; s1=s1/norm(s1);
s2=zeros(N,1); s2(1)=1;
s3=zeros(N,1); s3(2)=1;
S3=2*[s1 s2 s3]*[s1 s2 s3]'-eye(N);
u3=S3*v3;
A3=S3*A2;

% By now, the principle is clear.
```

```
U=S3*S2*S1; % U is an orthogonal matrix by construction, A3=U*A
% It follows that A=U'*A3 is a qr-factorization of A
norm(A-U'*A3,'fro'), A3
```

Eigenvalue decomposition

Let us assume that the square matrix A has a basis of linearly independent eigenvectors $\mathbf{g}_n,\ n = 1, 2, \ldots, N$. It was shown in the leaflet on "Matrices, Geometry and Mappings – eigenvalues, eigenvectors and diagonalization" that the matrix can be written

$$A = G\,\Lambda\,G^{-1},$$

where

$$G = (\mathbf{g}_1, \ldots, \mathbf{g}_N)$$

and

$$\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N).$$

This decomposition obviously represents a factorization of A, and we can ask what it is good for. Here are two answers:

Answer 1: If you are asked to compute A^{100}, here is how to do it:

$$\begin{aligned} A^{100} &= \left[G\,\Lambda\,G^{-1}\right]^{100} = G\,\Lambda\,G^{-1} \cdots G\,\Lambda\,G^{-1} = \\ &= G\,\Lambda^{100} G^{-1}, \end{aligned}$$

and Λ^{100} is a piece of cake.

———— o O o ————

As you know, eigenvectors can always be chosen to have unit length, so that G has length one columns. Are there any cases where G also has orthogonal columns? Yes indeed, as shown in "Matrices, Geometry and Mapplings – eigenvalues, eigenvectors and diagonalization", for Hermitian matrices A, i.e. for matrices such that

$$A^H = (A^T)^* = A,$$

the eigendecomposition takes the particularly simple form

$$A = G\,\Lambda\,G^H,$$

where G is a unitary matrix ($GG^H = I$).

Answer 2: In some signal processing applications, it is of interest to find a low-rank approximation of a positive definite (definition in the leaflet on Quadratic forms) Hermitian matrix A. What do we mean by that? Assume that the eigenvalues are collected in order so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. Now, let us collect the first r eigenvalues in Λ_1, the rest in Λ_2. Thus

$$\begin{aligned} A =& G\,\Lambda\,G^{-1} = G(\Lambda_1 + \Lambda_2)G^{-1} = \\ =& G\,\Lambda_1 G^{-1} + G\,\Lambda_2 G^{-1} \approx G\,\Lambda_1 G^{-1}, \end{aligned}$$

which will give the best rank r approximation of A in the least squares sense. If you want to speed up computations, you may partition G as well:

$$G\,\Lambda\,G^{-1} = [G_1\,G_2]\,\Lambda_1\,[G_1\,G_2]^H = G_1\,\Lambda_1\,G_1^H.$$

Singular Value Decomposition – SVD

We note with disappointment that so far we lack a general method to factorize an arbitrary matrix. There is one such method, the SVD. One intuitive way to argue for such a method starts from the fact that all Hermitian matrices ($A^H = A$) have a very nice eigenvalue decomposition

$$A = U \, \Lambda \, U^H,$$

where U is a unitary matrix with the eigenvectors of A as columns. So, let us start with an arbitrary matrix B. Then both $B^H B$ and $B \, B^H$ are Hermitian with non-negative eigenvalues only. Thus, there exist eigenvalue decompositions

$$B^H B = U_1 \, \Lambda_1 \, U_1^H,$$

and

$$B \, B^H = U_2 \, \Lambda_2 \, U_2^H.$$

In fact, $B^H \, B$ and $B \, B^H$ share their positive eigenvalues. To see this, assume that $B^H \, B\mathbf{x} = \lambda\mathbf{x}$ for some $\lambda > 0$ and $\mathbf{x} \neq 0$. Then $B \, B^H \, B\mathbf{x} = \lambda B\mathbf{x}$, and thus $B\mathbf{x}$ is an eigenvector of $B \, B^H$ with eigenvalue λ.

Equivalently, we can write

$$\Lambda_1 = U_1^H \, B^H B \, U_1,$$

and similarly for Λ_2.

Now expand

$$\Lambda_1 = U_1^H \, B^H \, U_2 \, U_2^H \, B \, U_1 = \left(U_2^H \, B \, U_1\right)^H \left(U_2^H \, B \, U_1\right)$$

and

$$\Lambda_2 = U_2^H \, B \, U_1 \, U_1^H \, B^H \, U_2 = \left(U_2^H \, B \, U_1\right) \left(U_2^H \, B \, U_1\right)^H.$$

These manipulations will focus our interest on the matrix

$$U_2^H \, B \, U_1,$$

and we ask ourselves what kind of matrix that will produce diagonal matrices Λ_1 and Λ_2 when used according to the formulas

$$X^H \, X = \text{diag}$$

$$X \, X^H = \text{diag}.$$

The answer is – intuitively – that X is diagonal though not square! We have arrived at the formulation of the SVD.

> Any matrix A can be factorized as $A = U_1 \, \Sigma \, U_2^H$, where U_1 and U_2 are unitary, U_1 holds the eigenvectors of $A \, A^H$, U_2 those of $A^H \, A$. The matrix Σ is diagonal, and holds the singular values of A, which equal the positive square root of the shared non-zero eigenvalues of $A \, A^H$ and $A^H \, A$.

Note 1: $\dim A = M \times N$

$\Longrightarrow$

$\dim U_1 = M \times M, \ \dim \Sigma = M \times N, \ \dim U_2 = N \times N$

There are two more possibilities for choosing the dimensions. If $M \neq N$, there is one version called "economy size". Which?

Note 2: $\text{rank } A = \text{rank } \Sigma$

Note 3: The first $r = \text{rank } A$ columns of U_1 span the range space of A. The last $N - r$ columns of U_2 span the null space of A.

We abstain from giving a proof of the SVD, but will give an example of its use. The example will focus on the solution to the equation

$$A\mathbf{x} = \mathbf{b},$$

and will give us the opportunity to (re)introduce a concept: the pseudo-inverse of a matrix.

As you all know, the solution to $A\mathbf{x} = \mathbf{b}$ exists in the following two cases:

- A is square, full rank. Solution:

$$\mathbf{x} = A^{-1}\mathbf{b}$$

- A is taller than wide, full rank. Least squares solution, formulated in terms of the projection matrix that projects onto the range space of A:

$$A\mathbf{x} = P_A\mathbf{b},$$

As $P_A = A(A^T A)^{-1}A^T$, we find

$$\mathbf{x} = (A^T A)^{-1}A^T\mathbf{b}.$$

Now, we may ask ourselves: is there *always* some kind of solution to $A\mathbf{x} = \mathbf{b}$? For instance

$$x + y = 2$$

or

$$\begin{cases} x + y + z = 2 \\ x + y + z = 4 \end{cases}$$

The answer is yes, there always exists some kind of solution. We speak of minimum norm solutions, for example

$$x + y = 2$$

has the minimum norm solution $x = y = 1$. It is obtained as the solution to

$$\arg\min \left(x^2 + y^2\right)$$

subject to the restriction

$$x + y = 2.$$

We also speak about minimum norm least squares solutions. For example

$$\begin{aligned} x + y + z &= 2 \\ x + y + z &= 4 \end{aligned}$$

has the minimum norm, least squares solution $x = y = z = 1$. One way to argue is

$$\begin{aligned} u &= 2 \\ u &= 4 \end{aligned}$$

has the least squares solution $u = 3$,

$$x + y + z = 3$$

has the minimum norm solution $x = y = z = 1$. Now:

> Using the concept of the pseudo-inverse denoted A^+, of a matrix A, all systems of equations $A\mathbf{x} = \mathbf{b}$ have the pseudo-solution $\mathbf{x} = A^+\mathbf{b}$.
> For the full rank cases – square A, least squares solution when A is taller than wide – this solution coincides with the known solutions. Also, it is a minimum norm solution for cases of more unknowns than independent equations.

Maybe we should discuss the construction of the pseudo-inverse of a given matrix A. Let us start with a diagonal matrix. The system of equations

$$\begin{cases} x = 2 \\ 2y = 3 \end{cases}$$

can be written

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

with solution

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix},$$

so that the pseudo-inverse and the inverse (here same thing) can be obtained by inverting the diagonal entries. What about

$$\begin{cases} x + 0y + 0z = 2 \\ 0x + 2y + 0z = 3 \\ 0x + 0y + 0z = 4 \end{cases},$$

corresponding to

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}.$$

Now, this seems stupid, but we can still argue for a minimum norm, least squares solution. The first two equations can be solved exactly — $x = 2,\ y = 3/2$ — and the third can never be solved, the squared error is independent of the choice of z. So, the minimum norm helps us to choose $z = 0$. Thus,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 3/2 \\ 0 \end{bmatrix}$$

Again, the pseudo-inverse is obtained by inverting the non-zero diagonal entries. This argument can be carried on to the conclusion:

> The pseudo-inverse of any diagonal matrix of dimensions $M \times N$ is given by the $N \times M$ diagonal matrix that has entries which are the inverse of the non-zero elements of the original. The rest of the diagonal entries are zero.

Now, it is a simple thing to derive the pseudo-inverse of any matrix using the SVD:

$$A = U_1 \, \Sigma \, U_2^H$$

$$\Longleftrightarrow$$

$$U_1^H \, A \, U_2 = \Sigma$$

$$\Longleftrightarrow$$

$$\Sigma^+ = \left(U_1^H \, A \, U_2\right)^+ = U_2^+ \, A^+ \left(U_1^H\right)^+ = U_2^H \, A^+ \, U_1$$

$$\Longleftrightarrow$$

$$A^+ = U_2 \, \Sigma^+ \, U_1^H$$

Note 4: $(A^+)^+ = A^{++} = A$

Note 5: $A^+ \, A = U_2 \, \Sigma^+ \, U_1^H \, U_1 \, \Sigma \, U_2^H = U_2 \, \Sigma^+ \, \Sigma \, U_2^H =$

$$= U_2 \begin{bmatrix} I & O \\ O & O \end{bmatrix} U_2^H = \begin{bmatrix} I & O \\ O & O \end{bmatrix}$$

Case 1: $\begin{bmatrix} \qquad \end{bmatrix} \begin{bmatrix} \\ \\ \end{bmatrix} = \begin{bmatrix} \; \end{bmatrix} = I$

Case 2: $\begin{bmatrix} \\ \\ \\ \end{bmatrix} \begin{bmatrix} \qquad \end{bmatrix} = \begin{bmatrix} \qquad \\ \\ \\ \end{bmatrix} = \begin{bmatrix} I & O \\ O & O \end{bmatrix}$

Note 6: The same goes for $A\,A^{+}$.

———— o O o ————

As an application example, consider image compression. A black and white image is of course nothing but a matrix, the pixels corresponding to the entries of the matrix. Call the image A, and perform an SVD:

$$A = U_1\,\Sigma\,U_2^H.$$

Now assume that the image has a lot of structure and some added noise. This means that Σ has some large singular values, from the structured image, and many small singular values, from the noise. Collect the large singular values in Σ_1. Then

$$A = U_1\,\Sigma\,U_2^H = U_1\,(\Sigma_1 + \Sigma_2)\,U_2^H \approx U_1\,\Sigma_1\,U_2^H$$

gives the best least squares approximation for a given rank (number of non-zero entries in Σ_1). Now, you only have to store/transmit the "economy size" version of the SVD. Let A be $M \times N$, rank $\Sigma_1 = r << M, N$. Then you can shorten U_1 to $M \times r$, Σ_1 is diagonal $r \times r$, and U_2^H is shortened to $r \times N$.

Please run the m-file svdgames in Matlab.

```
% svdgames.m
% some applications of the SVD and the pseudoinverse
clear

% The first application is simple

A=randn(4,2);
b=randn(4,1);
x=A\b; % Least squares solution to the overdetermined system

[U,S,V]=svd(A); % A=U*S*V', U,V unitary, S diagonal
Spinv=(S*diag(1./diag(S)./diag(S)))'; % The pseudoinverse to a diagonal matrix
Apinv=V*Spinv*U'; % Apinv is the LEFT inverse of A
norm(x-Apinv*b,'fro') % we get the same answer
pause

% Now, let us see a stupid example

A=[1 1 1 ; 1 1 1];
b=[2 ; 4];
x=A\b % Not liked by matlab, also wrong solution
[U,S,V]=svd(A);
Spinv=S'/S(1,1)/S(1,1); % invert the non-zero singular values
Apinv=V*Spinv*U';
x=Apinv*b % Produces the minimum norm, least
% squares solution to the stupid example
% namely x1=x2=x3=1
pause

% Now, let us see some noise reduction

N=70;
vec=1:N;
A=vec'*vec; % A has rank 1
noise=5*randn(N,N);
data=A+noise;
[U,S,V]=svd(data);
Strunc=zeros(N,N);
Strunc(1,1)=S(1,1); % As A has rank 1 we only keep the largest singular value
Aapprox=U*Strunc*V';
norm(A-data,'fro')
norm(A-Aapprox,'fro') % An improvement
```