Matrix Factorizations

LU Factorization

Definition ...

Given a square $k \times k$ matrix S, the LU factorization (or decomposition) represents S as the product of two triangular matrices,

$$S = L\,U\ ,$$

where L is lower triangular and U is upper triangular. To resolve identifiability problems, assume that the diagonal elements of L are $\ell_{ii} = 1$.

Computations ...

It is easiest to see how the algorithm works by writing down what needs to happen in the 3×3 case:

$$\begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{23} & s_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{23} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

If you write out the product, you will see that its possible to solve uniquely for the resulting elements ℓ_{ij} and u_{ij}. The number of operations is on the order of that required for the matrix product, k^2 dot products each of length k, or $O(k^3)$.

The computations can be done simply via a Gaussian elimination. Think of each step (*i.e.*, the process of zeroing an element) of Gaussian elimination as a matrix multiplication on the left of S. For example, if $s_{11} \neq 0$, we can zero s_{12} and s_{31} using the first row by forming the product of a lower triangular matrix and S:

$$\begin{pmatrix} 1 & 0 & 0 \\ -s_{21}/s_{11} & 1 & 0 \\ -s_{31}/s_{11} & 0 & 1 \end{pmatrix} \begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{23} & s_{33} \end{pmatrix} = \begin{pmatrix} s_{11} & s_{12} & s_{13} \\ 0 & s^*_{22} & s^*_{23} \\ 0 & s^*_{23} & s^*_{33} \end{pmatrix}$$

Assuming the needed divisors are not zero, we can accumulate a product like this as $L^{-1}S = U$. (Yes, the inverse of a lower triangular matrix is lower triangular.) If the needed divisor is zero, however, one can *pivot*, swapping rows in order to move a non-zero element into position.

The LispStat function `lu-decomp` does this chore, but its output is typically more complex than you would expect (and may even be complex!), including pivots for example. More often, you will use `lu-decomp` indirectly, as it is called by `lu-solve`, `determinant`, and `inverse`.

Applications of LU ...

Once you have an LU decomposition, it becomes simple to

- Find the determinant of S, $|S| = |L|\,|U| = |U| = \prod u_{ii}$,
- Solve a system of equations $Sx = b$ via solving first $LUx = L(Ux) = b$ for $Lc = b$, then solving $Ux = c$. Solving each of these takes $O(k^2)$ operations.
- Invert S (assuming it is non-singular), by solving equations of the form $SI_j = LUI_j = 1_j$, where I_k is a $k \times k$ identity matrix and 1_j is the indicator vector vector with elements δ_{i-j} (*i.e.*, it has a one in location j and zeros elsewhere).

Cholesky Factorization

Definition ...

The Cholesky factorization applies to $k \times k$, symmetric, positive semidefinite matrices (*i.e.* covariance matrices). It can be thought of as a variation on an LU factorization, but with the factors $U' = L$ so that

$$S = LL' \,.$$

In a sense, the Cholesky factor behaves like the square root of a matrix. For example, if

$$X \overset{\text{iid}}{\sim} N_k(0, I) \quad \text{and} \quad C = LL' \,,$$

then

$$LX \sim N_k(0, LL' = C) \,.$$

Indeed, this simple result provides one of the most common uses of the Cholesky factorization in statistics: the generation of multivariate normals with arbitrary covariance matrix.

Computations ...

To see how the Cholesky factorization works, again think about what it would take to solve a simple 3×3 example:

$$\begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{23} & s_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{23} & 1 \end{pmatrix} \begin{pmatrix} \ell_{11} & \ell_{21} & \ell_{31} \\ 0 & \ell_{22} & \ell_{32} \\ 0 & 0 & \ell_{33} \end{pmatrix} = \begin{pmatrix} \ell_{11}^2 & \ell_{11}\ell_{21} & \ell_{11}\ell_{31} \\ \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots \end{pmatrix}$$

Assuming $s_{11} > 0$ (as it must be if S is p.d.), it's easy to solve for the first column of L. Now think recursively. To find the rest of L, you once again need to solve a triangular system, but with slightly modified elements. As with the LU method, the operation count is $O(k^3)$.

The LispStat function `chol-decomp` performs the Cholesky factorization. The output includes an indication of whether the input matrix S is positive definite.

Applications of Cholesky ...

There is a very nice way to think about the Cholesky factorization of a covariance matrix. Reversing the notion used in generating Gaussian r.v.'s, note that if $C = LL'$, then $L^{-1}C(L')^{-1} = I$. That is, the Cholesky factors also provide a method for converting correlated normals into uncorrelated normals. For example, if $(X_1, \ldots, X_k) \sim N_k(0, C)$, then (with superscripts denoted elements from the inverse)

$$\text{Cov}(\ell^{11}X_1, \ell^{21}X_1 + \ell^{22}X_2) = 0 \; .$$

By moving the constants around we get

$$\text{Cov}(X_1, \frac{\ell^{21}}{\ell^{22}}X_1 + X_2) = 0$$

which looks awfully like a regression (the residuals of X_2 regressed on X_1 are uncorrelated with X_1).

A very special case of the Cholesky factorization is important in time series analysis. The covariance matrix of so-called stationary time series is *Toeplitz*, symmetric with constants along each diagonal. For these matrices, the Cholesky factorization gives the collection of autoregressive models of increasing order, AR(1), AR(2),... AR(k). The special version of Cholesky that's used there is known as Levinson's recursion.

Levinson's recursion ...

Details elaborating its role in sequence of projections...

Closest p.d. approximation ...

The Cholesky factorization breaks down if the matrix being factored is not p.s.d. As in LispStat, one can then perturb the matrix being factor by adding dI_k to the diagonal with $d > 0$ a positive constant. The size of the perturbation is one measure of how "far" the matrix is from being positive semidefinite. The SVD discussed later gives another way.

Eigenvalue Decomposition

Definition ... ???

All $k \times k$ square matrices S possess a spectral representation

$$S = \sum_{j=1}^{k} \lambda_j e_j e_j' \qquad (1)$$

(a sum of rank 1 matrices) where

$$Se_j = \lambda_j e_j$$

and $e_j'e_k = 0$ if $\lambda_j \neq \lambda_k$. The eigenvalues λ_j and eigenvectors e_j may be complex. If the eigenvalues are distinct and nonzero, then the eigenvectors are orthogonal. For later convenience, assume assume the eigenvalues are ordered by their distance from zero,

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_k| \geq 0 \ .$$

Computation ...

Eigenvalue-eigenvector decompositions are among the hardest to compute, and are in general only obtained as the approximate result of an iterative algorithm. LispStat offers the functions `eigen`, `eigenvalues`, and `eigenvectors`, albeit each is restricted to symmetric matrices. Unfortunately, this means that you cannot use the `eigen` code to find the zeros of arbitrary polynomials.

Covariances ...

Suppose that C is symmetric, as with a covariance matrix. Then all of the λ_j and e_j are real, and we can write

$$SE = E\Lambda, \quad E'E = I, \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_k) \ ,$$

and $E = (e_1, \ldots, e_k)$ is the matrix of eigenvectors. The eigenvectors define orthogonal linear combinations of the underlying r.v.'s which have variance λ_j.

Principal components ...

Eigenvalues/vectors give the solution of the classical extremal problem

$$\max_{x'x=1, x \in R^k} x'Cx \ .$$

That is, find the linear combination of variables having the largest variance, with the restriction that the sum of squared weights is 1. Minimizing the usual Lagrangian expression

$$x'Cx - \lambda(x'x - 1)$$

implies that $Cx = \lambda x$. Thus, x is the eigenvector with largest eigenvalue. The problem continues by finding that linear combination with largest variance which is uncorrelated with the first. Surprise, it's determined by the second eigenvector. Etc...

Functions of a matrix (square root) ...

Although the Cholesky decomposition gives one square root, the eigenvalue formula is equally appealing, though not lower triangular. Namely, define

$$S^{1/2} = \sum_j \sqrt{\lambda_j} e_j e_j' \ .$$

You can check that $S = S^{1/2}S^{1/2}$. In general, the spectral decomposition (1) gives a nice way to extend any scalar function f to matrices,

$$f(S) = \sum_j f(\lambda_j) e_j e_j' \ .$$

Generalized inverse ...

If the matrix S is nonsingular, the previous idea suggests that the the inverse of such a matrix ought to be

$$S^{-1} = \sum_j \frac{1}{\lambda_j} e_j e_j'$$

You can check that this intuition is indeed correct.

When the matrix S is singular, it has at least one zero eigenvalue so the previous expression does not apply. In this case, we can define a *generalized inverse* (unlike S^{-1}, generalized inverses are defined in various ways and are not unique)

$$S^{-} = \sum_{j=1}^{m} \frac{1}{\lambda_j} e_j e_j' \ , \quad \lambda_j \neq 0 \text{ for } j \leq m \leq k.$$

Using this formulation, we have

$$SS^{-} = (\sum_j \lambda_j e_j e_j')(\sum_{j=1}^{m} \frac{1}{\lambda_j} e_j e_j') = \sum_{j=1}^{m} e_j e_j' = M_m \neq I_k$$

When solving the normal equations for a regression with a singular cross-product matrix (let $S = X'X$), we obtain

$$(X'X)\hat{\beta} = X'Y \quad \Rightarrow \quad M\hat{\beta} = (X'X)^{-}X'Y \ .$$

Thus $\hat{\beta}$ is not unique since $Me_k = 0$. We can add multiples of any eigenvector associated with zero eigenvalue to $\hat{\beta}$ and still solve the normal equations.

Matrix norms and inner-products ...

The spectral representation gives nice forms for the two most popular norms for square matrices:

$$\text{Frobenius norm: } \|S\|_F^2 = \sum_{i,j} s_{ij}^2 = \sum_j \lambda_j^2$$

and

$$\text{Operator norm or } L_2 \text{ norm: } \|S\| = \sup_{y \neq 0} \frac{\|Sy\|}{\|y\|} = |\lambda_1| \ .$$

The Frobenius norm of a matrix comes with the associated inner-product

$$\langle S, T \rangle_F = \text{ trace}(ST') \tag{2}$$

QR Decomposition

Definition ...

Unlike the previous decompositions, the QR decomposition does not require the matrix in question to be square. For the $n \times k$ matrix X with $n \geq k$, the QR decomposition is

$$X = Q_{n \times k} \, R_{k \times k} \ , \quad Q'Q = I, \quad R \text{ upper triangular.}$$

Computation ...

The QR decomposition is the result of applying the Gram-Schmidt orthogonalization to the columns of X. In effect, the values in R are regression coefficients.

Here's how the calculations proceed. First, define the first column of Q as the normalized first column of X,

$$Q_1 = X_1/\|X_1\| \ ,$$

where $\|x\|^2 = \sum_i x_i^2$. Now regress Q_1 out of the rest of the columns of X and store the resulting coefficients as the first row of R,

$$X_j^* = X_j - (X_j' Q_1) Q_1 \ ,$$

implying that $X_1' X_j^* = 0, \ j = 2, \ldots, k$. Now continue recursively, determining the rest of Q and the remaining rows of R from the matrix $(X_2^*, \ldots, X_k^*)$. This row-based method of computing the QR decomposition, also called the modified Gram-Schmidt algorithm, is known to be more stable numerically than the column-based alternative. The calculation scheme requires order $O(nk^2)$ operations, coming from $k + (k-1) + \cdots + 1$ dot products of n elements.

Relationship to Cholesky ...

Given that $X = QR$, note that

$$X'X = (QR)'QR = R'(Q'Q)R = R'R$$

which is a lower triangular (let $L = R'$) factorization of the cross-product matrix. The elements of R have a simple interpretation from the initial description, namely as regression coefficients.

Applications ...

The QR decomposition has some simple uses, such as determining the rank (count the number of non-zero diagonal values of R) of a matrix or creating an orthogonal set of regressors.

In this latter context, the QR decomposition is quite useful. For example, if $X = QR$, then the usual linear model

$$Y = X\beta + \epsilon, \quad \epsilon \overset{\text{iid}}{\sim} E\epsilon_i = 0, \, \text{Var}(\epsilon_i) = \sigma^2 \ ,$$

becomes

$$Y = (QR)\beta + \epsilon = Qc + \epsilon, \quad c = R\beta \ .$$

The least squares estimator for c is trivially

$$\hat{c} = (Q'Q)^{-1} Q'Y = Q'Y$$

and

$$\text{Var}(\hat{c}) = \sigma^2 (Q'Q) = \sigma^2 I_k \ ,$$

so that the elements of $\hat{c}$ are uncorrelated (independent under normality). Of course, if you want to work with β instead, you need to solve (this is an order $O(k^2)$ calculation... inversion is order k^3)

$$R\hat{\beta} = \hat{c}$$

for which

$$\text{Var}(\hat{\beta}) = \sigma^2 R^{-1}(R^{-1})' = \sigma^2(R'R)^{-1} = \sigma^2(X'X)^{-1} \ .$$

A particular benefit of doing regression this way is that it avoids the direct calculation of $X'X$ which squares the condition number of the problem, making numerical errors more insidious.

Augmented matrix for regression ...

Typically, when the QR is used in regression, the initial algorithm is applied to the augmented matrix $(X|Y)$. This leaves the coefficients $\hat{c}$ in positions $1, 2, \ldots, k$ of the last column of the augmented R. The last column of the augmented Q are the residuals, and the square of the last diagonal element of R is the residual sum of squares. Also, it is advisable in this context to place a column of one's as the first column of X so that the first step of the algorithm centers the remaining vectors, reducing the size of subsequent inner products (and increasing the numerical precision).

Singular Value Decomposition SVD

Definition ...

The SVD generalizes the eigenvalue or spectral decomposition of a square matrix to $n \times k$ matrices, with $n \geq k$. Given any $n \times k$ real-valued matrix X, the decomposes or "factors" X as a product of two orthogonal matrices and a diagonal matrix:

$$X_{n\times k} = U_{n\times k} D_{k\times k} V'_{k\times k} \ ,$$

where the columns of U and V are orthogonal and D is diagonal,

$$D = \text{diag}(\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0) \ .$$

The diagonal values are known as the singular values. Corresponding to the rank-one decomposition (1) offered by the spectral decomposition, we have

$$X = \sum_{j=1}^{k} \sigma_j u_j v_j' \ . \tag{3}$$

Matrix norms ...

The SVD gives expressions for the matrix norms defined previously for square matrices. You can check that

$$\|X\|_F^2 = \sum_j \sigma_j^2, \quad \|X\| = \sigma_1$$

Interpretations ...

By substituting and noting the orthogonality,

$$X'X = V\,D^2\,V'\ ,$$

implying that σ_j^2 are the eigenvalues of the cross-product matrix $X'X$ and that the columns of V are the eigenvectors. Similarly, if we consider the outer product,

$$XX' = U\,D^2\,U'\ .$$

To get a better feel for the columns of U, consider the projection matrix (or "hat" matrix) of regression. In a regression model

$$Y = X\beta + \epsilon\ ,$$

the fitted values can be written as

$$\hat{Y} = X\hat{\beta} = X(X'X)^{-1}X'Y = H\,Y$$

where

$$H = X(X'X)^{-1}X' = (UDV')(VD^2V')^{-1}(UDV')' = UU'\ .$$

Thus, the fitted values are a linear combination of the columns of U,

$$\hat{Y} = U(U'Y)\ ,$$

and the residuals live in the orthogonal complement of U.

Approximation theorem ...

Some interesting statistical applications of the SVD arise from its use to approximate a matrix by one of lower rank. The Eckart-Young-Mirsky theorm shows that the best L_2 approximation to an indicated matrix is found from the SVD:

$$\min_{\text{rank}(Y)=m} \|X-Y\|_2^2 = \|X-X_m\|_2^2 = \sigma_{m+1}^2$$

and

$$\min_{\text{rank}(Y)=m} \|X-Y\|_F^2 = \|X-X_m\|_F^2 = \sum_{m+1}^{k} \sigma_j^2$$

where for $m \le k$

$$X_m = \sum_{j=1}^{m} \sigma_j u_j v_j'\ .$$

In effect, to form X_m one zeros the singular values $\sigma_{m+1}, \ldots, \sigma_k$.

Proof: The proof of this result is quite simple, once you use the right geometric argument. Using the inner-product (2), you can define a basis for the space of $n \times k$

matrices. The dimension of the space is nk. The usual coordinate system with basis vectors $c_i \in R^n$ and $r_j \in R^k$ (zeros everywhere but in the indexed location) gives

$$X = \sum_{i,j} x_{i,j} c_i r_j' \quad \Rightarrow \quad \langle X,\, c_\ell r_m' \rangle = x_{\ell,m}$$

The SVD gives the coordinates of X using a different set of basis vectors, namely $u_i v_j', (i = 1, \ldots, n;\ j = 1, \ldots, k)$. To get the closest approximation to X in this coordinate system is thus a simple projection which does not include the last components of the sum (3).

Total least squares TLS ...

Statistics, with its focus on prediction, often leads one to ignore the common problem in regression: The predictors often have just as much error as the response. One way to deal with this is rather than think of how to approximate the response Y as a linear combination of the predictors, instead try to

$$\min \| [X|Y] - [\tilde{X}|\tilde{Y}] \|$$

over all $n \times (k+1)$ matrices $[\tilde{X}|\tilde{Y}]$ subject to the condition $\tilde{Y}$ lies in the column span of $\tilde{X}$. Any set of coefficients $\tilde{\beta}$ that describe

$$\tilde{Y} = \tilde{X}\tilde{\beta} \quad \text{or} \quad [\tilde{X}|\tilde{Y}](\tilde{\beta}', -1)' = 0$$

are the chosen "regression coefficients". It can be shown that this TLS estimator $\tilde{\beta}$ is consistent for β when the X's are contaminated by error (ordinary least squares is *not consistent* in this context).

SVD and total least squares ...

If the rank of $[X|Y]$ is k or smaller, the system is degenerate and an exact solution may be found. Assuming on the other hand that $\sigma_{k+1} > 0$, the Eckart-Young-Mirsky theorem cited above shows that the SVD provides the answer,

$$[X|Y] - [\tilde{X}|\tilde{Y}] = \sigma_{k+1} u_{k+1} v_{k+1}' \ ,$$

which is a rank one adjustment to the augmented matrix $[X|Y]$. Since the approximating matrix is

$$[\tilde{X}|\tilde{Y}] = \sum_{j=1}^{k} \sigma_j u_j v_j'$$

the needed coefficient vector is just a multiple of v_{k+1} which has a -1 in the last position and "regression coefficients" are the first k elements

$$\tilde{\beta}_j = \frac{-1}{v_{k+1,k+1}} v_{k+1,j} \ .$$

Note that $v_{k+1,k+1} \neq 0$ since we have assumed $\sigma_{k+1} > 0$.

Closed form of TLS solution ...

Surprisingly, there is a simple expression for the TLS solution $\tilde{\beta}$ which closely resembles the OLS solution:

$$\tilde{\beta} = (X'X - \sigma_{k+1} I_k)^{-1} X'Y \ . \tag{4}$$

(*Proof:*. v_{k+1} is an eigenvector of $[X|Y]'[X|Y]$ with eigenvalue σ^2_{k+1}. Pulling off the "top row" of this expression leads to (4).)

Comments on TLS ...

Thinking about the source of bias in the OLS solution suggests that this is a useful approach. Assume the vectors $y, x \in R^n$ have mean zero and we want to estimate the regression coefficient β of y on x in the familiar model

$$y = x\beta + \epsilon$$

However, assume as well that x has been contaminated by some independent measurement error $u \in R^n$, $\text{Var}(u_i) = \sigma^2_u$, so that we do not see x, but rather $x_u = x + u$. If we now do the usual OLS regression, with x_u in place of x, we find that the estimator is biased and inconsistent,

$$\hat{\beta} = \frac{x_u' y}{x_u' x_u'} \overset{\text{P}}{\rightarrow} \frac{\text{Cov}(x, y)}{\text{Var}(x) + \sigma^2_u}$$

The singular value σ_{k+1} gives an estimate of the level of measurement error.

One gets a bit queasy, though, since this estimator moves in the opposite direction of common statistical estimators which shrink toward zero. You can read more about this method in van Huffel and Vandewalle (1991), *The Total Least Squares Problem*, SIAM, Philadelphia. Other solutions of this so-called "errors in variables" model include the use of instrumental variables in econometrics.