Singular Value and Eigenvalue Decompositions

Frank Dellaert

May 2008

1 The Singular Value Decomposition

The singular value decomposition (SVD) factorizes a linear operator $A : R^n \rightarrow R^m$ into three simpler linear operators:

1. Projection $z = V^T x$ into an r-dimensional space, where r is the rank of A
2. Element-wise multiplication with r singular values σ_i, i.e., $z' = Sz$
3. Transformation $y = Uz'$ to the m-dimensional output space

Combining these statements, A can be re-written as

$$A = USV^T \tag{1}$$

with U an $m \times r$ orthonormal matrix spanning A's column space $im(A)$, S an $r \times r$ diagonal matrix of singular values, and V an $n \times r$ orthonormal matrix spanning A's row space $im(A^T)$.

2 The Eigenvalue Decomposition

The eigenvalue decomposition applies to mappings from R^n to itself, i.e., a linear operator $A : R^n \rightarrow R^n$ described by a square matrix. An eigenvector e of A is a vector that is mapped to a scaled version of itself, i.e., $Ae = \lambda e$, where λ is the corresponding eigenvalue. For a matrix A of rank r, we can group the r non-zero eigenvalues in an $r \times r$ diagonal matrix Λ and their eigenvectors in an $n \times r$ matrix E, and we have

$$AE = E\Lambda$$

Furthermore, if A is full rank ($r = n$) then A can be factorized as

$$A = E\Lambda E^{-1}$$

which is a diagonalization similar to the SVD (1). In fact, *if and only if* A is symmetric[1] and positive definite (abbreviated SPD), we have that the SVD and the eigen-decomposition coincide

$$A = USU^T = E\Lambda E^{-1}$$

with $U = E$ and $S = \Lambda$.

Given a non-square matrix $A = USV^T$, two matrices and their factorization are of special interest:

$$A^T A = VS^2V^T \tag{2}$$

$$AA^T = US^2U^T \tag{3}$$

Thus, for these matrices the SVD on the original matrix A can be used to compute their SVD. And since these matrices are by definition *SPD*, this is also their eigen-decomposition, with eigenvalues $\Lambda = S^2$.

[1] if we allow complex matrices, A must be hermitian, i.e., A's conjugate transpose $A^* = A$

3 Principal Components Analysis

An important application is principal components analysis (PCA). To motivate this, consider a normally distributed, r-dimensional vector $x \sim N(0, I_r)$ with zero mean and unit covariance matrix. In other words, each component of the vector x is drawn independently from a 1-dimensional Gaussian with zero mean and unit variance, i.e., *white noise*. If we transform the vector x to an m-dimensional output space using the following linear transformation

$$y = \mu + Wx$$

with W of size $m \times r$, then the m-dimensional vectors y will be distributed as $y \sim N(\mu, WW^T)$.

Given a $m \times n$ data matrix Y of n data samples y, PCA uses eigen-analysis to recover the (scaled) *principal components* W. After subtracting the sample mean μ from all vectors y forming the matrix A, the eigen-decomposition of the sample covariance matrix AA^T is obtained by (3):

$$AA^T = US^2U^T = (US)(US)^T = WW^T$$

Hence, the data can be *whitened* by

$$x = W^T(y - \mu)$$

Just as a sanity check, the resulting covariance of x is indeed unity:

$$XX^T = W^T(AA^T)W = W^T(WW^T)W = I_r$$

Two more facts regarding PCA are worth noting:

1. If the dimensionality m of the data matrix Y is very large, it is more efficient to use the eigen-decomposition (2) of A^TA and obtain the principal components as $W = AV$.
2. The $m \times r$ matrix W contains the *scaled* principal components. Clearly, the normalized principal components are the columns of U, and their lengths are the singular values σ.

Finally, it is interesting that to sample from the density $y \sim N(\mu, WW^T)$ one can proceed in two ways:

1. Sample from $x \sim N(0, I_r)$, and form $y = \mu + Wx$, i.e., generate a white latent vector x and use the principal components W of the data Y to generate an m-dimensional vector.
2. Sample from $c \sim N(0, I_n)$, and form $y = \mu + Ac$, i.e., simply form a linear combination of the original (centered) data points, with coefficients $\left\{c_j\right\}_{j=1}^{n}$ drawn from zero-mean, unit-variance Gaussians.

4 A Simple Monte Carlo PCA Algorithm

If the data lives in a small subspace of rank r, then simply drawing $s \geq r$ samples Y_s from the data and performing PCA will with high probability recover a set of scaled principal components $W_s = U_sS_s$ (of size $m \times r$) that span the subspace. Then, forming the $r \times n$ matrix X_s of latent variables

$$X_s = W_s^T A$$

for *all* centered samples A will not quite whiten the data. However, by doing a second decomposition on the covariance of X_s

$$X_sX_s^T = W_2W_2^T$$

the latent variables are whitened. Hence, if we form $W = W_sW_2$ and $X = W^TA = W_2^TX_s$, we have

$$XX^T = W_2^T\left(X_sX_s^T\right)W_2 = W_2^T\left(W_2W_2^T\right)W_2 = I_r$$

If the data is very high-dimensional, i.e., $m \gg n$, a similar algorithm can be devised by sampling rows.