Foundations of Computer Vision

Wesley. E. Snyder
North Carolina State University

Hairong Qi
University of Tennessee, Knoxville

Last Edited February 8, 2017

Apply this operator to a scalar, f, and we get a vector which does have meaning, the gradient of f:

$$\nabla f = \left[\begin{array}{cc} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{array} \right]^{\mathrm{T}}. \tag{3.23}$$

Similarly, if $\mathbf{f}$ is a vector, we may define the *divergence* using the inner (dot) product (in all the following definitions, only the two-dimensional form of the ∇ operator defined in Eq. 3.21 is used. However, remember that the same concepts apply to operators of arbitrary dimension):

$$div\mathbf{f} = \nabla\mathbf{f} = \left[\begin{array}{cc} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{array} \right] \left[\begin{array}{c} f_1 \\ f_2 \end{array} \right] = \frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y}. \tag{3.24}$$

We will also have opportunity to use the outer product of the ∇ operator with a matrix, which is the Jacobian:

$$\nabla \times \mathbf{f} = \left[\begin{array}{c} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{array} \right] \left[\begin{array}{cc} f_1 & f_2 \end{array} \right] = \left[\begin{array}{cc} \frac{\partial f_1}{\partial x} & \frac{\partial f_2}{\partial x} \\ \frac{\partial f_1}{\partial y} & \frac{\partial f_2}{\partial y} \end{array} \right]. \tag{3.25}$$

3.2.7 Eigenvalues and Eigenvectors

If matrix A and vector $\mathbf{x}$ are conformable, then one may write the *characteristic equation*

$$A\mathbf{x} = \lambda\mathbf{x}, \lambda \in \Re. \tag{3.26}$$

Since $A\mathbf{x}$ is a linear operation, A may be considered as a transformation which maps $\mathbf{x}$ onto itself with only a change in length. There may be more than one *eigenvalue*[6], λ, which satisfies Eq. 3.26. For $\mathbf{x} \in \Re^d$, A will have exactly d eigenvalues (which are not, however, necessarily distinct). These may be found by solving $\det(A - \lambda I) = 0$. (But for $d > 2$, we do not recommend this method. Use a numerical package instead.)

Given some eigenvalue λ, which satisfies Eq. 3.26, the corresponding $\mathbf{x}$ is called the corresponding *eigenvector*.[7]

3.2.8 Eigendecomposition

If a positive semidefinite matrix, A, is also symmetric, then it may be written as $A = BB^{\mathrm{T}}$ for some matrix B. Many of the matrices encountered in Computer Vision are positive semidefinite. In particular, covariance matrices, which we will encounter several times, have this property.

[6] "Eigen-" is the German prefix meaning "principal" or "most important." These are NOT named for Mr. Eigen.

[7] For any given matrix, there are only a few eigenvalue/eigenvector pairs.

Assuming we know how to compute eigenvalues and eigenvectors, a positive semidefinite matrix can be written as

$$A = E\Lambda E^{\mathrm{T}} , \tag{3.27}$$

where E is a matrix in which each column is an eigenvector of A, and Λ is a square, diagonal matrix with corresponding eigenvalues on the diagonal. Should A not be positive semidefinite and symmetric, but still invertible, the form simply changes to

$$A = E\Lambda E^{-1} , \tag{3.28}$$

We can find the null space easily by taking the eigendecomposition and considering all the eigenvectors which correspond to eigenvalues of zero. As an example, using the same matrix as we used in Eq. 3.7 earlier (which is NOT symmetric), we find

$$A = \begin{bmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \\ 2 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 0.7560 & 0.9469 & 0.4082 \\ 0.2927 & -0.1438 & -0.8165 \\ 0.5855 & -0.2877 & 0.4082 \end{bmatrix} \times$$
$$\begin{bmatrix} 10.5826 & 0 & 0 \\ 0 & 1.4174 & 0 \\ 0 & 0 & -0.0000 \end{bmatrix} \begin{bmatrix} 0.3727 & 0.6398 & 0.9069 \\ 0.7585 & -0.0884 & -0.9353 \\ 0.0000 & -0.9798 & 0.4899 \end{bmatrix} .$$

Observe that the right column of E (the left-hand matrix) is the eigenvector corresponding to the zero eigenvalue, and it is precisely the null space we found in the example in section 3.2.3. Also note that because A is not symmetric, we had to use E^{-1} for the third matrix in the product.

In this example, if we had two zero eigenvalues, the two corresponding eigenvectors would be a basis for the space of elements of the null space. That is, any vector which is a linear sum of those two vectors would be in the null space.

But of course, this only works for square matrices!

In the next section, we generalize this concept and develop *singular value decomposition*, or SVD.

3.2.9 Singular Value Decomposition

While eigendecomposition is restricted to square matrices, SVD can be applied to any matrix. SVD decomposes a matrix A into the product of three matrices:

$$A = UDV^{\mathrm{T}} , \tag{3.29}$$

where U and V are orthonormal matrices, and D is diagonal with values on the diagonal sorted from largest to smallest.

To understand what this means, suppose you had a machine that could compute a matrix decomposition just as described in Eq. 3.29, and consider using it to decompose A. Now, since $A = UDV^{\mathrm{T}}$, we know that $A^{\mathrm{T}} = VDU^{\mathrm{T}}$. Multiplying A on the right by A^{T} produces $AA^{\mathrm{T}} = UDV^{\mathrm{T}}VDU^{\mathrm{T}}$, but since V is orthonormal, $V^{\mathrm{T}}V = I$, and AA^{T} may be written as

$$AA^{\mathrm{T}} = UD^2U^{\mathrm{T}} \, . \tag{3.30}$$

and the columns of U (see Eq. 3.27) will be the eigenvectors of AA^{T}. Similarly, one can show that the columns of V will be the eigenvectors of $A^{\mathrm{T}}A$. D will be a diagonal matrix with the square root of the eigenvalues of $A^{\mathrm{T}}A$ on the diagonal sorted in descending order[8].

The diagonal elements of D are referred to as the *singular values* of A, so the singular values of A are the square roots of the eigenvalues of AA^{T}.

This observation tells us one way to compute the SVD: use eigendecomposition. But more important: eigendecomposition can only be applied to square matrices, and both AA^{T} and $A^{\mathrm{T}}A$ are square, for ANY matrix A.

Now suppose our problem is to find the null space of a 3×3 matrix which is of rank 2, just like the problem in section 3.2.3. Using the same original matrix, we compute

$$AA^{\mathrm{T}} = \begin{bmatrix} 77 & 32 & 64 \\ 32 & 14 & 28 \\ 64 & 28 & 56 \end{bmatrix}, A^{\mathrm{T}}A = \begin{bmatrix} 21 & 30 & 39 \\ 30 & 45 & 60 \\ 39 & 60 & 81 \end{bmatrix} \, . \tag{3.31}$$

Decomposing AA^{T} using eigendecomposition, we find

$$AA^{\mathrm{T}} = \begin{bmatrix} 0.7242 & -0.6896 & -0.0000 \\ 0.3084 & 0.3239 & 0.8944 \\ 0.6168 & 0.6477 & -0.4472 \end{bmatrix} \begin{bmatrix} 145.1397 & 0 & 0 \\ 0 & 1.8603 & 0 \\ 0 & 0 & -0.0000 \end{bmatrix} \times \begin{bmatrix} 0.7242 & 0.3084 & 0.6168 \\ -0.6896 & 0.3239 & 0.6477 \\ -0.0000 & 0.8944 & -0.4472 \end{bmatrix} \, . \tag{3.32}$$

Similarly, the eigendecomposition of $A^{\mathrm{T}}A$ is

$$A^{\mathrm{T}}A = \begin{bmatrix} 0.3684 & -0.8352 & -0.4082 \\ 0.5565 & -0.1536 & 0.8165 \\ 0.7447 & 0.5280 & -0.4082 \end{bmatrix} \begin{bmatrix} 145.1397 & 0 & 0 \\ 0 & 1.8603 & 0 \\ 0 & 0 & -0.0000 \end{bmatrix} \times \begin{bmatrix} 0.3684 & 0.5565 & 0.7447 \\ -0.8352 & -0.1536 & 0.5280 \\ -0.4082 & 0.8165 & -0.4082 \end{bmatrix} \, . \tag{3.33}$$

[8] All numerical packages that compute SVD do this type of sorting

Choosing U from the eigenvectors of AA^{T} and V from the eigenvectors of $A^{\mathrm{T}}A$, and taking the square roots of the diagonal elements of the center matrix of either one, we can write

$$U\Lambda V^{\mathrm{T}} = \begin{bmatrix} 0.7242 & -0.6896 & -0.0000 \\ 0.3084 & 0.3239 & 0.8944 \\ 0.6168 & 0.6477 & -0.4472 \end{bmatrix} \begin{bmatrix} 12.0474 & 0 & 0 \\ 0 & 1.3639 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.3684 & 0.5565 & 0.7447 \\ -0.8352 & -0.1536 & 0.5280 \\ -0.4082 & 0.8165 & -0.4082 \end{bmatrix}, \tag{3.34}$$

which is the SVD of A. Thinking about the null space of A, we realize that the null vector (in this particular case) of A is the column of V (or row of V^{T}) corresponding to the zero singular value.

3.3 Introduction to Function Minimization

Minimization[9] of functions is a pervasive element of engineering: One is always trying to find the set of parameters which minimizes some function of those parameters. Notationally, we state the problem as: Find the vector $\hat{\mathbf{x}}$ which produces a minimum of some function $H(\mathbf{x})$:

$$\hat{H} = \min_{\hat{\mathbf{x}}} H(\mathbf{x}) \tag{3.35}$$

where $\mathbf{x}$ is some d-dimensional parameter vector, and H is a scalar function of $\mathbf{x}$, often referred to as an *objective function*. We denote the $\mathbf{x}$ which results in the minimal H as $\hat{\mathbf{x}}$

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} H(\mathbf{x}). \tag{3.36}$$

There are two distinct reasons for learning how to minimize a function:

- We may need to find a way to perform some process which has a minimal cost, minimal run time, minimal programming difficulty, etc.
- We may need to solve some problem that we don't (immediately) know how to solve. For example, we might seek to draw a contour in an image which separates dark areas from light areas with the additional desire that the regions should be as smooth as possible.

[9] In this book, essentially EVERY computer vision topic will be discussed in terms of some sort of minimization, so get used to it!