

Linear Algebra: Orthogonality

Daven Farnham

August 2015

1 Orthogonality

Two vectors v and w are orthogonal if they are perpendicular to each other. The two vectors' dot product, then, is zero.

The dot product can be thought of as a measure of the degree to which two vectors point in the same direction. Perpendicularity, and a dot product of 0, therefore, denote that the two vectors have no overlap; if you were to find the projection of v onto w , it would be zero.

The matrix of projection onto the x-axis is:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad (1)$$

since given any vector in R^2 , $[x \ y]$, this will isolate the x component.

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 5 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad (2)$$

Let's say you're given some random vector, say $[4 \ 3]$. Find the matrix of projection onto this vector.

To create the matrix of projection start with some vector, w , parallel to $[4 \ 3]$, multiplied by a scalar. You can choose any vector w , but then you'll have to scale it to correctly represent the projection.

There's much more to this, but I've done it a number of times on paper. With almost any problem where you're trying to determine a formula for projections, decompose the vector being projected into components: $x^{\parallel} + x^{\perp}$. The perpendicular vector, then, will be orthogonal to some subspace V while the parallel component will be in the subspace. Find an equation for the perpendicular component, then multiply this by any vector u_i in the subspace to get zero, then solve.

If you have an **orthonormal** basis of some subspace, then the projection of some vector x onto the subspace is:

$$proj_V(x) = (x \cdot u_1)u_1 + \dots + (x \cdot u_n)u_n.$$

Thinking of this intuitively, you're just scaling each of the unit vectors by the scalar $(x \cdot u_i)$ to get the correct projection.

If you weren't dealing with orthonormal vectors, then the equation for the projection of a particular vector, x , onto v , given a vector w parallel to v is:

$$\frac{(x \cdot w)}{w \cdot w} w = proj_v x$$

Orthonormal vectors simply make everything easier since they eliminate the denominator from the equation above.

1.1 Uniqueness

Previously, it was proved that if you have a basis of some subspace, meaning the only linear relation between the vectors is the trivial one, then any representation of some vector x in the subspace is unique. Proving by contradiction, assume there are two ways to represent the same vector:

$$\begin{aligned} a_1 v_1 + \dots + a_n v_n &= x \\ b_1 v_1 + \dots + b_n v_n &= x \\ a_1 v_1 + \dots + a_n v_n &= b_1 v_1 + \dots + b_n v_n \\ (a_1 - b_1) v_1 + \dots + (a_n - b_n) v_n &= 0 \\ a_n &= b_n \end{aligned}$$

You know this since you're dealing with linearly independent vectors forming a basis of the subspace V ; the only way you'll get a linear relation is if it's the trivial one, and that'll only happen when a_n equals b_n .

So given some basis of a subspace V , there is **only one, unique representation of a vector x in that subspace.**

If I'm given some matrix A in R^n , with column vectors v_1, v_2, \dots, v_n , then to create some other vector x in R^n I need to find the coefficients of the column vectors. For example:

$$A = \begin{bmatrix} 1 & 4 \\ 2 & 7 \end{bmatrix} \tag{3}$$

$$x_1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} + x_2 \begin{bmatrix} 4 \\ 7 \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \end{bmatrix} \tag{4}$$

$$32 \begin{bmatrix} 1 \\ 2 \end{bmatrix} - 8 \begin{bmatrix} 4 \\ 7 \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \end{bmatrix} \quad (5)$$

I solved the above through Gaussian Elimination, but if the vectors you're dealing with are orthonormal, it's much easier to find the coefficients to the linear combination. If, say, I had a different matrix B :

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (6)$$

It's obvious when working with the standard vectors that the answer is going to be 0 and 8. However, given some non-obvious orthonormal vectors, simply find the projection of $\begin{bmatrix} 0 & 8 \end{bmatrix}$ onto each of the basis vectors and that'll give you the coefficients.

$$\left(\begin{bmatrix} 0 \\ 8 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) = 0 \quad (7)$$

$$\left(\begin{bmatrix} 0 \\ 8 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) = 8 \quad (8)$$

The projection equation for orthonormal vectors makes use of this fact; it is simply adding up the projection of a particular vector onto each of the basis vectors.

1.2 Least-Squares and Approximations

Orthogonality and projections come in handy when trying to approximate an equation for some system that **isn't consistent**. A system that's inconsistent is one that can't be solved through Gaussian elimination: you'll end up with some row where $0 = !0$.

Let's say I have some matrix A and another vector, b , that isn't a member of the image of A . This means $A\vec{x} = \vec{b}$ will be an inconsistent system of equations, since no coefficient vector \vec{x} will give you \vec{b} .

The goal, then, is to find a coefficient vector x^* that'll give you a vector in the image of A that is as close as possible to b . This vector has to be the projection of b onto the image of A . So, you want to find the vector that minimizes the error from $b - Ax^*$:

$$\begin{aligned} b - Ax^* \\ A^T(b - Ax^*) &= 0 \\ A^Tb - A^TAx^* &= 0 \\ A^Tb &= A^TAx^* \end{aligned}$$

The reason you can use the transpose of the A matrix is because you know Ax^* is the $proj_{Im(A)}b$. Therefore, $b - Ax^*$ will give you a vector that is orthogonal to the image of A . This vector, dotted with any vector in the Image of A , will give you zero.

$$image(A) = \{v_1, v_2, v_3\} = \begin{bmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{bmatrix} \quad (9)$$

If you dot each one of these column vectors with a vector orthogonal to the image, you'll get zero. You could write it like this:

$$\begin{bmatrix} v_1 \end{bmatrix} \cdot \begin{bmatrix} | \\ (b - Ax^*) \\ | \end{bmatrix} = 0 \quad (10)$$

$$\begin{bmatrix} v_2 \end{bmatrix} \cdot \begin{bmatrix} | \\ (b - Ax^*) \\ | \end{bmatrix} = 0 \quad (11)$$

$$\begin{bmatrix} v_3 \end{bmatrix} \cdot \begin{bmatrix} | \\ (b - Ax^*) \\ | \end{bmatrix} = 0 \quad (12)$$

But an easier way is to notice that, if you flip the matrix and use its transpose, then you can use simple matrix-vector multiplication:

$$\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ - & v_3 & - \end{bmatrix} \begin{bmatrix} | \\ (b - Ax^*) \\ | \end{bmatrix} = 0 \quad (13)$$

1.2.1 Examples

This comes in handy in any situation where you don't have a consistent system. Let's say you have a scatter plot of 3 points: $\{1, 2\}$, $\{0, 3\}$, $\{2, 8\}$. You can't draw a line between these. The slope between $\{1, 2\}$ and $\{0, 3\}$ is -1 while the slope between $\{1, 2\}$ and $\{2, 8\}$ is 4. Instead, let's write this as a system of equations.

Using $mx + b = y$:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 8 \end{bmatrix} \quad (14)$$

Use the above least-squares equation to approximate a line for the plot:

$$\begin{bmatrix} 1 & 0 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 2 & 1 \end{bmatrix} x^* \quad (15)$$

Solving for x^* you find the coefficient vector for m and b that creates a vector within the image of the matrix (i.e. makes the system consistent). So this coefficient vector in R^2 , when multiplied by the matrix, creates a vector in R^3 that makes the system consistent and is the closest possible approximation to $\begin{bmatrix} 2 & 3 & 8 \end{bmatrix}$.

That two dimensional vector, then, (in this case $\begin{bmatrix} 5 \\ 2 \end{bmatrix}, \begin{bmatrix} 11 \\ 6 \end{bmatrix}$) represents the line that most closely, or fits best, those nonlinear points. It makes a vector in R^3 that makes the matrix consistent and is the closest approximation to $\begin{bmatrix} 2 & 3 & 8 \end{bmatrix}$.

See Mathematica Notebook "Approximations".

1.2.2 Conclusion

You don't just have to use least squares approximations for straight lines. You can also approximate curves with this method. Overall, this is a super powerful method; you're eliminating noise and trying to match a best fit line to the data by minimizing error.