Generalized Rotation, Grand and Random Tours

By
Daniel B. Carr
George Mason University

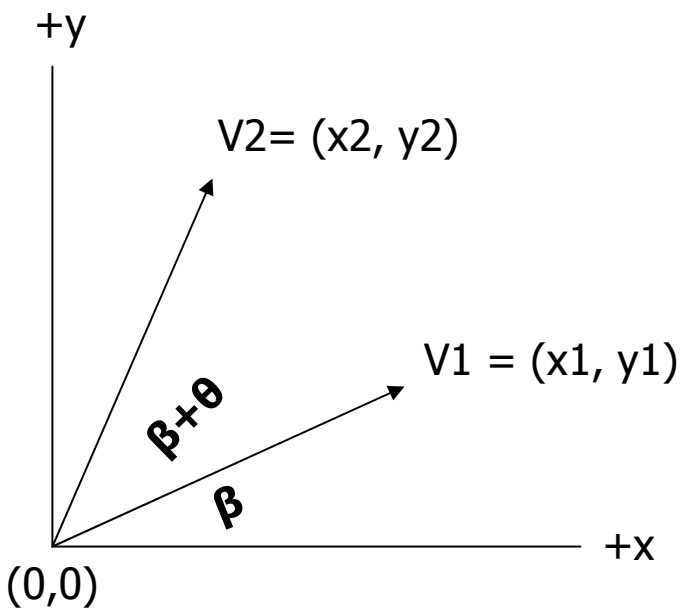
2-D Rotation Via Double Angle Formulas

$r = \sqrt{x1^2 + y1^2} = ||V1||$

$r = \sqrt{x2^2 + y2^2} = ||V2||$

$x1 = r*\cos(\beta)$ **$y1 = r*\sin(\beta)$**

$x2 = r*\cos(\beta + \theta)$ $y2 = \sin(\beta + \theta)$



$\cos(\beta + \theta) = \cos(\beta)\cos(\theta) - \sin(\beta)\sin(\theta)$

$\sin(\beta + \theta) = \sin(\beta)\cos(\theta) + \cos(\beta)\sin(\theta)$

$x2 = r*(\cos(\beta)\cos(\theta) - \sin(\beta)\sin(\theta))$

$x2 = \mathbf{r*\cos(\beta)}\cos(\theta) - \mathbf{r*\sin(\beta)}\sin(\theta)$

$x2 = x1*\cos(\theta) - y1*\sin(\theta)$

$x2 = \cos(\theta)*x1 - \sin(\theta)*y1$

$y2 = r*(\sin(\beta)\cos(\theta) + \cos(\beta)\sin(\theta))$

$y2 = \mathbf{r*\sin(\beta)}\cos(\theta) + \mathbf{r*\cos(\beta)}\sin(\theta)$

$y2 = y1*\cos(\theta) + x1*\sin(\theta)$

$y2 = \sin(\theta)*x1 + \cos(\theta)*y1$

Rotation in 2-D By Matrix Multiplication

- Rotating point V1=(x1,y1) theta degrees into point V2 = (x2, y2)
- The 2 x 2 matrix on the right multiplies the column vector on the right
- The rotation matrix is a Givens matrix

$$\begin{bmatrix} x2 \\ y2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & = \cos(\theta) \end{bmatrix} * \begin{bmatrix} x1 \\ y1 \end{bmatrix}$$

Rotation in 3-D About the Z axis

- In 3-D we can rotate θ degrees about the Z axis keeping the z coordinate fixed
- Again we can use a Givens rotation matrix
- Below ∗ stands for matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotation in 3-D About the Y-axis

- We can rotate about the Y-axis with a Givens rotation matrix
- Consider the point (1, 0, 0) and rotate θ = 90 degrees. Is the result (0, 0, 1)?

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotation in 3-D About the Y-axis

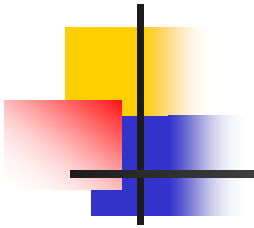
- We want +X to rotate into +Z
- Note the upper right position of –sin(θ)

+Z							+X
0		cos(90)= 0	0	-sin(90)= -1			1
0	=	0	1	0	*		0
1		sin(90)=1	0	cos(90)=0			0

Rotation in 3-D About the X-axis

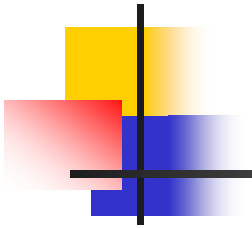
- With a Givens rotation matrix we rotate +Y into +Z so –sin(θ) is on the lower left
- Note that the point (0, 1, 0) when rotated θ = 90 degrees results in (0, 0, 1)

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



Observations

- The determinant of rotation matrices is always 1
  - Otherwise the transformation would expand/contract an image composed of points.
- Our rotations were implicitly relative to the origin (0,0,0)
  - To rotate about a point, translate the data so that point becomes the origin, rotate, and then translate back
  - With natural homogeneous coordinates all this can be done with three matrix multiplications.
    - (Translate_back * Rotate * Translate) * Data
      - Here each column in Data is a case. We often use the transpose
    - Matrix multiplication is associative so we can multiply the little matrices first to save on intermediate storage. Data can be big
- Aligning one vector from the origin with another from origin
  - This can always be done with two rotations about different fixed axes. (Multiplying two rotation matrices give us a single matrix for direction rotation)



Moving on to 4-D and Higher

- In this class we use right hand coordinates for 3-D
  - The right hand thumb points to the right for +x
  - The index finger points up for +y
  - The middle finger when bent the way the joint was designed points towards us, +z
- In 4-D and higher dimensions in the class
  - We don’t bother with such issues
  - We don’t worry about switching signs of sines

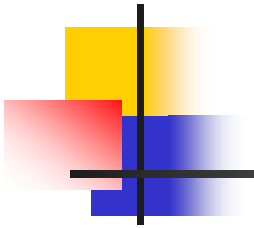
A 4-D Givens Rotation for a Fixed Y and Z Subspace

X′	=	cos(θ)	0	0	sin(θ)	*	X
Y′		0	1	0	0		Y
Z′		0	0	1	0		Z
W′		-sin(θ)	0	0	cos(θ)		W

A 4-D Givens Rotation for a Fixed Y and W Subspace

X′	=	cos(θ)	0	sin(θ)	0	*	X
Y′		0	1	0	0		Y
Z′		-sin(θ)	0	cos(θ)	0		Z
W′		0	0	0	1		W

Generalized Rotation in d-Dimensions



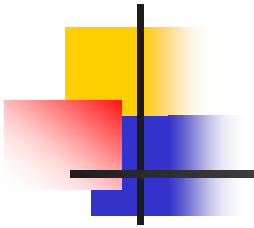
- Create Givens matrices for all pairs of variables in d-dimensions
- The rotation matrix product of all the individual Givens matrices
  - In 4D
    - 4 variables choose 2 pairs = 6 matrices
    - We need 6 rotation angles, one for each matrix
    - The product of the six matrices and align two 4-D vectors from form the origin.
  - In 5D
    - 5 choose 2 pairs = 10 matrices
    - We need 10 angles, one for each matrix

One composite rotation matrix for 4 Dimensions

- $M_{4d} = M_{xy} * M_{xz} * M_{xw} * M_{yz} * M_{yw} * M_{zw}$
- Indexing by the fixed subspace

$$M_{YZ} = \begin{vmatrix} \cos(a4) & 0 & 0 & -\sin(a4) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin(a4) & 0 & 0 & \cos(a4) \end{vmatrix}$$

How can we generate a sequence of six angles to get close the all possible views ?



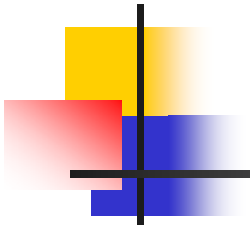
The Grand Tour

- Developed by Asimov (1985) and Asimov and Buja (1985)
  - Define a time sequence of 2-D views whose basis vectors come arbitrarily close to any two-D plane through the origin
    - The collection of 2-D planes through the origin is a Grassmannian manifold
  - There are several methods
  - The torus method
    - This uses the special orthogonal group, denoted SO(d), of matrices with determinant 1 to transform d basis vectors and produce new coordinates.
    - We a sequence of angles to form is a continuous space filling path through SO(d)

A Space Filling Path Through SO(d)

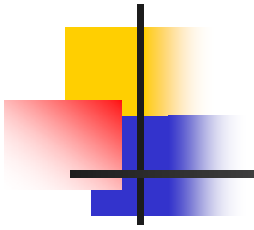
- Let k = d choose 2
- Let $t^*(a_1, a_2, \ldots, a_k)$ base 2*pi be angles used at time t
- Pick the angles as the square root of prime numbers
  - In theory the sequence will never repeat
  - In practice finite precision could lead to repeats

How Many Views Are there?



- Squint angle fractions from Tukey and Tukey (1981)
  - Fraction of a full (d-1) within 5 degrees of any specified direction
    - d=4: 1/526
    - d=5: 1/92196
    - d=7: 1/14560051
    - d=9: 1/2190180925
  - Of course fitting caps together is another issue.
    - Can you generate n equally space points on a d-dimensional sphere for an arbitrary n?

2-D Random Tour

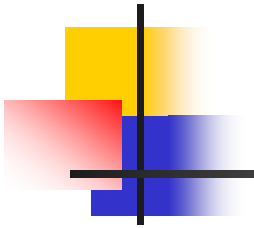


- Let D be a n x d data set with n cases and d variables
  - This assumes that D has been suitably transformed so linear combinations make sense
  - We discuss spherizing and other transformations later
- Construct two d x 2 matrices A1 and A2
  - Each matrix has two orthonormal column vectors
  - Use Gram-Schmidt, Cholesky decomposition, or even regression to make two random vectors of length d orthogonal (dot product = 0) and normalize them (sum of squares = 1)
- Plot all point pairs (rows) of B where
  - $B_{n \times 2} = D_{n \times d} * (\cos(\theta)*A1 + \sin(\theta)*A2)_{d \times 2}$
  - Varying values of θ from 0 to pi/2
  - Use double buffering to avoid flashes
- When a =90 degrees
  - Replace A1 with A2
  - Create a new random orthonormal basis to replace A2
  - Set θ = 0 and step through values of θ, plotting from B as before

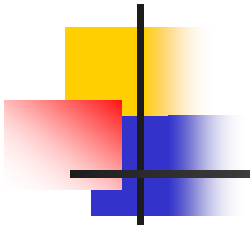
A reminder on Gram-Schmidt Orthogonalization

- Let x1 and x2 be two column vectors
- Normalize x1
  - $x1new = x1/sqrt(x1^T x1)$
- Get a, the regression coefficient
  - $a = x2^T x1new$
- Get residuals
  - res = x2 - a*x1new
- Normalize the residuals
  - $X2new = res/sqrt(res^T res)$

Viewing Tours in More Dimensions

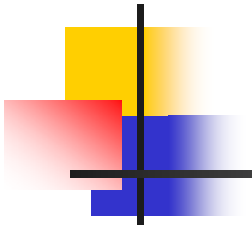


- The original tours used just 2 coordinates and viewed using 2-D scatterplots
- Dr. Edward Wegman developed generalized tours
  - The grand tour was just picking off the first two coordinates
  - The random tour can have d orthonormal basis vectors in the matrices A1 and A2 and produce d new coordinates
  - He finds interesting patterns quickly even though there are a huge number of squint angles
- Many graphics can show more 2 coordinates
  - A 3-D scatterplot can show triples
  - A stereo ray glyph shows 4 coordinates (Carr and Nicholson 1988, Explor4)
    - The ray angle is always shown in the plane of the display and the angle range limited to 180 degrees
  - Scatterplot matrices can show all pairs of the d coordinates
  - Parallel coordinates sequence of pairs of the d coordinates
- First implementation: ExplorN (SGI) by Carr, Luo and Wegman in1992
  - CrystalVision, a port to Windows by Luo and Wegman has most of this
    - The ray implementation with range from 0 to 360 degrees is confusing
    - No hyperplane slicing, etc.



Continuing Advances

- Whip spin control by Cook and Buja
  - (need to locate reference)
  - In 2-D plots whip spin (spinning about the origin) is not helpful in seeing patterns
- Projection pursuit methods
  - See XGobi and Ggobi
- Missing data methods
- Image/pixel tours: Wegman and Luo



Projection Pursuit

- Projection pursuit methods
  - Define a measure of interest computed for the plots to be view
    - Clottedness, holes, etc
    - Progressively modify the basis vectors to increase this measure
  - Your instructor prefers projection pursuit methods when available
    - Likes the holes algorithm for less overplotting
    - Still uses random tour to get starting points
    - Notes that random image/pixels tours can be very informative
      - Are project pursuit methods are available?
  - Will be discussed more later