Machine Learning for Signal Processing

Fundamentals of Linear Algebra - 2

Class 3. 5 Sep 2013

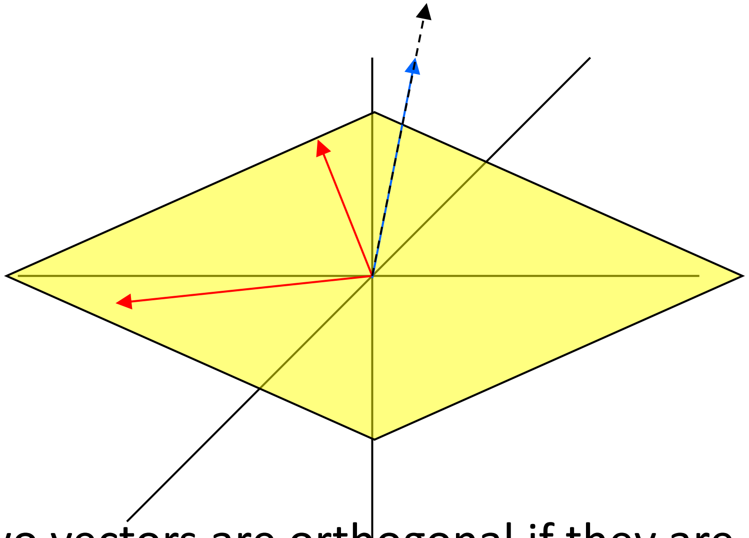
Instructor: Bhiksha Raj

Overview

- Vectors and matrices
- Basic vector/matrix operations
- Various matrix types
- Projections

- More on matrix types
- Matrix determinants
- Matrix inversion
- Eigenanalysis
- Singular value decomposition

Orthogonal/Orthonormal vectors



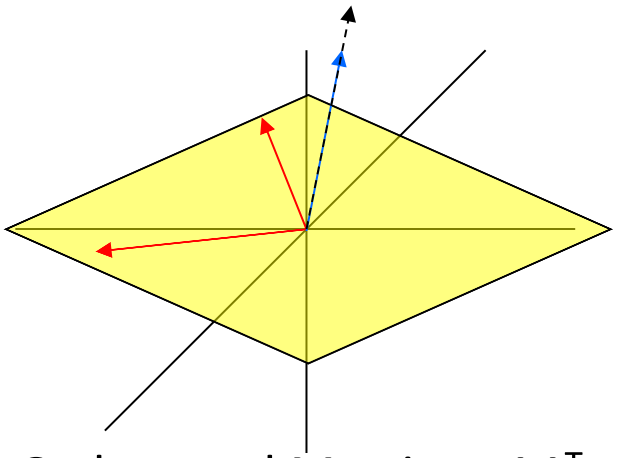
$$A = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad B = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$A.B = 0 \quad \Rightarrow \quad xu + yv + zw = 0$$

- Two vectors are orthogonal if they are perpendicular to one another
  - A.B = 0
  - A vector that is perpendicular to a plane is orthogonal to *every* vector on the plane

- Two vectors are ortho*normal* if
  - They are orthogonal
  - The length of each vector is 1.0
  - Orthogonal vectors can be made orthonormal by normalizing their lengths to 1.0

Orthogonal matrices



$$\begin{bmatrix} \sqrt{0.5} & -\sqrt{0.125} & \sqrt{0.375} \\ \sqrt{0.5} & \sqrt{0.125} & -\sqrt{0.375} \\ 0 & \sqrt{0.75} & 0.5 \end{bmatrix}$$

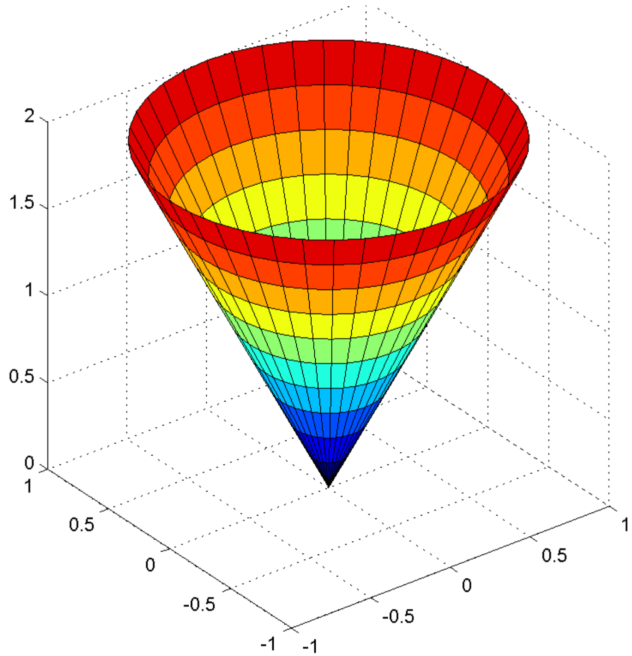
- Orthogonal Matrix : $AA^T = A^TA = I$
  - The matrix is square
  - All row vectors are orthonormal to one another
    - Every vector is perpendicular to the hyperplane formed by all other vectors
  - All column vectors are also orthonormal to one another
  - **Observation:** In an orthogonal matrix if the length of the row vectors is 1.0, the length of the column vectors is also 1.0
  - **Observation**: **In an orthogonal matrix no more than one row can have all entries with the same polarity (+ve or –ve)**

Orthogonal and Orthonormal Matrices

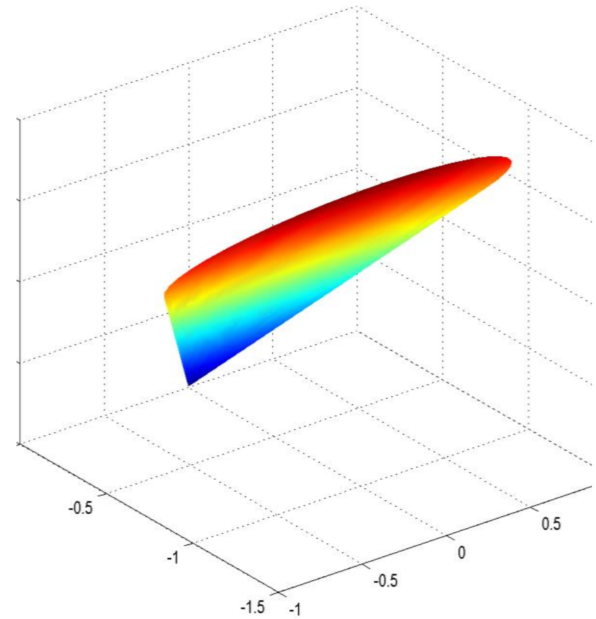
- Orthogonal matrices will retain the **length** and **relative angles between** transformed vectors
  - Essentially, they are combinations of rotations, reflections and permutations
  - Rotation matrices and permutation matrices are all orthonormal

- If the vectors in the matrix are not unit length, it cannot be orthogonal
  - AA^T != I, A^TA != I
  - AA^T = Diagonal or A^TA = Diagonal, but not both
  - If all the entries are the same length, we can get $AA^T = A^TA$ = Diagonal, though
- A non-square matrix cannot be orthogonal
  - AA^T=I or A^TA = I, but not both

Matrix Rank and Rank-Deficient Matrices



P * Cone =

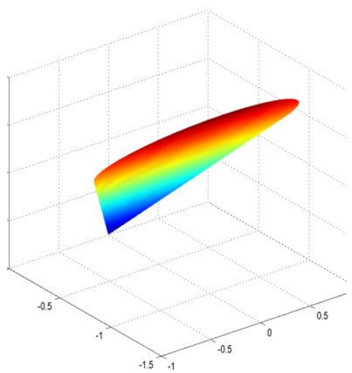


- Some matrices will eliminate one or more dimensions during transformation
  - These are *rank deficient* matrices
  - The rank of the matrix is the dimensionality of the transformed version of a full-dimensional object

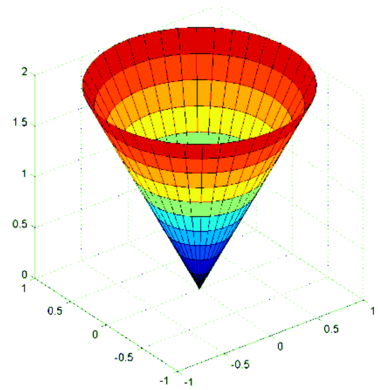
Matrix Rank and Rank-Deficient Matrices

P =

1.0000	0	0
0	0.2500	-0.4330
0	-0.4330	0.7500

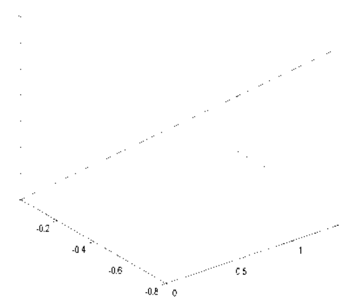


Rank = 2



P2 =

0.5000	-0.2500	0.4330
-0.2500	0.1250	-0.2165
0.4330	-0.2165	0.3750

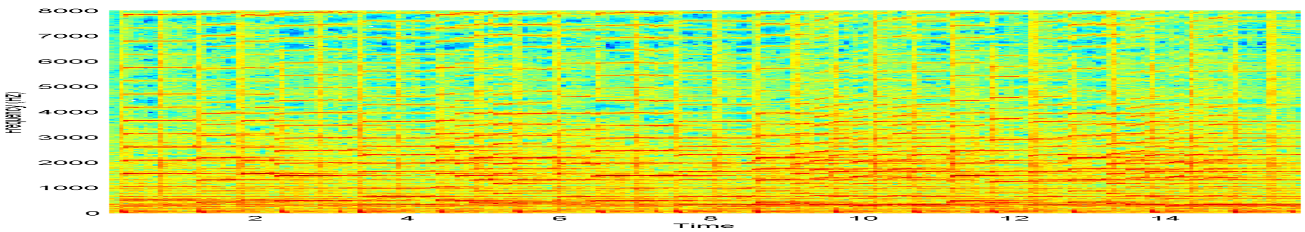


Rank = 1

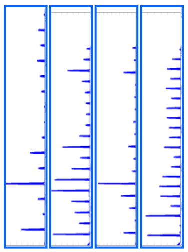
- Some matrices will eliminate one or more dimensions during transformation
  - These are *rank deficient* matrices
  - The rank of the matrix is the dimensionality of the transformed version of a full-dimensional object

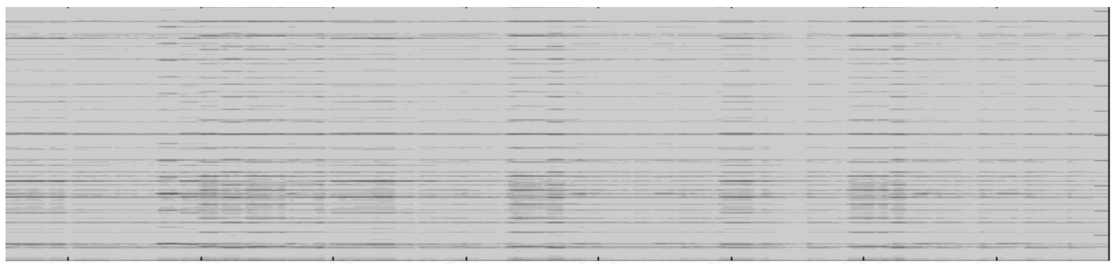
Projections are often examples of rank-deficient transforms

M =



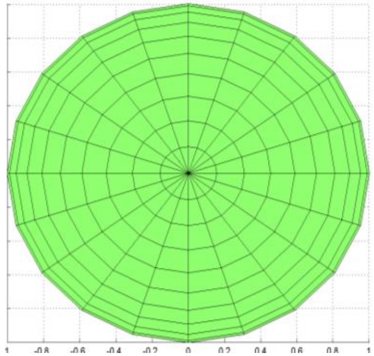
W =

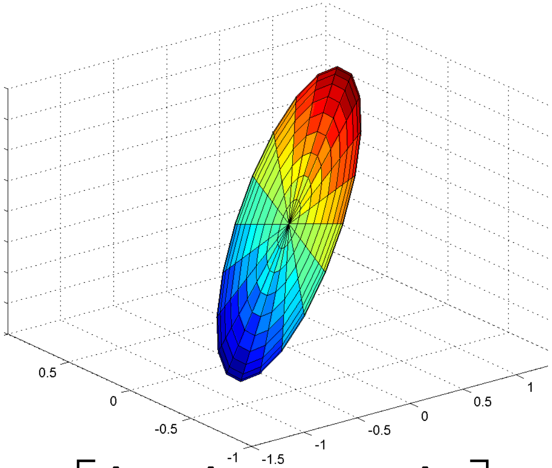




- $P = W (W^T W)^{-1} W^T$; Projected Spectrogram = P*M
- The original spectrogram can never be recovered
  - P is rank deficient
- P explains all vectors in the new spectrogram as a mixture of only the 4 vectors in W
  - There are only a maximum of 4 ***independent*** bases
  - Rank of P is 4

Non-square Matrices





$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \end{bmatrix}$$

X = 2D data

$$\begin{bmatrix} .8 & .9 \\ .1 & .9 \\ .6 & 0 \end{bmatrix}$$

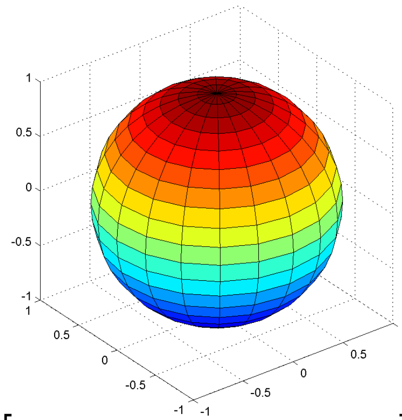
P = transform

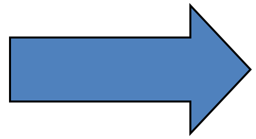
$$\begin{bmatrix} \hat{x}_1 & \hat{x}_2 & . & . & \hat{x}_N \\ \hat{y}_1 & \hat{y}_2 & . & . & \hat{y}_N \\ \hat{z}_1 & \hat{z}_2 & . & . & \hat{z}_N \end{bmatrix}$$

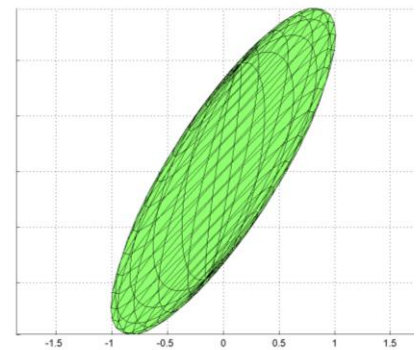
PX = 3D, rank 2

- Non-square matrices add or subtract axes
  - More rows than columns → add axes
    - But does not increase the dimensionality of the data

Non-square Matrices







$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \\ z_1 & z_2 & . & . & z_N \end{bmatrix}$$

X = 3D data, rank 3

$$\begin{bmatrix} .3 & 1 & .2 \\ .5 & 1 & 1 \end{bmatrix}$$

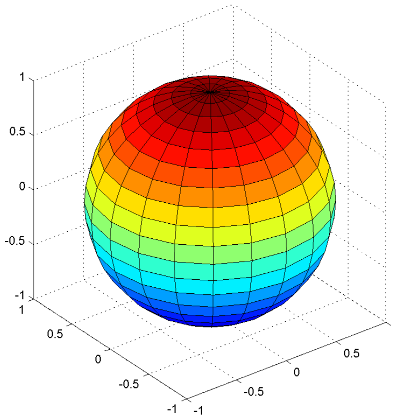
P = transform

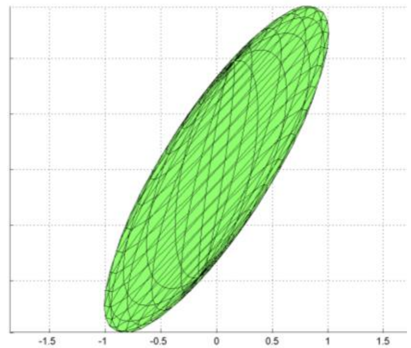
$$\begin{bmatrix} \hat{x}_1 & \hat{x}_2 & . & . & \hat{x}_N \\ \hat{y}_1 & \hat{y}_2 & . & . & \hat{y}_N \end{bmatrix}$$

PX = 2D, rank 2

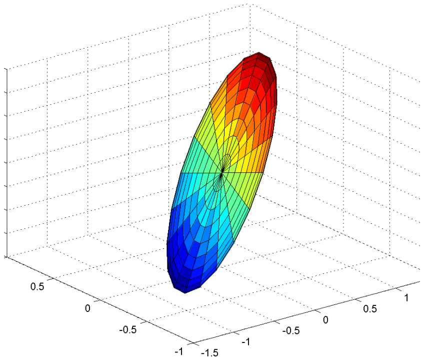
- Non-square matrices add or subtract axes
  - More rows than columns → add axes
    - But does not increase the dimensionality of the data
  - Fewer rows than columns → reduce axes
    - May reduce dimensionality of the data

The Rank of a Matrix





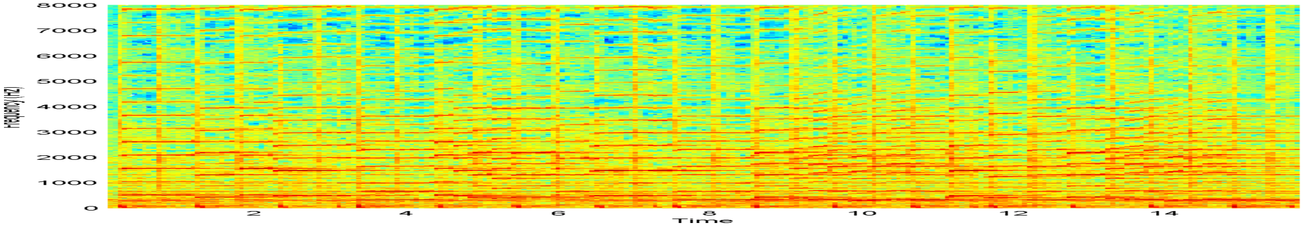
$$\begin{bmatrix} .3 & 1 & .2 \\ .5 & 1 & 1 \end{bmatrix}$$

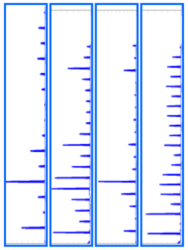


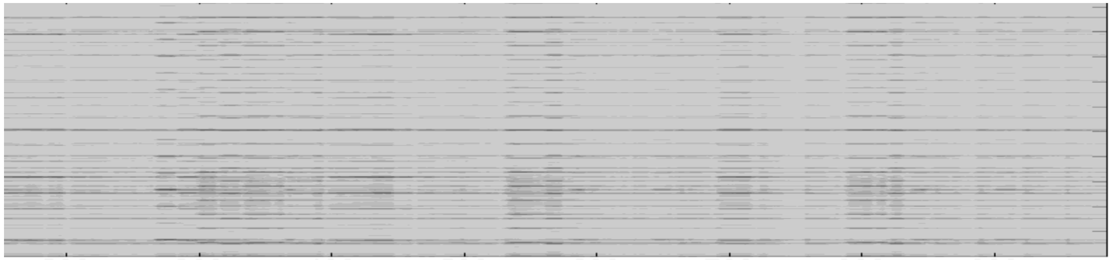
$$\begin{bmatrix} .8 & .9 \\ .1 & .9 \\ .6 & 0 \end{bmatrix}$$

- The matrix rank is the dimensionality of the transformation of a full-dimensioned object in the original space
- The matrix can never *increase* dimensions
  - Cannot convert a circle to a sphere or a line to a circle
- The rank of a matrix can never be greater than the lower of its two dimensions

The Rank of Matrix

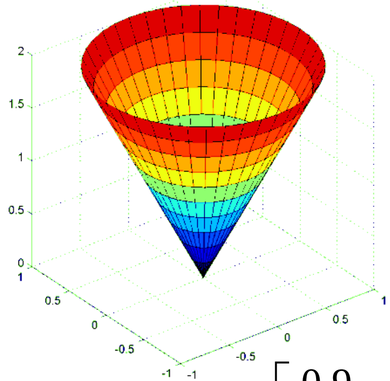
M = 

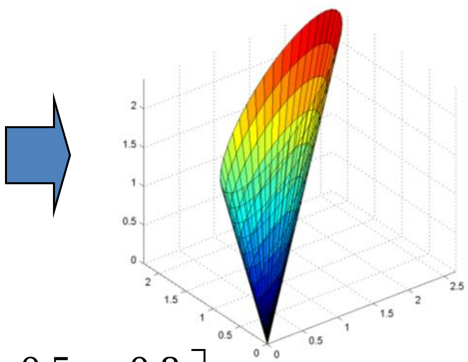




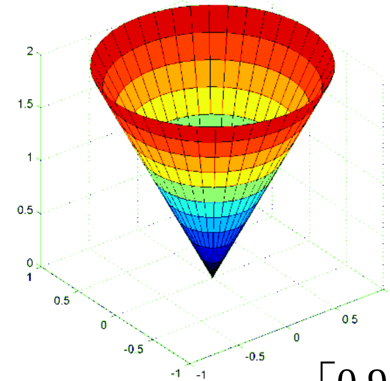
- Projected Spectrogram = P * M
  - Every vector in it is a combination of only 4 bases
- The rank of the matrix is the *smallest* no. of bases required to describe the output
  - E.g. if note no. 4 in P could be expressed as a combination of notes 1,2 and 3, it provides no additional information
  - Eliminating note no. 4 would give us the same projection
  - The rank of P would be 3!

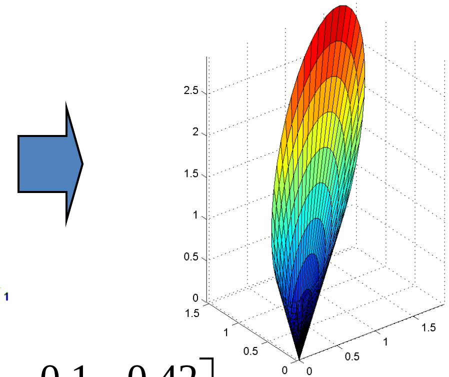
Matrix rank is unchanged by transposition





$$\begin{bmatrix} 0.9 & 0.5 & 0.8 \\ 0.1 & 0.4 & 0.9 \\ 0.42 & 0.44 & 0.86 \end{bmatrix}$$



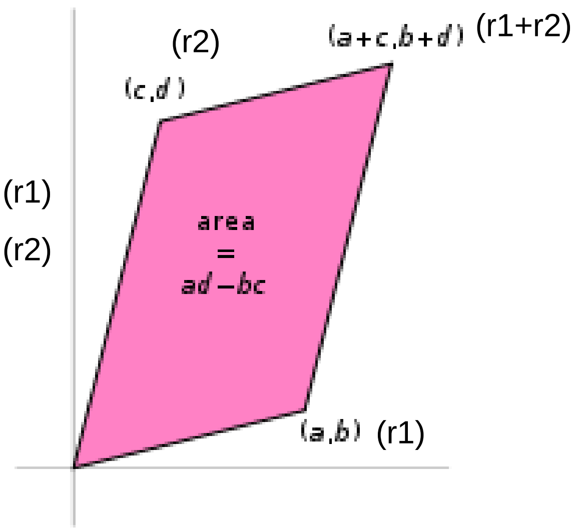


$$\begin{bmatrix} 0.9 & 0.1 & 0.42 \\ 0.5 & 0.4 & 0.44 \\ 0.8 & 0.9 & 0.86 \end{bmatrix}$$

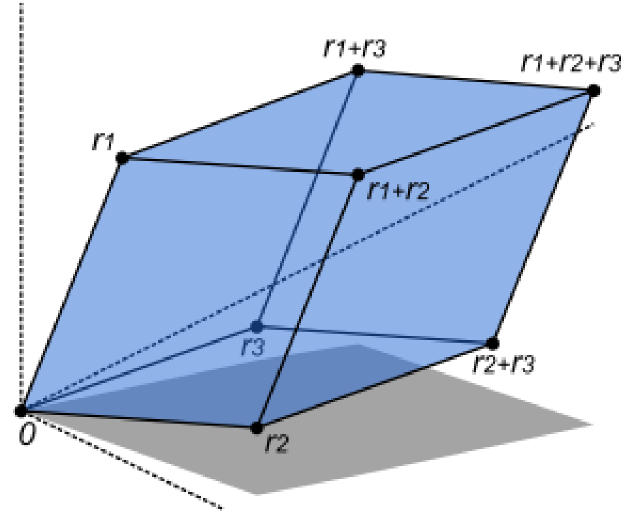
- If an N-dimensional object is compressed to a K-dimensional object by a matrix, it will also be compressed to a K-dimensional object by the transpose of the matrix

Matrix Determinant

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{matrix} \text{(r1)} \\ \text{(r2)} \end{matrix}$$



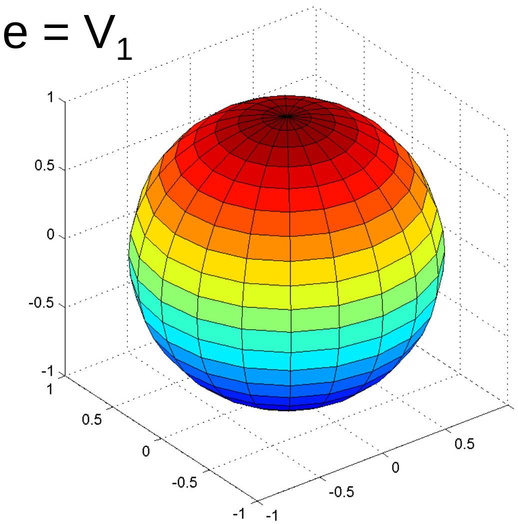
$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$



- The determinant is the "volume" of a matrix
- Actually the volume of a parallelepiped formed from its row vectors
  - Also the volume of the parallelepiped formed from its column vectors
- Standard formula for determinant: in text book

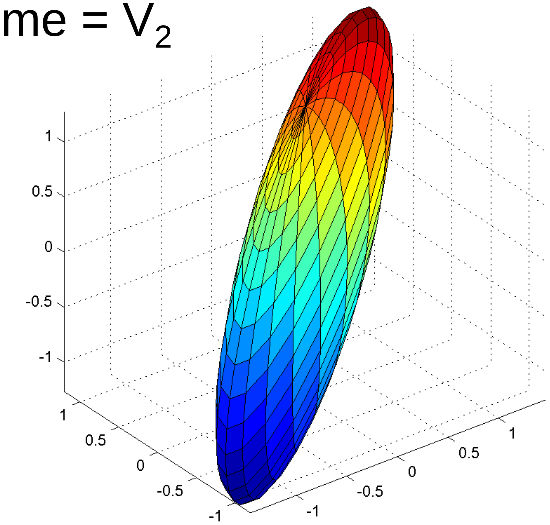
Matrix Determinant: Another Perspective

Volume = V_1



$$\begin{bmatrix} 0.8 & 0 & 0.7 \\ 1.0 & 0.8 & 0.8 \\ 0.7 & 0.9 & 0.7 \end{bmatrix}$$

Volume = V_2



- The determinant is the ratio of N-volumes
  - If V_1 is the volume of an N-dimensional object "O" in N-dimensional space
    - O is the complete set of points or vertices that specify the object
  - If V_2 is the volume of the N-dimensional object specified by A*O, where A is a matrix that transforms the space
  - $|A| = V_2 / V_1$

Matrix Determinants

- Matrix determinants are *only defined for square matrices*
  - They characterize volumes in linearly transformed space of the same dimensionality as the vectors

- Rank deficient matrices have determinant 0
  - Since they compress full-volumed N-dimensional objects into zero-volume N-dimensional objects
    - E.g. a 3-D sphere into a 2-D ellipse: The ellipse has 0 volume (although it does have area)

- Conversely, all matrices of determinant 0 are rank deficient
  - Since they compress full-volumed N-dimensional objects into zero-volume objects

Multiplication properties

- Properties of vector/matrix products
  - Associative

$$\mathbf{A}\cdot(\mathbf{B}\cdot\mathbf{C}) = (\mathbf{A}\cdot\mathbf{B})\cdot\mathbf{C}$$

  - Distributive

$$\mathbf{A}\cdot(\mathbf{B}+\mathbf{C}) = \mathbf{A}\cdot\mathbf{B}+\mathbf{A}\cdot\mathbf{C}$$

  - NOT commutative!!!

$$\mathbf{A}\cdot\mathbf{B} \neq \mathbf{B}\cdot\mathbf{A}$$

    - *left multiplications ≠ right multiplications*
  - Transposition

$$(\mathbf{A}\cdot\mathbf{B})^T = \mathbf{B}^T\cdot\mathbf{A}^T$$

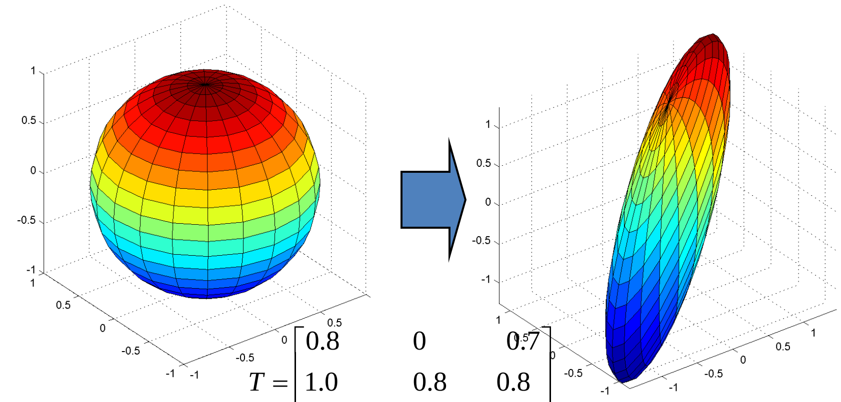
Determinant properties

- Associative for square matrices $$|\mathbf{A}\cdot\mathbf{B}\cdot\mathbf{C}| = |\mathbf{A}|\cdot|\mathbf{B}|\cdot|\mathbf{C}|$$
  - Scaling volume sequentially by several matrices is equal to scaling once by the product of the matrices
- Volume of sum != sum of Volumes $$|(\mathbf{B}+\mathbf{C})| \neq |\mathbf{B}| + |\mathbf{C}|$$
- Commutative
  - The order in which you scale the volume of an object is irrelevant

$$|\mathbf{A}\cdot\mathbf{B}| = |\mathbf{B}\cdot\mathbf{A}| = |\mathbf{A}|\cdot|\mathbf{B}|$$

Matrix Inversion

- A matrix transforms an N-dimensional object to a different N-dimensional object

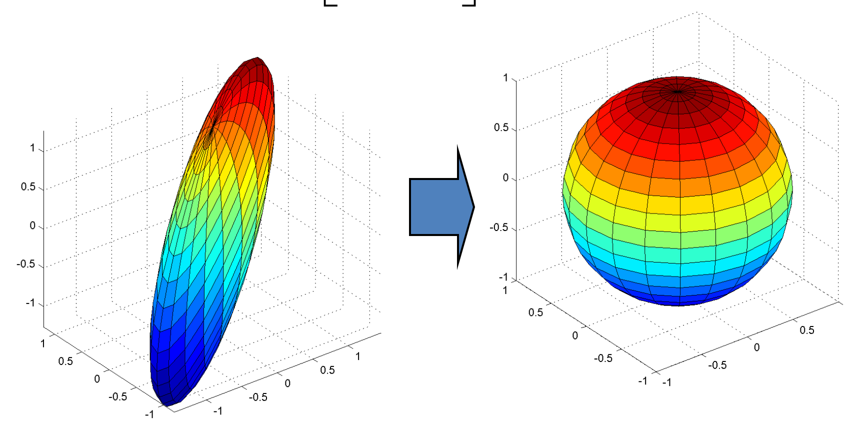


$$T = \begin{bmatrix} 0.8 & 0 & 0.7 \\ 1.0 & 0.8 & 0.8 \\ 0.7 & 0.9 & 0.7 \end{bmatrix}$$

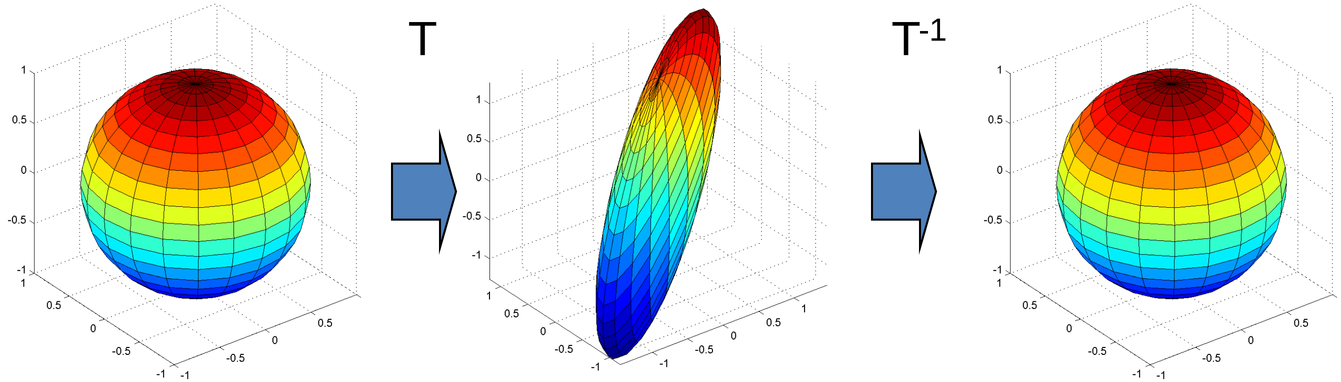
- What transforms the new object back to the original?
  - The *inverse transformation*

$$Q = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix} = T^{-1}$$

- The inverse transformation is called the matrix inverse



Matrix Inversion

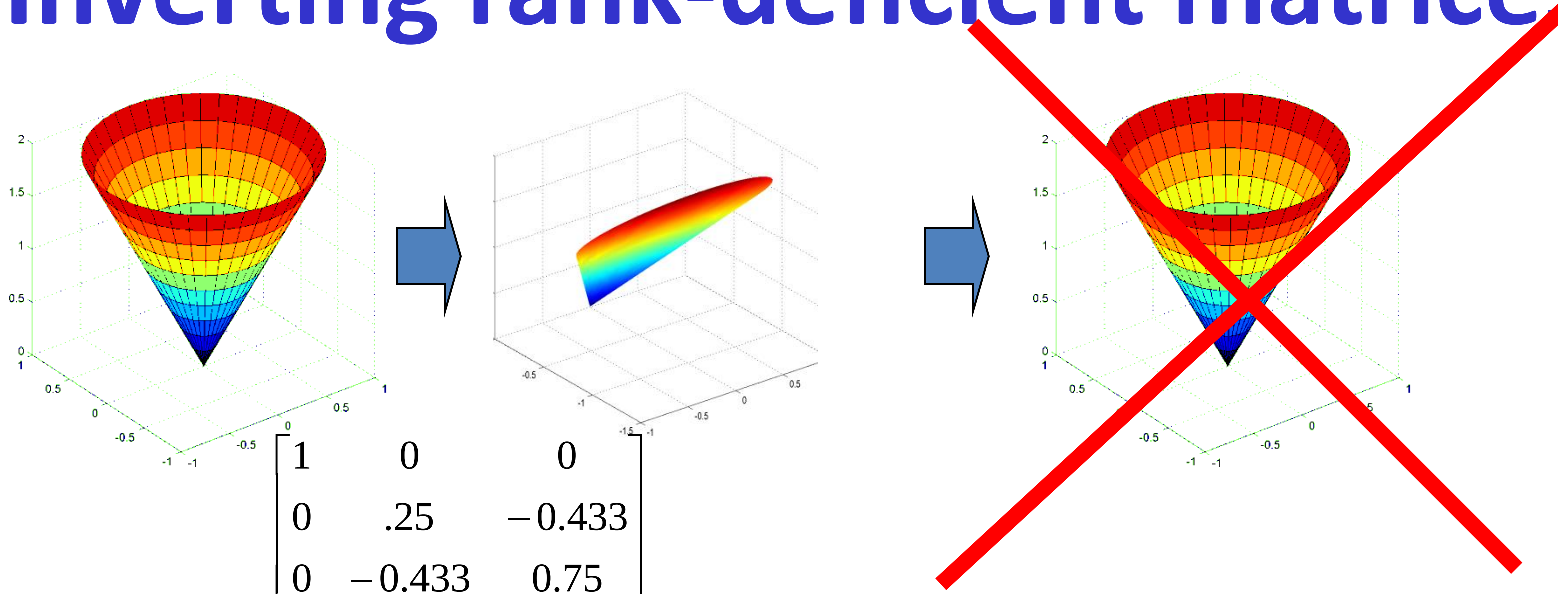


$T^{-1}*T*D = D \rightarrow T^{-1}T = I$

- The product of a matrix and its inverse is the identity matrix
  - Transforming an object, and then inverse transforming it gives us back the original object

$T*T^{-1}*D = D \rightarrow TT^{-1} = I$

Inverting rank-deficient matrices



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & .25 & -0.433 \\ 0 & -0.433 & 0.75 \end{bmatrix}$$

- Rank deficient matrices “flatten” objects
  - In the process, multiple points in the original object get mapped to the same point in the transformed object
- It is not possible to go “back” from the flattened object to the original object
  - Because of the many-to-one forward mapping
- Rank deficient matrices have no inverse

Revisiting Projections and Least Squares

- Projection computes a *least squared error* estimate
- For each vector V in the music spectrogram matrix
  - Approximation: V_{approx} = a*note1 + b*note2 + c*note3..

$$T = \begin{bmatrix} \text{note1} & \text{note2} & \text{note3} \end{bmatrix} \qquad V_{approx} = T\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

  - Error vector E = V – V_{approx}
  - Squared error energy for V e(V) = norm(E)^2
- Projection computes V_{approx} for all vectors such that Total error is minimized
- **But WHAT ARE "a" "b" and "c"?**

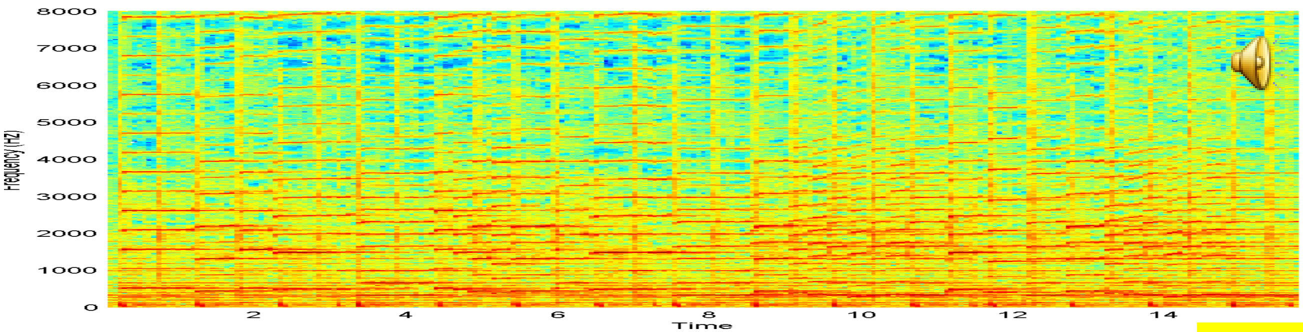
The Pseudo Inverse (PINV)

$$V_{approx} = T\begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow V \approx T\begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow \begin{bmatrix} a \\ b \\ c \end{bmatrix} = PINV(T) * V$$

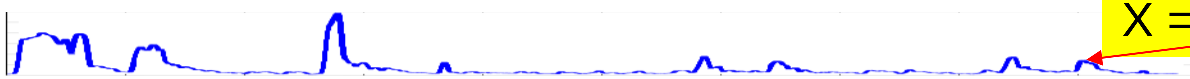
- We are approximating spectral vectors V as the transformation of the vector $[a\ b\ c]^T$
  - Note – we're viewing the collection of bases in T as a transformation
- The solution is obtained using the *pseudo inverse*
  - This give us a *LEAST SQUARES* solution
    - If T were square and invertible Pinv(T) = T^{-1}, and $V=V_{approx}$

Explaining music with one note

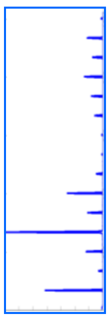
M =

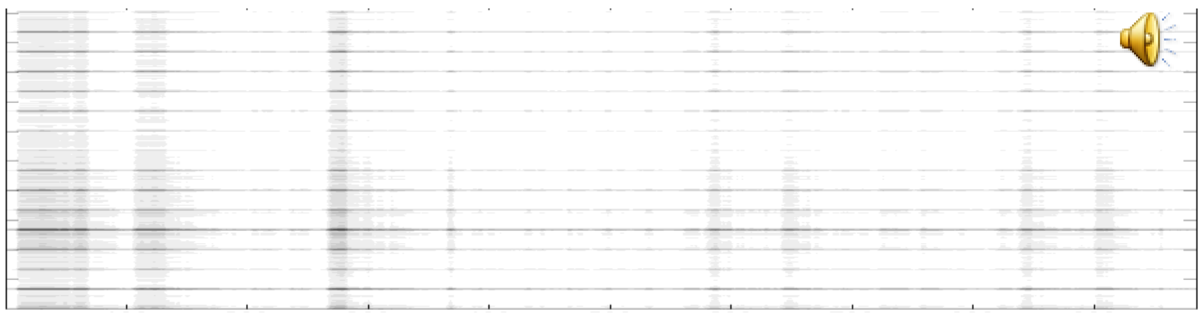


X =PINV(W)*M



W =



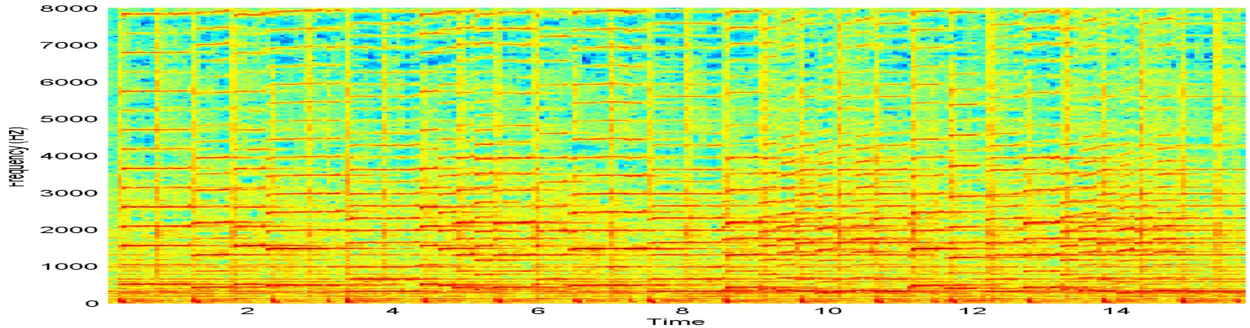


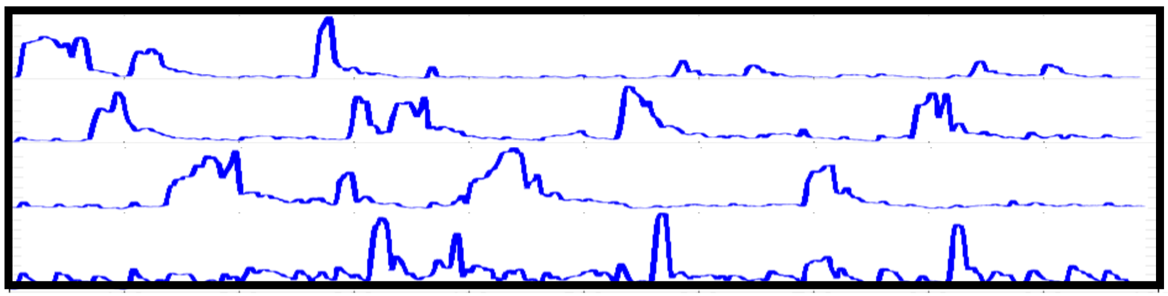
- Recap: P = W $(W^TW)^{-1}$ W^T, Projected Spectrogram = P*M
- **Approximation: M = W*X**
- The amount of W in each vector = X = PINV(W)*M
- W*Pinv(W)*M = Projected Spectrogram
  - W*Pinv(W) = Projection matrix!!

PINV(W) = $(W^TW)^{-1}W^T$

Explanation with multiple notes

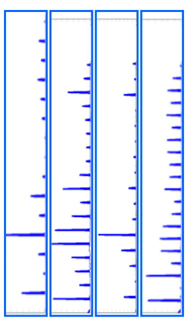
M =

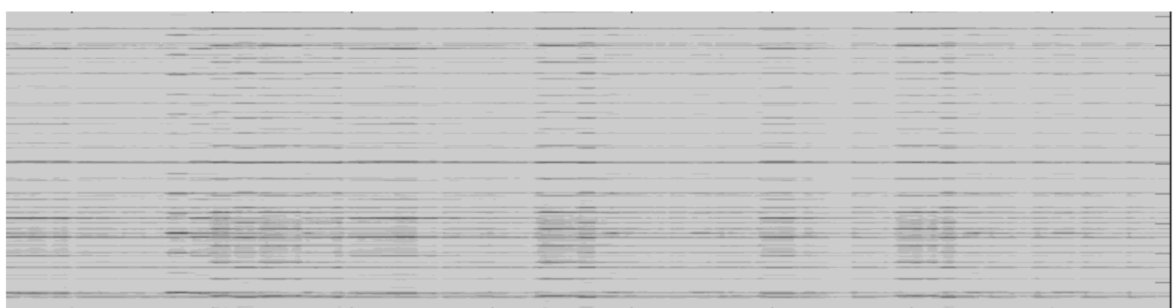




X=PINV(W)M

W =

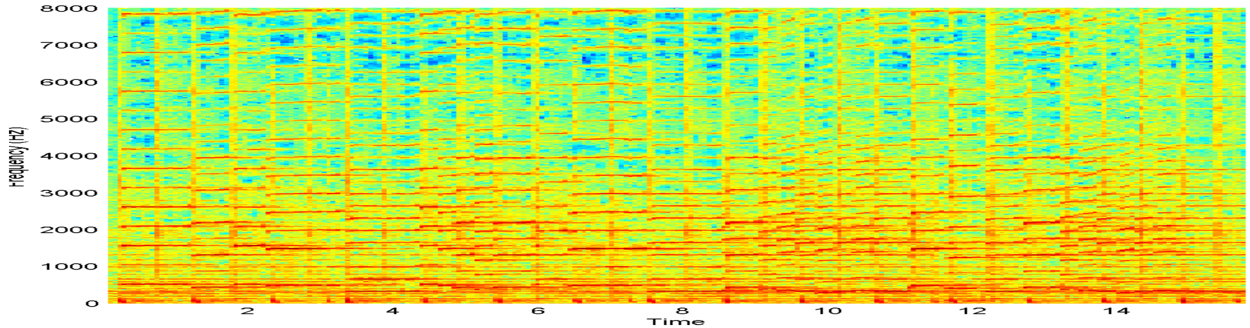




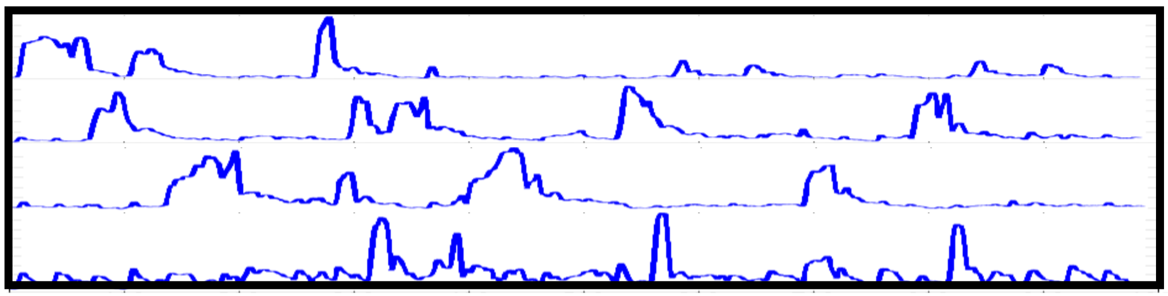
- X = Pinv(W) * M; Projected matrix = W*X = W*Pinv(W)*M

How about the other way?

M =



V =



W = ?

U = ?

- $WV \approx M$ $\quad W = M\,Pinv(V) \quad U = WV$

Pseudo-inverse (PINV)

- Pinv() applies to non-square matrices
- Pinv (Pinv (A))) = A
- A*Pinv(A)= projection matrix!
  - Projection onto the columns of A

- If A = K x N matrix and K > N, A projects N-D vectors into a higher-dimensional K-D space
  - Pinv(A) = NxK matrix
  - Pinv(A)*A = I in this case
- Otherwise A * Pinv(A) = I

Matrix inversion (division)

- The inverse of matrix multiplication
  - Not element-wise division!!
- Provides a way to "undo" a linear transformation
  - Inverse of the unit matrix is itself
  - Inverse of a diagonal is diagonal
  - Inverse of a rotation is a (counter)rotation (its transpose!)
  - Inverse of a rank deficient matrix does not exist!
    - But pseudoinverse exists
- For square matrices: Pay attention to multiplication side!

$$\mathbf{A}\cdot\mathbf{B}=\mathbf{C},\ \ \mathbf{A}=\mathbf{C}\cdot\mathbf{B}^{-1},\ \ \mathbf{B}=\mathbf{A}^{-1}\cdot\mathbf{C}$$

- If matrix is not square use a matrix pseudoinverse:

$$\mathbf{A}\cdot\mathbf{B}\approx\mathbf{C},\ \mathbf{A}=\mathbf{C}\cdot\mathbf{B}^{+},\ \mathbf{B}=\mathbf{A}^{+}\cdot\mathbf{C}$$

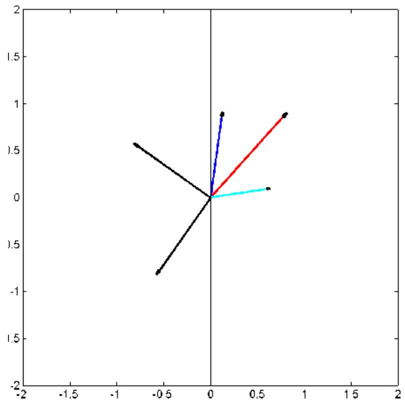
- MATLAB syntax: inv(a), pinv(a)

Eigenanalysis

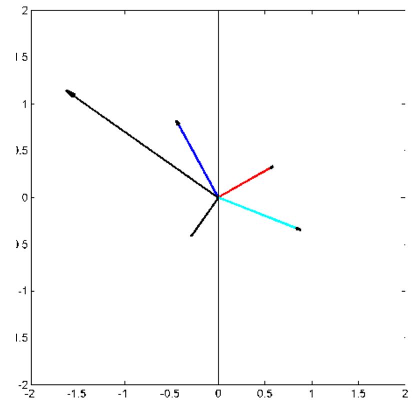
- If something can go through a process mostly unscathed in character it is an *eigen*-something
  – Sound example:
- A vector that can undergo a matrix multiplication and keep pointing the same way is an *eigenvector*
  – Its length can change though
- How much its length changes is expressed by its corresponding *eigenvalue*
  – Each eigenvector of a matrix has its eigenvalue
- Finding these "eigenthings" is called eigenanalysis

EigenVectors and EigenValues

Black vectors are eigen vectors



$$M = \begin{bmatrix} 1.5 & -0.7 \\ -0.7 & 1.0 \end{bmatrix}$$

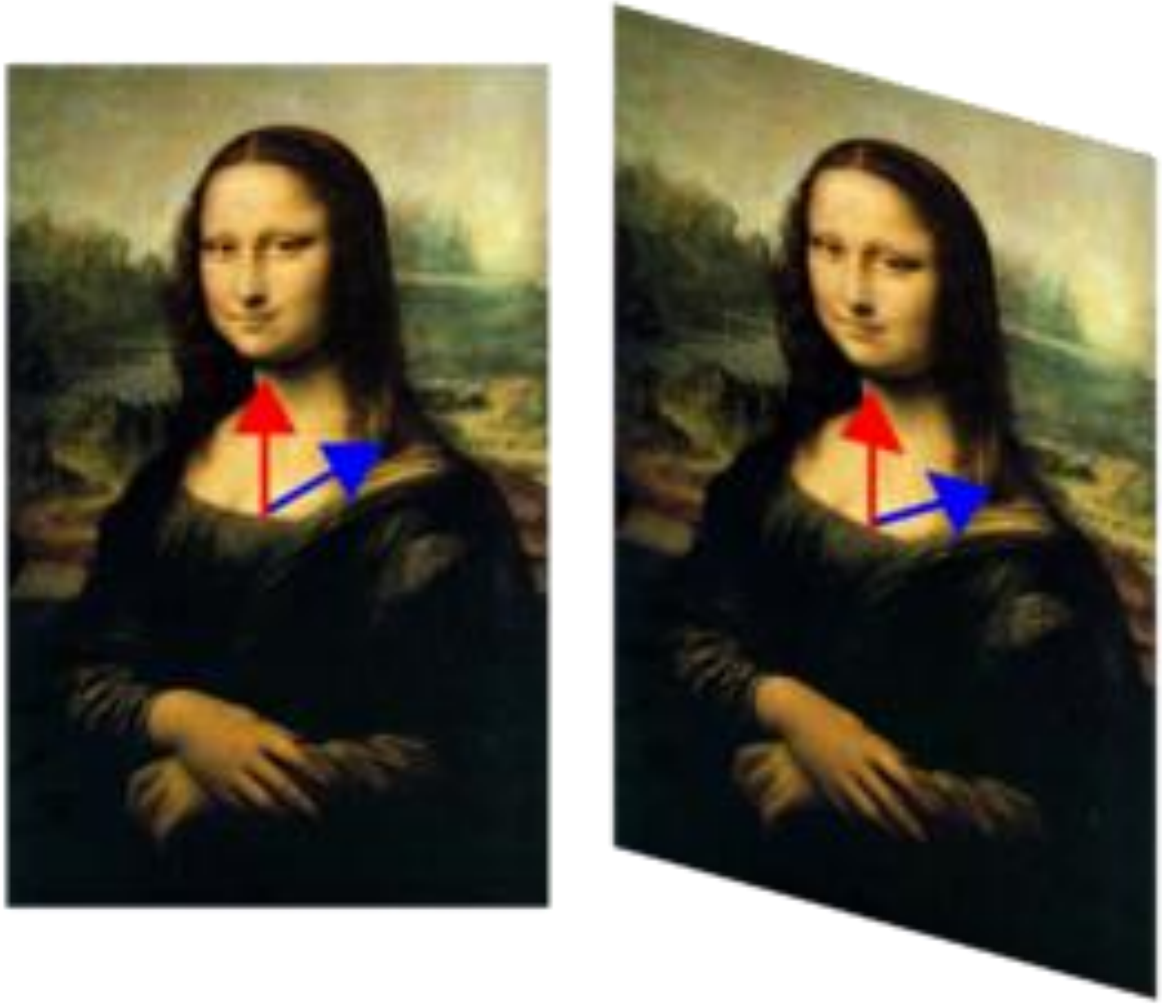


- Vectors that do not change angle upon transformation
  - They may change length

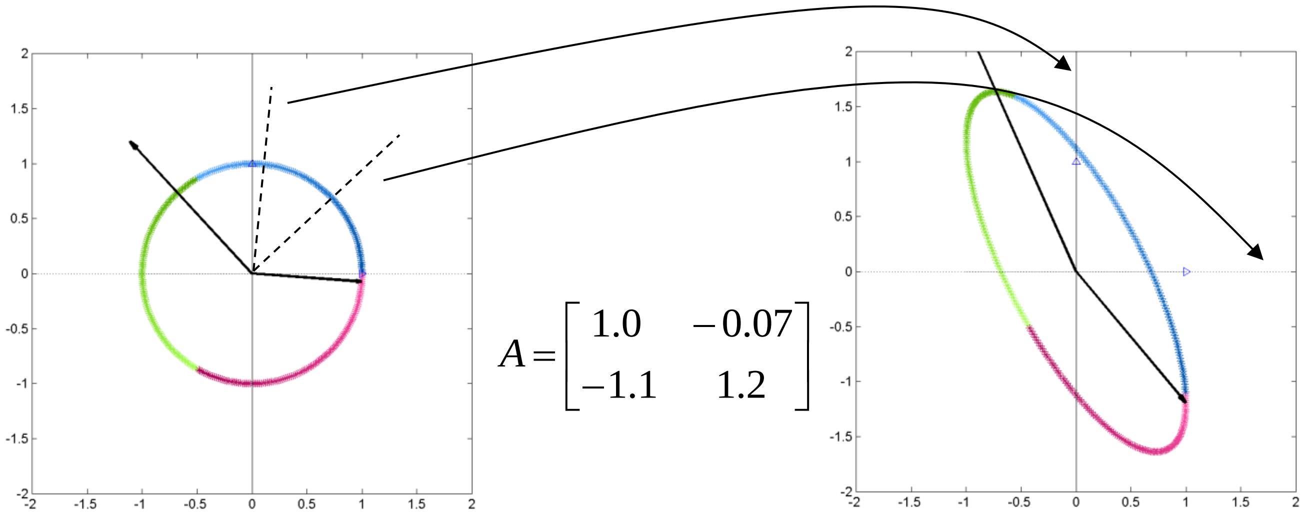
$$MV = \lambda V$$

  - V = eigen vector
  - λ = eigen value

Eigen vector example



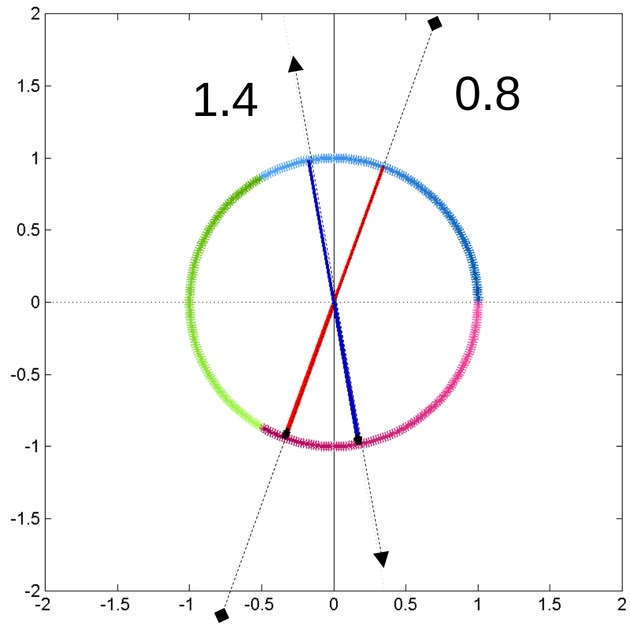
Matrix multiplication revisited



$$A = \begin{bmatrix} 1.0 & -0.07 \\ -1.1 & 1.2 \end{bmatrix}$$

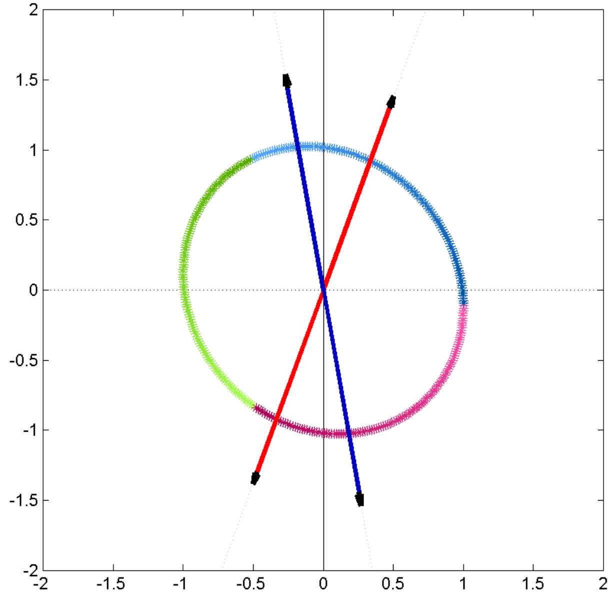
- Matrix transformation “transforms” the space
  - Warps the paper so that the normals to the two vectors now lie along the axes

A stretching operation



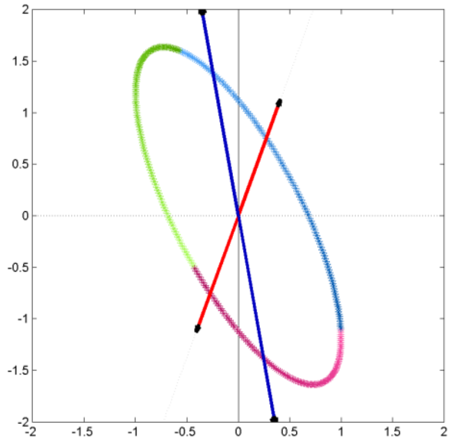
- Draw two lines
- Stretch / shrink the paper along these lines by factors λ_1 and λ_2
  - The factors could be negative – implies flipping the paper
- The result is a transformation of the space

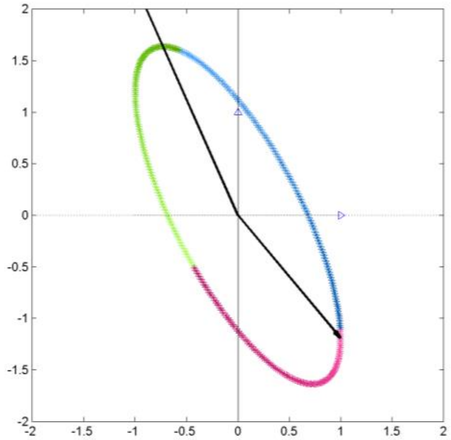
A stretching operation



- Draw two lines
- Stretch / shrink the paper along these lines by factors λ_1 and λ_2
  - The factors could be negative – implies flipping the paper
- The result is a transformation of the space

Physical interpretation of eigen vector





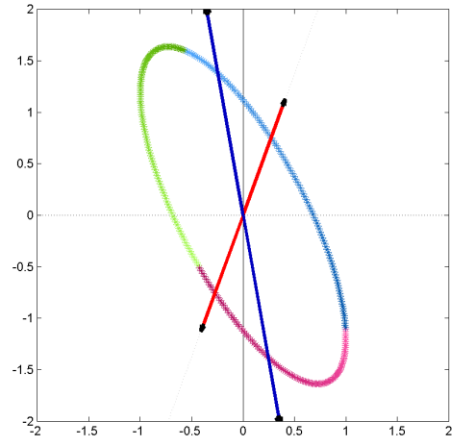
- The result of the stretching is exactly the same as transformation by a matrix
- The axes of stretching/shrinking are the eigenvectors
  - The degree of stretching/shrinking are the corresponding eigenvalues
- The EigenVectors and EigenValues convey all the information about the matrix

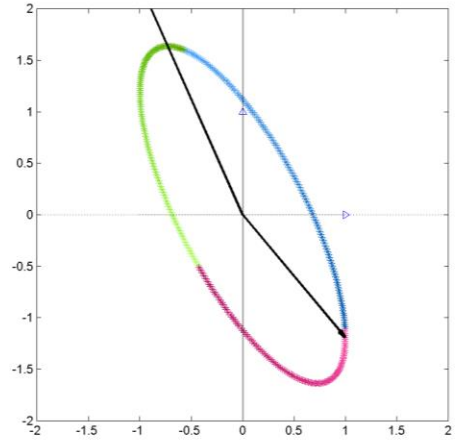
Physical interpretation of eigen vector

$$V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

$$M = V \Lambda V^{-1}$$





- The result of the stretching is exactly the same as transformation by a matrix
- The axes of stretching/shrinking are the eigenvectors
  - The degree of stretching/shrinking are the corresponding eigenvalues
- The EigenVectors and EigenValues convey all the information about the matrix

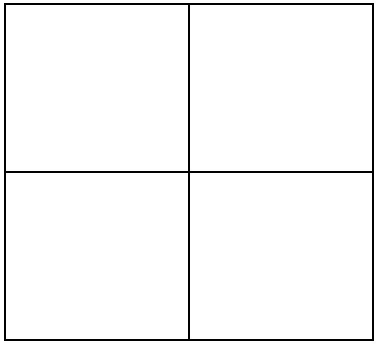
Eigen Analysis

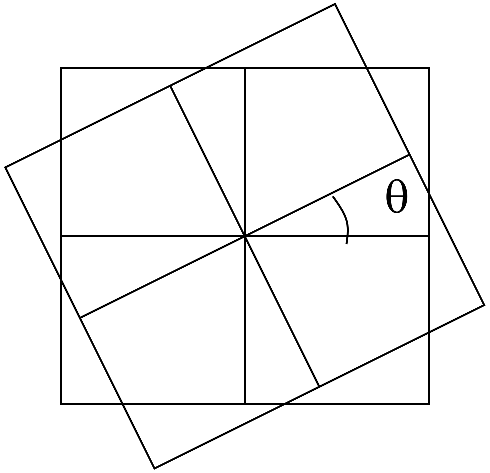
- Not all square matrices have nice eigen values and vectors
  - E.g. consider a rotation matrix

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$X_{new} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

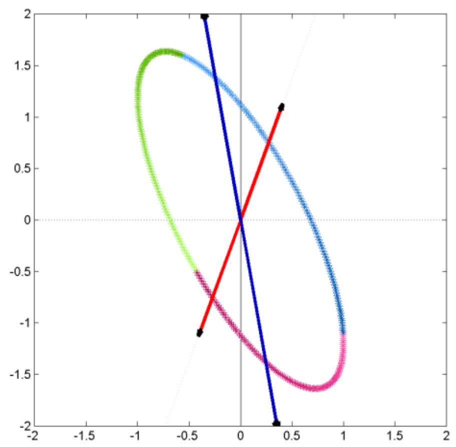




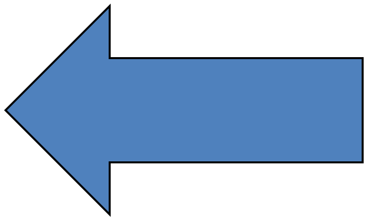
  - This rotates every vector in the plane
    - No vector that remains unchanged

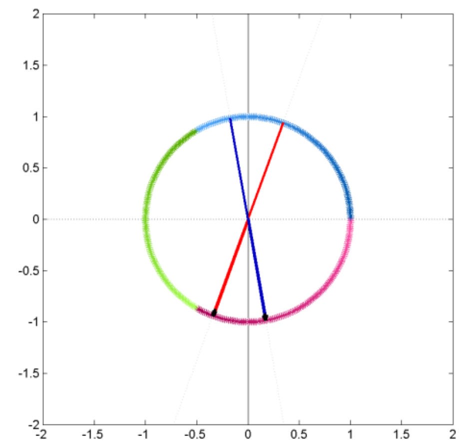
- In these cases the Eigen vectors and values are complex

Singular Value Decomposition



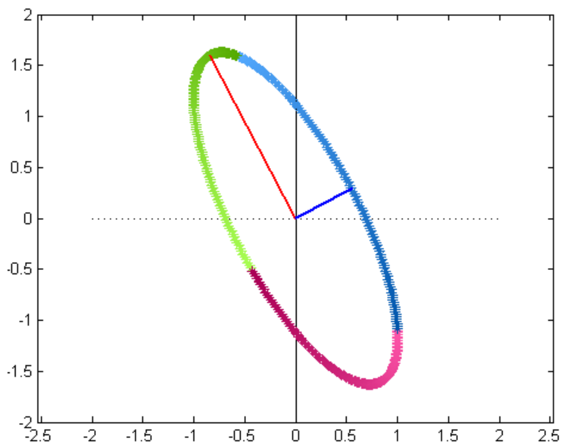
$$A = \begin{bmatrix} 1.0 & -0.07 \\ -1.1 & 1.2 \end{bmatrix}$$



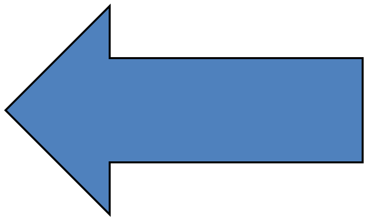


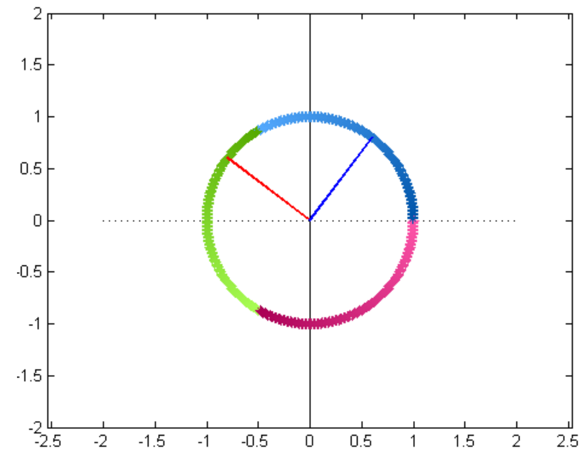
- Matrix transformations convert circles to ellipses
- Eigen vectors are vectors that do not change direction in the process
- There is another key feature of the ellipse to the left that carries information about the transform
  - Can you identify it?

Singular Value Decomposition



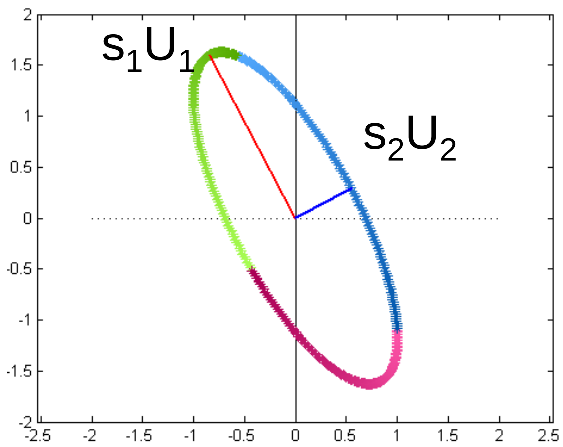
$$A = \begin{bmatrix} 1.0 & -0.07 \\ -1.1 & 1.2 \end{bmatrix}$$





- The major and minor axes of the transformed ellipse define the ellipse
  - They are at right angles
- These are transformations of right-angled vectors on the original circle!

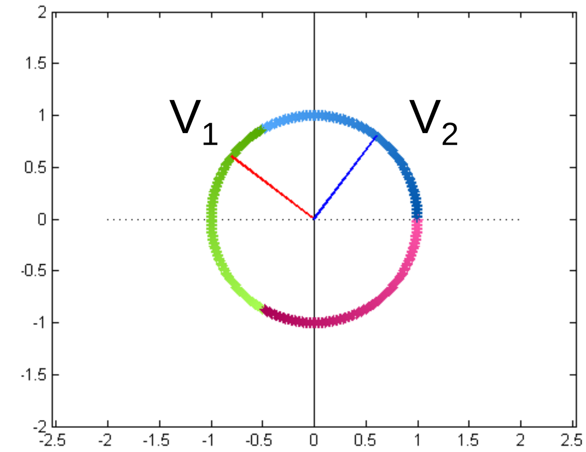
Singular Value Decomposition



$$A = \begin{bmatrix} 1.0 & -0.07 \\ -1.1 & 1.2 \end{bmatrix}$$

$$A = U\ S\ V^T$$

matlab:
[U,S,V] = svd(A)



- U and V are orthonormal matrices
  - Columns are orthonormal vectors
- S is a diagonal matrix

- The *right singular vectors* in V are transformed to the *left singular vectors* in U
  - And scaled by the *singular values* that are the diagonal entries of S

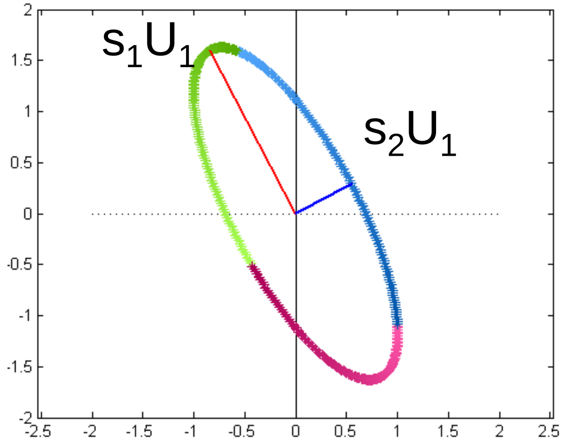
Singular Value Decomposition

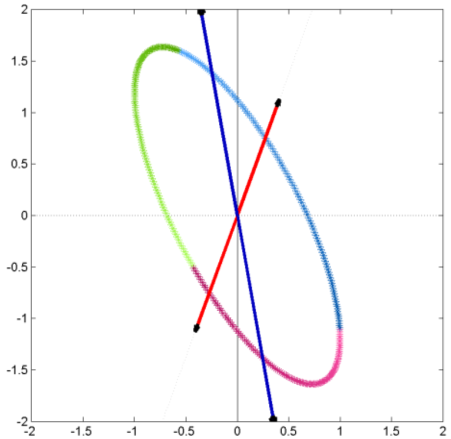
- The left and right singular vectors are not the same
  - If A is not a square matrix, the left and right singular vectors will be of different dimensions

- The singular values are always real

- The largest singular value is the largest amount by which a vector is scaled by A
  - Max (|Ax| / |x|) = s_{max}
- The smallest singular value is the smallest amount by which a vector is scaled by A
  - Min (|Ax| / |x|) = s_{min}
  - This can be 0 (for low-rank or non-square matrices)

The Singular Values

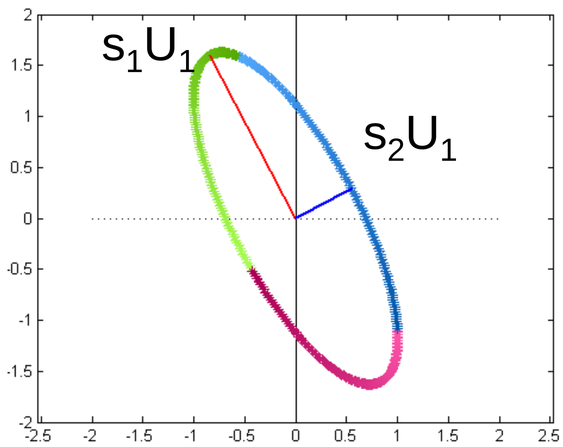


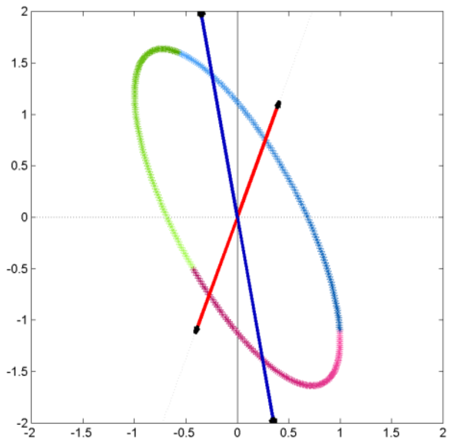


- Square matrices: product of singular values = determinant of the matrix
  - This is also the product of the *eigen* values
  - I.e. there are two different sets of axes whose products give you the area of an ellipse

- For any "broad" rectangular matrix A, the largest singular value of any square submatrix B cannot be larger than the largest singular value of A
  - An analogous rule applies to the smallest singular value
  - This property is utilized in various problems, such as compressive sensing

SVD vs. Eigen Analysis

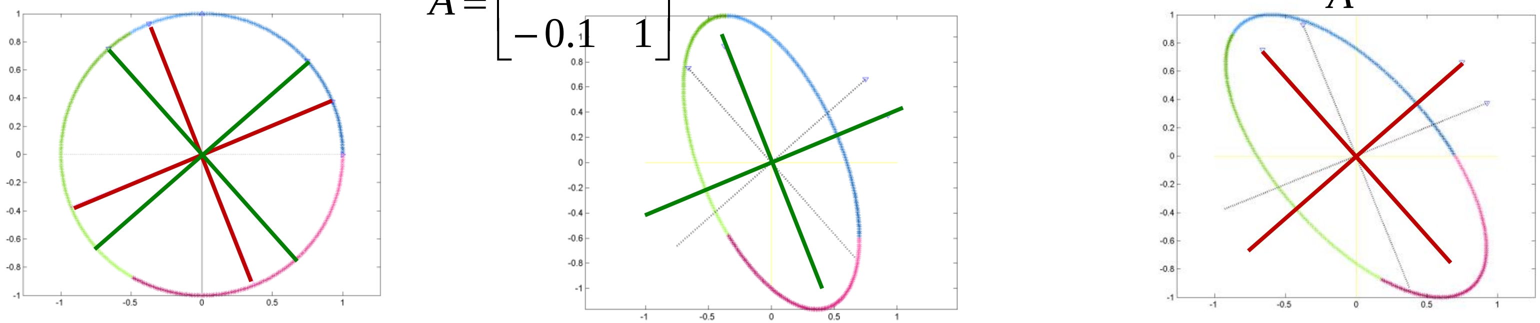




- Eigen analysis of a matrix **A**:
  - Find two vectors such that their absolute directions are not changed by the transform
- SVD of a matrix **A**:
  - Find two vectors such that the *angle* between them is not changed by the transform
- For one class of matrices, these two operations are the same

A matrix vs. its transpose

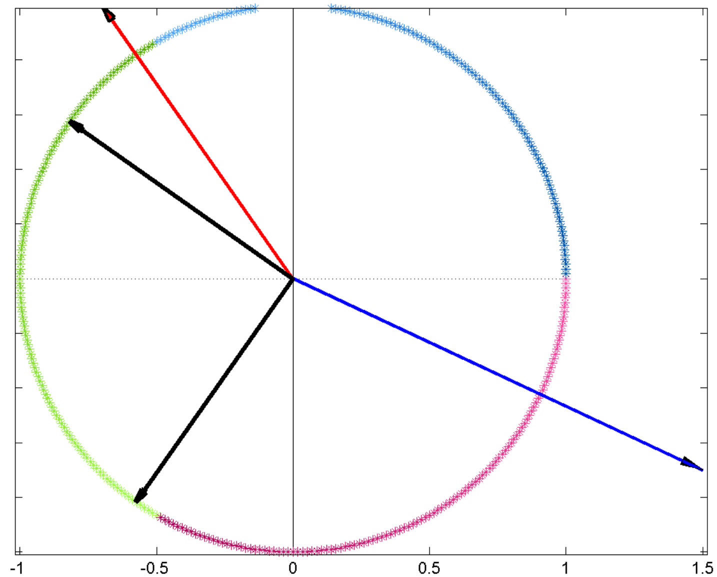
$$A = \begin{bmatrix} .7 & 0 \\ -0.1 & 1 \end{bmatrix}$$



- Multiplication by matrix A:
  - Transforms right singular vectors in V to left singular vectors U
- Multiplication by its transpose A^T:
  - Transforms *left* singular vectors U to right singular vector V
- A A^T : Converts V to U, then brings it back to V
  - Result: Only scaling

Symmetric Matrices

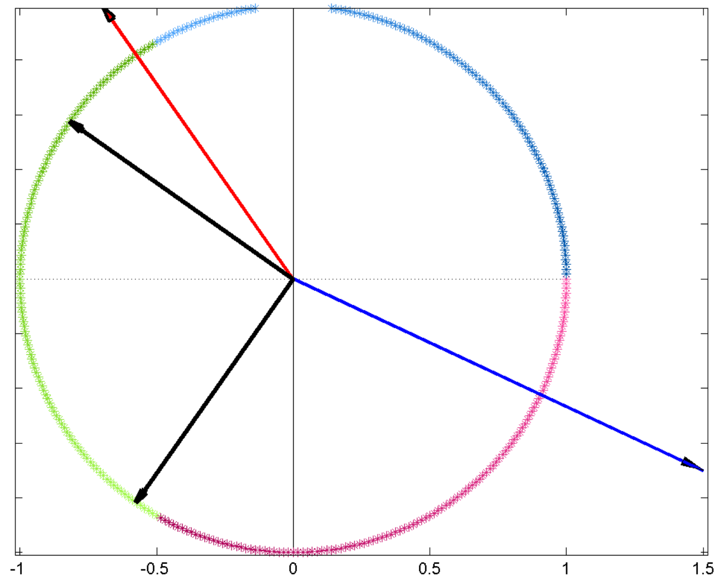
$$\begin{bmatrix} 1.5 & -0.7 \\ -0.7 & 1 \end{bmatrix}$$



- Matrices that do not change on transposition
  - Row and column vectors are identical
- The left and right singular vectors are identical
  - U = V
  - A = U S U^T
- They are identical to the *Eigen vectors* of the matrix
- Symmetric matrices do not rotate the space
  - Only scaling and, if Eigen values are negative, reflection

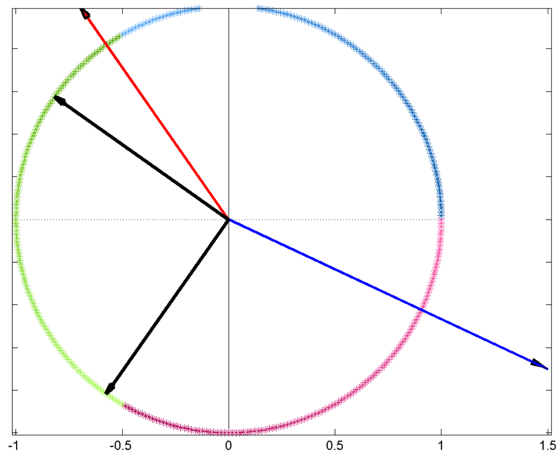
Symmetric Matrices

$$\begin{bmatrix} 1.5 & -0.7 \\ -0.7 & 1 \end{bmatrix}$$

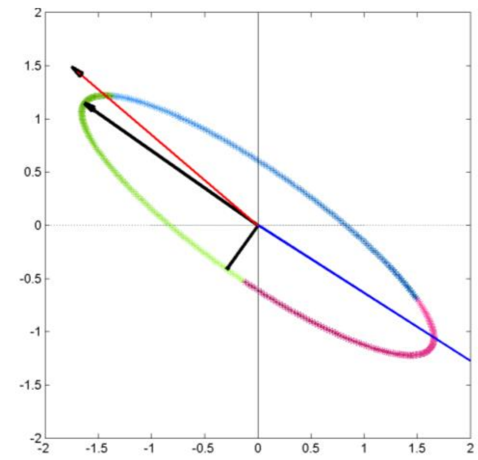


- Matrices that do not change on transposition
  - Row and column vectors are identical
- Symmetric matrix: Eigen vectors and Eigen values are always real
- Eigen vectors are always orthogonal
  - At 90 degrees to one another

Symmetric Matrices



$$\begin{bmatrix} 1.5 & -0.7 \\ -0.7 & 1 \end{bmatrix}$$



- Eigen vectors point in the direction of the major and minor axes of the ellipsoid resulting from the transformation of a spheroid
  - The eigen values are the lengths of the axes

Symmetric matrices

- Eigen vectors V_i are orthonormal
  - $V_i^T V_i = 1$
  - $V_i^T V_j = 0, i != j$
- Listing all eigen vectors in matrix form V
  - $V^T = V^{-1}$
  - $V^T V = I$
  - $V V^T = I$
- $M V_i = \lambda V_i$
- In matrix form : $M V = V \Lambda$
  - Λ is a diagonal matrix with all eigen values
- $M = V \Lambda V^T$

Square root of a symmetric matrix

$$C = V\Lambda V^T$$

$$Sqrt(C) = V.Sqrt(\Lambda).V^T$$

$$Sqrt(C).Sqrt(C) = V.Sqrt(\Lambda).V^T V.Sqrt(\Lambda).V^T$$

$$= V.Sqrt(\Lambda).Sqrt(\Lambda)V^T = V\Lambda V^T = C$$

- The *square root* of a symmetric matrix is easily derived from the Eigen vectors and Eigen values
  - The Eigen values of the *square root* of the matrix are the square roots of the Eigen values of the matrix
  - For correlation matrices, these are also the "singular values" of the data set

Definiteness..

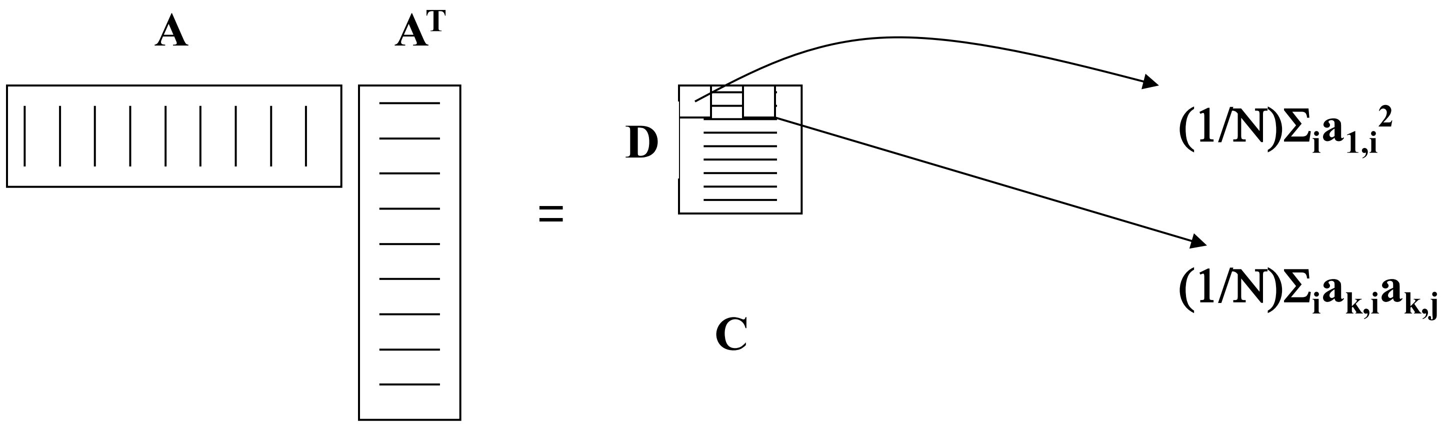
- SVD: Singular values are always positive!
- Eigen Analysis: Eigen values can be real or imaginary
  - Real, positive Eigen values represent stretching of the space along the Eigen vector
  - Real, *negative* Eigen values represent stretching and *reflection* (across origin) of Eigen vector
  - Complex Eigen values occur in conjugate pairs

- A square (symmetric) matrix is **positive definite** if all Eigen values are real and positive, and are greater than 0
  - Transformation can be explained as **stretching** and **rotation**
  - If any Eigen value is **zero**, the matrix is positive *semi-definite*

Positive Definiteness..

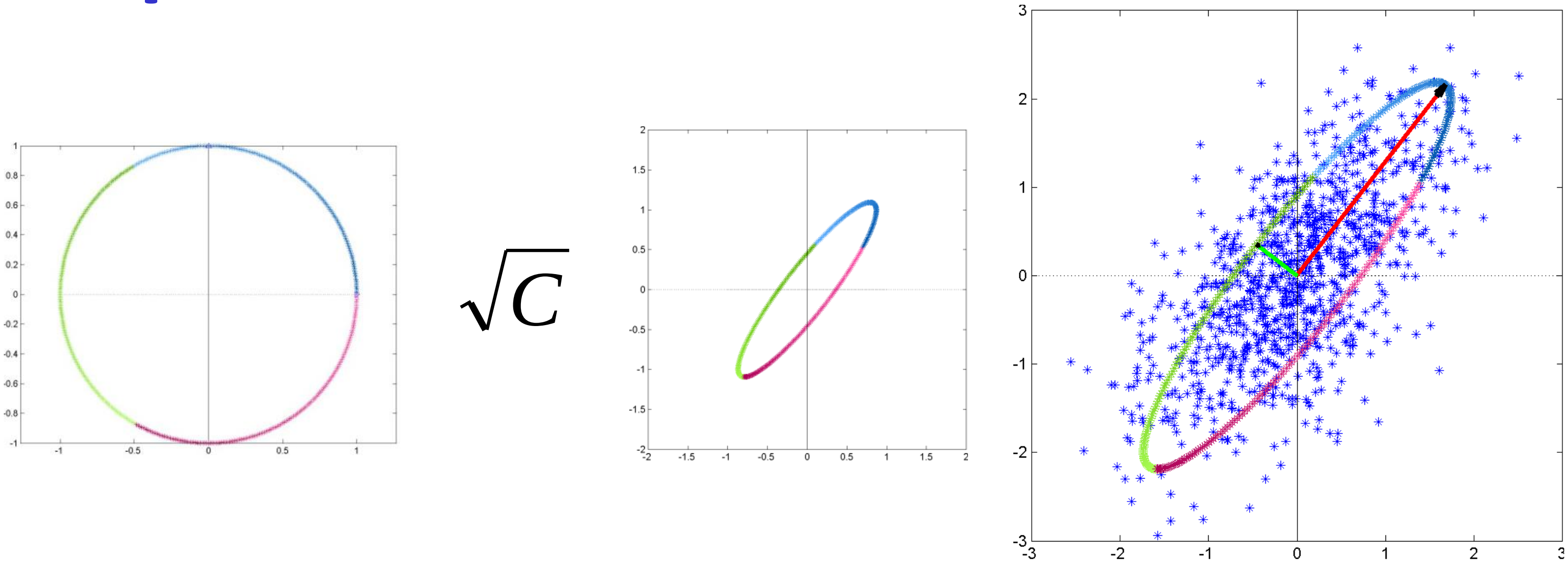
- Property of a positive definite matrix: Defines inner product norms
  - x^TAx is always positive for any vector x if A is positive definite
- Positive definiteness is a test for validity of *Gram* matrices
  - Such as correlation and covariance matrices
  - We will encounter other gram matrices later

The Correlation and Covariance Matrices



- Consider a set of column vectors ordered as a DxN matrix A
- The correlation matrix is
  - $C = (1/N)\, AA^T$
    - If the average (mean) of the vectors in A is subtracted out of all vectors, C is the ***covariance*** matrix
    - **covariance = correlation + mean * mean^T**
- Diagonal elements represent average of the squared value of each dimension
  - Off diagonal elements represent how two components are related
    - How much knowing one lets us guess the value of the other

Square root of the *Covariance* Matrix

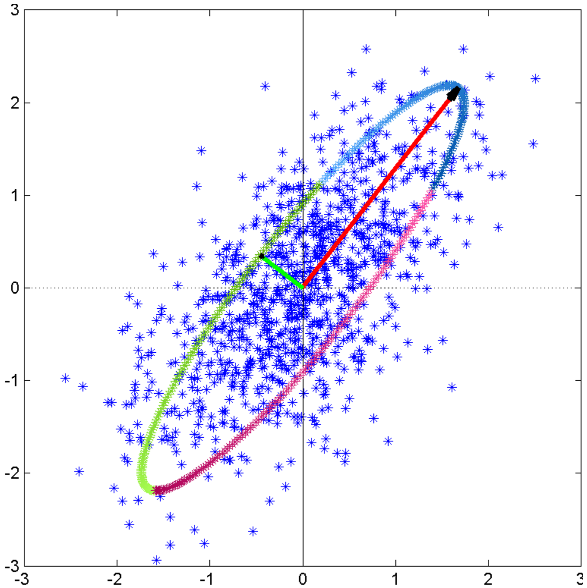


- The square root of the covariance matrix represents the elliptical scatter of the data
- The Eigenvectors of the matrix represent the major and minor axes
  - “Modes” in direction of scatter

The *Correlation* Matrix

Any vector $V = a_{V,1}$ * eigenvec1 + $a_{V,2}$ *eigenvec2 + ..

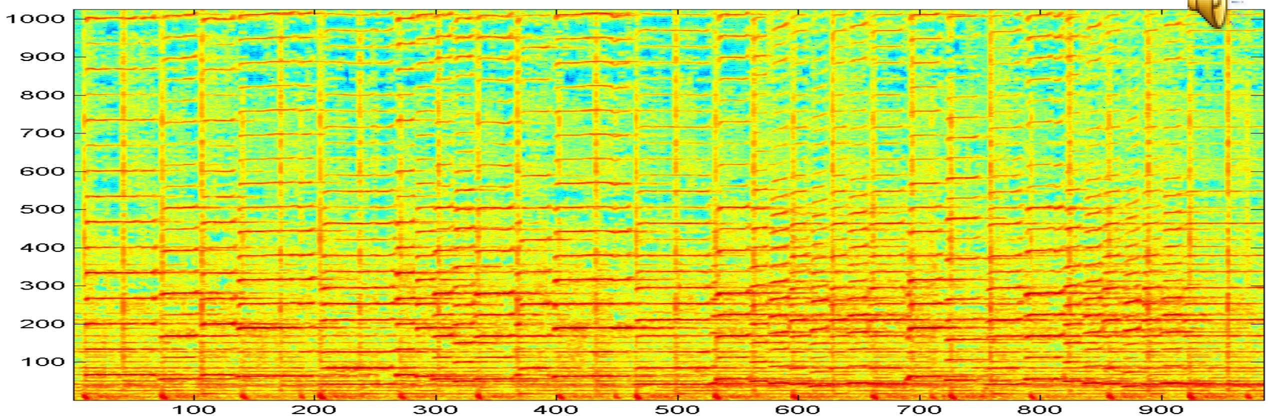
$\Sigma_V a_{V,i}$ = eigenvalue(i)



- Projections along the N Eigen vectors with the largest Eigen values represent the N greatest "energy-carrying" components of the matrix

- Conversely, N "bases" that result in the least square error are the N best Eigen vectors

An audio example



- The spectrogram has 974 vectors of dimension 1025
- The covariance matrix is size 1025 x 1025
- There are 1025 eigenvectors

Eigen Reduction

$$M = spectrogram$$ 1025x1000

$$C = M.M^T$$ 1025x1025

V = 1025x1025

$$[V, L] = eig(C)$$

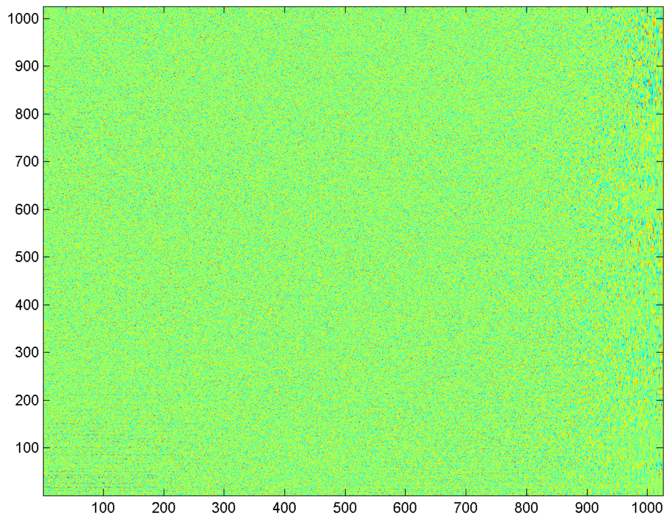
$$V_{reduced} = [V_1 \quad . \quad . \quad V_{25}]$$ 1025x25

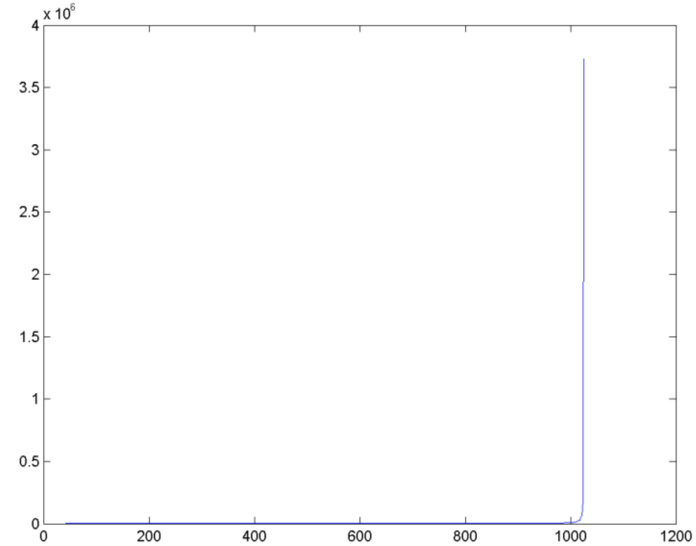
$$M_{\text{lowdim}} = Pinv(V_{reduced})M$$ 25x1000

$$M_{reconstructed} = V_{reduced} M_{\text{lowdim}}$$ 1025x1000

- Compute the Correlation
- Compute Eigen vectors and values
- Create matrix from the 25 Eigen vectors corresponding to 25 highest Eigen values
- Compute the weights of the 25 eigenvectors
- To reconstruct the spectrogram – compute the projection on the 25 Eigen vectors

Eigenvalues and Eigenvectors

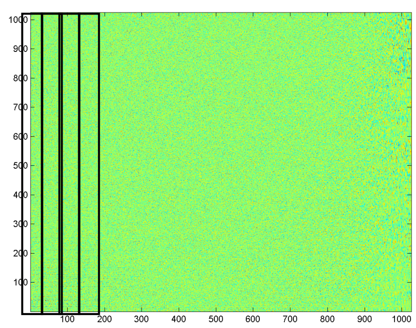


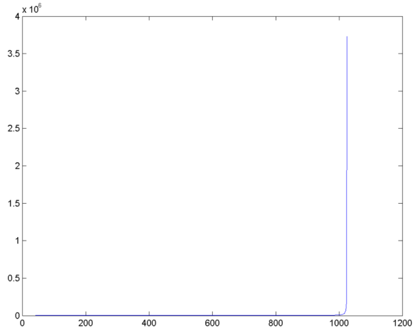


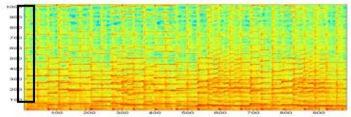
- Left panel: Matrix with 1025 eigen vectors
- Right panel: Corresponding eigen values
  - Most Eigen values are close to zero
    - The corresponding eigenvectors are "unimportant"

$$M = spectrogram$$
$$C = M.M^T$$
$$[V, L] = eig(C)$$

Eigenvalues and Eigenvectors





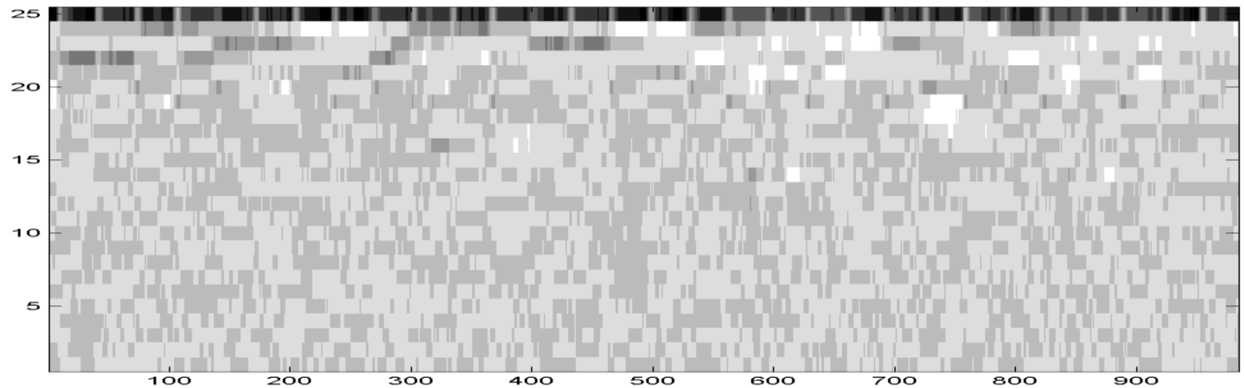


Vec = a1 *eigenvec1 + a2 * eigenvec2 + a3 * eigenvec3 …

- The vectors in the spectrogram are linear combinations of all 1025 Eigen vectors
- The Eigen vectors with low Eigen values contribute very little
  - The average value of a_i is proportional to the square root of the Eigenvalue
  - Ignoring these will not affect the composition of the spectrogram

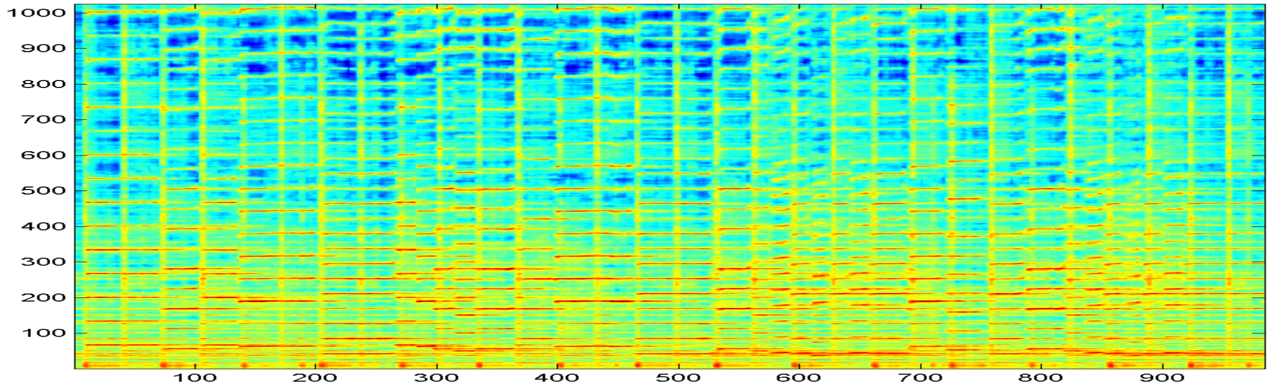
An audio example

$$V_{reduced} = [V_1 \quad . \quad . \quad V_{25}]$$
$$M_{lowdim} = Pinv(V_{reduced})M$$



- The same spectrogram projected down to the 25 eigen vectors with the highest eigen values
  - Only the 25-dimensional weights are shown
    - The weights with which the 25 eigen vectors must be added to compose a least squares approximation to the spectrogram

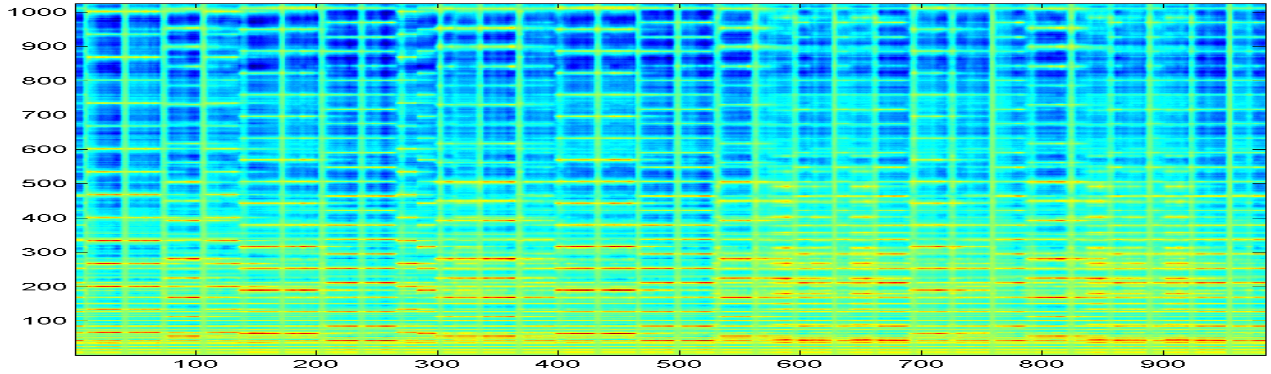
An audio example



$$M_{reconstructed} = V_{reduced} M_{lowdim}$$

- The same spectrogram constructed from only the 25 Eigen vectors with the highest Eigen values
  - Looks similar
    - With 100 Eigenvectors, it would be indistinguishable from the original
  - Sounds pretty close
  - But now sufficient to store 25 numbers per vector (instead of 1024)

With only 5 eigenvectors



- The same spectrogram constructed from only the 5 Eigen vectors with the highest Eigen values
  - Highly recognizable

Correlation vs. Covariance Matrix

- Correlation:
  - The N Eigen vectors with the largest Eigen values represent the N greatest "energy-carrying" components of the matrix
  - Conversely, N "bases" that result in the least square error are the N best Eigen vectors
    - Projections onto these Eigen vectors retain the most energy

- Covariance:
  - the N Eigen vectors with the largest Eigen values represent the N greatest *"variance-carrying"* components of the matrix
  - Conversely, N "bases" that retain the maximum possible variance are the N best Eigen vectors

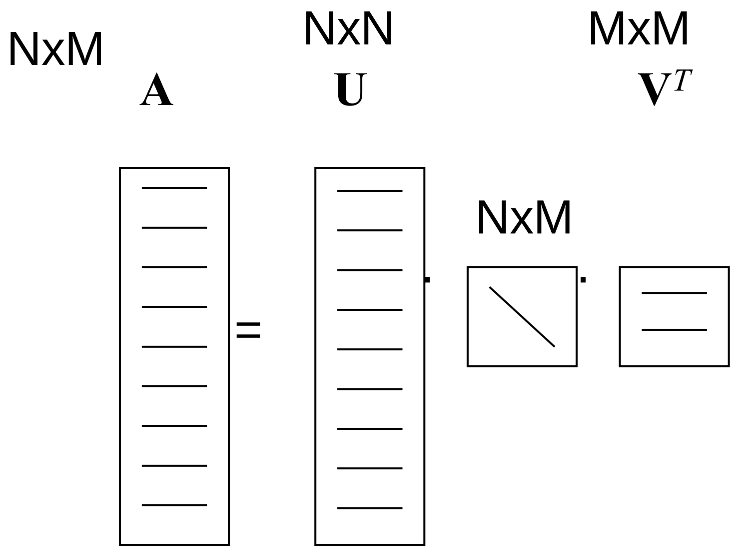
Eigenvectors, Eigenvalues and Covariances/Correlations

- The eigenvectors and eigenvalues (singular values) derived from the correlation matrix are important
- Do we need to actually compute the correlation matrix?
  - No
- Direct computation using Singular Value Decomposition

SVD vs. Eigen decomposition

- Singular value decomposition is analogous to the Eigen decomposition of the correlation matrix of the data
  - SVD: $D = U\,S\,V^T$
  - $DD^T = U\,S\,V^T\,V\,S\,U^T = U\,S^2\,U^T$
- The “left” singular vectors are the Eigen vectors of the correlation matrix
  - Show the directions of greatest importance
- The corresponding singular values are the square roots of the Eigen values of the correlation matrix
  - Show the importance of the Eigen vector

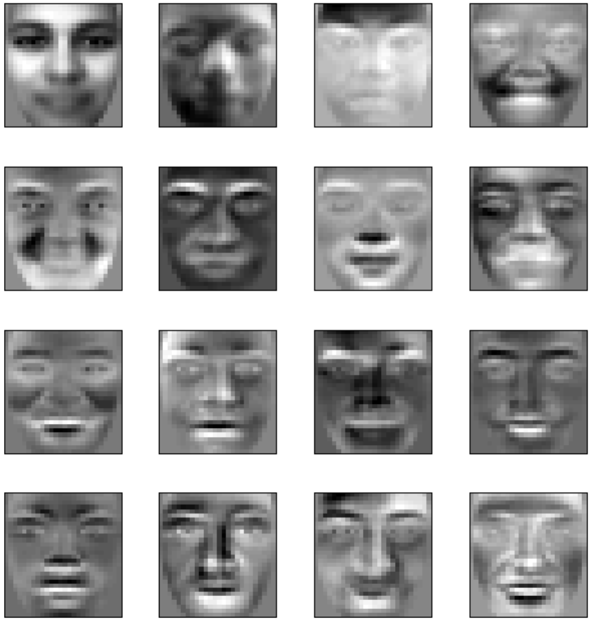
Thin SVD, compact SVD, reduced SVD



- **SVD can be computed much more efficiently than Eigen decomposition**
- Thin SVD: Only compute the first N columns of U
  - All that is required if N < M
- Compact SVD: Only the left and right singular vectors corresponding to non-zero singular values are computed

Why bother with Eigens/SVD

- Can provide a unique insight into data
  - Strong statistical grounding
  - Can display complex interactions between the data
  - Can uncover irrelevant parts of the data we can throw out
- Can provide *basis functions*
  - A set of elements to compactly describe our data
  - Indispensable for performing compression and classification
- Used over and over and still perform amazingly well



Eigenfaces
Using a linear transform of the above "eigenvectors" we can compose various faces

Trace

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

$$Tr(A) = a_{11} + a_{22} + a_{33} + a_{44}$$

$$Tr(A) = \sum_i a_{i,i}$$

- The trace of a matrix is the sum of the diagonal entries
- It is equal to the sum of the Eigen values!

$$Tr(A) = \sum_i a_{i,i} = \sum_i \lambda_i$$

Trace

- Often appears in Error formulae

$$D = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & a_{32} & a_{33} & a_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix} \qquad C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}$$

$$E = D - C \qquad error = \sum_{i,j} E_{i,j}^2 \qquad error = Tr(EE^T)$$

- Useful to know some properties..

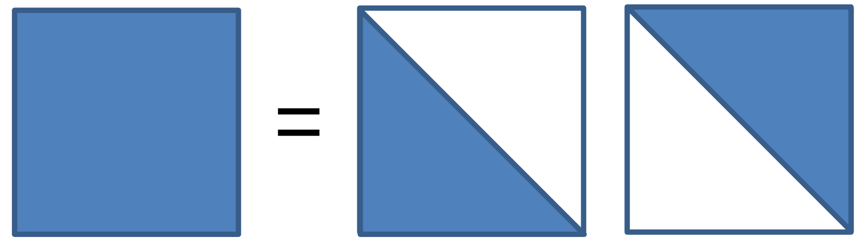
Properties of a Trace

- Linearity: $Tr(A+B) = Tr(A) + Tr(B)$
  $Tr(c.A) = c.Tr(A)$

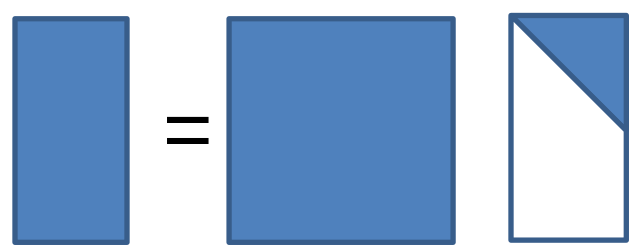
- Cycling invariance:
  - $Tr(ABCD) = Tr(DABC) = Tr(CDAB) = Tr(BCDA)$
  - $Tr(AB) = Tr(BA)$

- Frobenius norm $F(A) = \Sigma_{i,j} a_{ij}^2 = Tr(AA^T)$

Decompositions of matrices



- Square A: LU decomposition
  - Decompose A = L U
  - L is a *lower triangular* matrix
    - All elements above diagonal are 0
  - R is an *upper triangular* matrix
    - All elements below diagonal are zero
  - Cholesky decomposition: A is symmetric, $L = U^T$



- QR decompositions: A = QR
  - Q is orthgonal: $QQ^T = I$
  - R is upper triangular
- Generally used as tools to compute Eigen decomposition or least square solutions

Making vectors and matrices in MATLAB

- Make a row vector:
  `a = [1 2 3]`
- Make a column vector:
  `a = [1;2;3]`
- Make a matrix:
  `A = [1 2 3;4 5 6]`
- Combine vectors
  `A = [b c]` or `A = [b;c]`
- Make a random vector/matrix:
  `r = rand(m,n)`
- Make an identity matrix:
  `I = eye(n)`
- Make a sequence of numbers
  `c = 1:10` or `c = 1:0.5:10` or `c = 100:-2:50`
- Make a ramp
  `c = linspace( 0, 1, 100)`

Indexing

- To get the *i*-th element of a vector
  ```
  a(i)
  ```
- To get the *i*-th *j*-th element of a matrix
  ```
  A(i,j)
  ```
- To get from the *i*-th to the *j*-th element
  ```
  a(i:j)
  ```
- To get a *sub-matrix*
  ```
  A(i:j,k:l)
  ```
- To get segments
  ```
  a([i:j k:l m])
  ```

Arithmetic operations

- Addition/subtraction

  `C = A + B` or `C = A - B`
- Vector/Matrix multiplication

  `C = A * B`
  - Operant sizes must match!
- Element-wise operations
  - Multiplication/division

    `C = A .* B` or `C = A ./ B`
  - Exponentiation

    `C = A.^B`
  - Elementary functions

    `C = sin(A)` or `C = sqrt(A)`, …

Linear algebra operations

- Transposition
  `C = A'`
  - If `A` is complex also conjugates use `C = A.'` to avoid that
- Vector norm
  `norm(x)` (also works on matrices)
- Matrix inversion
  `C = inv(A)` if `A` is square
  `C = pinv(A)` if `A` is not square
  - `A` might not be invertible, you'll get a warning if so
- Eigenanalysis
  `[u,d] = eig(A)`
  - `u` is a matrix containing the eigenvectors
  - `d` is a diagonal matrix containing the eigenvalues
- Singular Value Decomposition
  `[u,s,v] = svd(A)` or `[u,s,v] = svd(A,0)`
  - "thin" versus regular SVD
  - `s` is diagonal and contains the singular values

Plotting functions

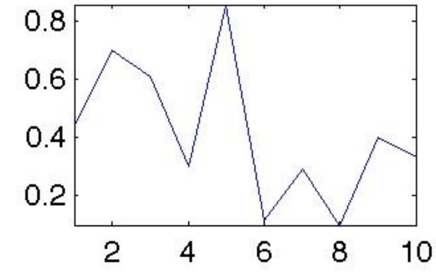
- 1-d plots

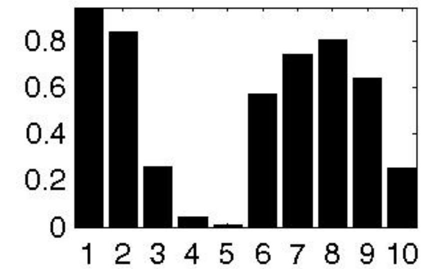
  `plot(x)`
  - if `x` is a vector will plot all its elements
  - If `x` is a matrix will plot all its column vectors

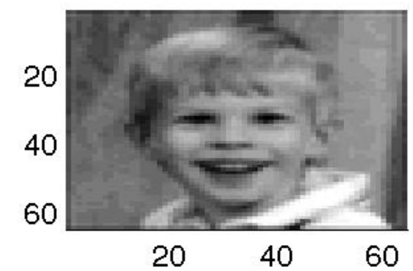
  `bar(x)`
  - Ditto but makes a bar plot
- 2-d plots

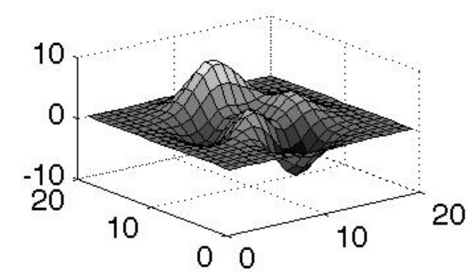
  `imagesc(x)`
  - plots a matrix as an image

  `surf(x)`
  - makes a surface plot









Getting help with functions

- The `help` function
  - Type `help` followed by a function name
- Things to try

```
help help
help +
help eig
help svd
help plot
help bar
help imagesc
help surf
help ops
help matfun
```

- Also check out the tutorials and the mathworks site