Fundamentals of Linear Algebra

Class 2. 27 August 2009

Instructor: Bhiksha Raj

Overview

- Vectors and matrices
- Basic vector/matrix operations
- Vector products
- Matrix products
- Various matrix types
- Matrix inversion
- Matrix interpretation
- Eigenanalysis
- Singular value decomposition

Book

- Fundamentals of Linear Algebra, Gilbert Strang

- Important to be very comfortable with linear algebra
  - Appears repeatedly in the form of Eigen analysis, SVD, Factor analysis
  - Appears through various properties of matrices that are used in machine learning, particularly when applied to images and sound

- Today's lecture: Definitions
  - Very small subset of all that's used
  - Important subset, intended to help you recollect

Incentive to use linear algebra

- Pretty notation!

$$\mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{y} \quad \longleftrightarrow \quad \sum_j y_j \sum_i x_i a_{ij}$$

- Easier intuition
  - *Really convenient geometric interpretations*
  - Operations easy to describe verbally
- Easy code translation!

```
for i=1:n
  for j=1:m
    c(i)=c(i)+y(j)*x(i)*a(i,j)
  end
end
```

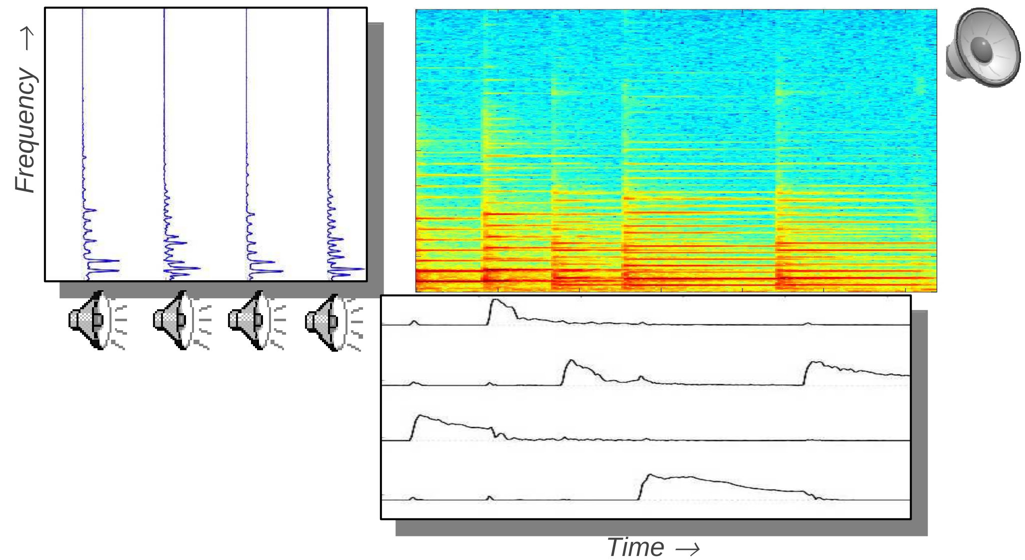
$\longleftrightarrow$

```
C=x*A*y
```

And other things you can do

Rotation + Projection + Scaling

From Bach's Fugue in Gm



Decomposition (NMF)

- Manipulate Images
- Manipulate Sounds

Scalars, vectors, matrices, …

- A *scalar* a is a number
  - a = 2, a = 3.14, a = -1000, etc.
- A *vector* **a** is a linear arrangement of a collection of scalars

$$\mathbf{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} 3.14 \\ -32 \end{bmatrix}$$

- A *matrix* **A** is a rectangular arrangement of a collection of vectors

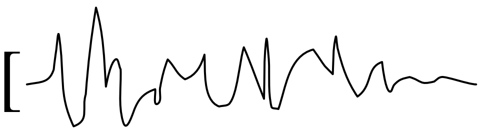
$$\mathbf{A} = \begin{bmatrix} 3.12 & -10 \\ 10.0 & 2 \end{bmatrix}$$

- MATLAB syntax: `a=[1 2 3], A=[1 2;3 4]`

Vector/Matrix types and shapes

- Vectors are either column or row vectors

$$\mathbf{c} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} a & b & c \end{bmatrix}, \quad \mathbf{s} = [$$

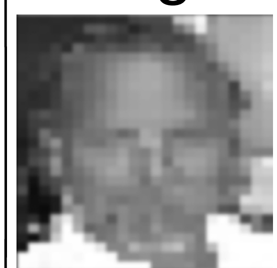


$$]$$

  - A sound can be a vector, a series of daily temperatures can be a vector, etc …

- Matrices can be square or rectangular

$$\mathbf{S} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}, \quad \mathbf{M} = [$$



$$]$$

  - Images can be a matrix, collections of sounds can be a matrix, etc …

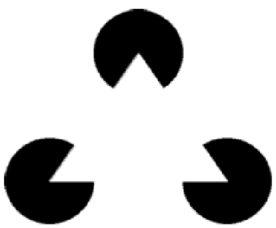
Dimensions of a matrix

- The matrix size is specified by the number of rows and columns

$$\mathbf{c} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \ \mathbf{r} = \begin{bmatrix} a & b & c \end{bmatrix}$$

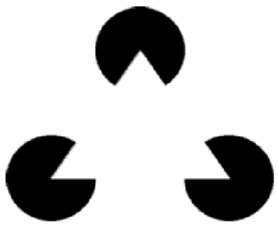
  - c = 3x1 matrix: 3 rows and 1 column
  - r = 1x3 matrix: 1 row and 3 columns

$$\mathbf{S} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \ \mathbf{R} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$



  - S = 2 x 2 matrix
  - R = 2 x 3 matrix
  - Pacman = 321 x 399 matrix

Representing an image as a matrix



```
>> X(1:32:end,1:40:end)

ans =

     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     0     1     1     1
     1     1     1     1     0     0     0     1     1     1
     1     1     1     1     0     1     0     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     0     1     1     1     1     1     0     1
     1     0     0     1     1     1     1     1     0     0
     1     0     0     0     1     1     1     0     0     0
     1     0     0     0     1     1     1     0     0     0
     1     1     1     1     1     1     1     1     1     1
```

$$
\begin{array}{c}
Y \\ X \\ v
\end{array}
\begin{bmatrix}
1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & 10 \\
1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & 10 \\
1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1
\end{bmatrix}
$$

$$
\begin{bmatrix} 1 & 1 & . & 1 & 1 & . & 0 & 0 & 0 & . & . & 1 \end{bmatrix}
$$

Values only; X and Y are implicit

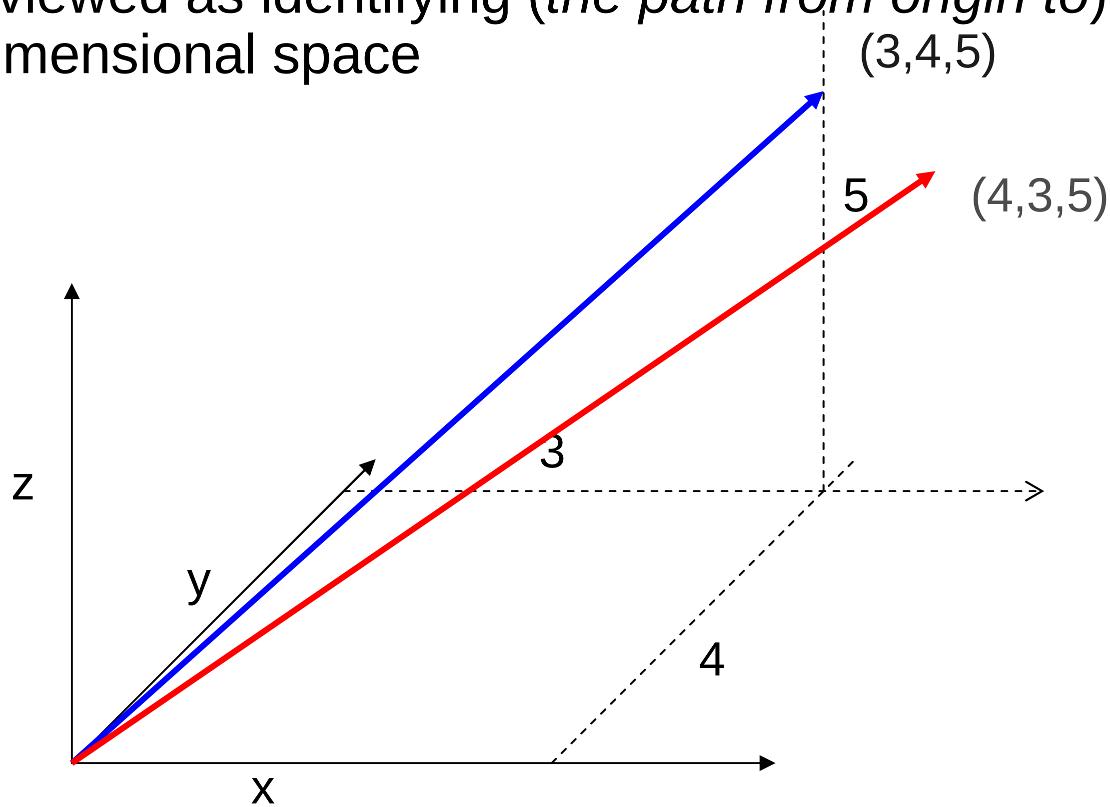
- 3 pacmen
- A 321x399 matrix
  - Row and Column = position
- A 3x128079 matrix
  - Triples of x,y and value
- A 1x128079 vector
  - “Unraveling” the matrix
- Note: All of these can be recast as the matrix that forms the image
  - Representations 2 and 4 are equivalent
    - The position is not represented

Example of a vector

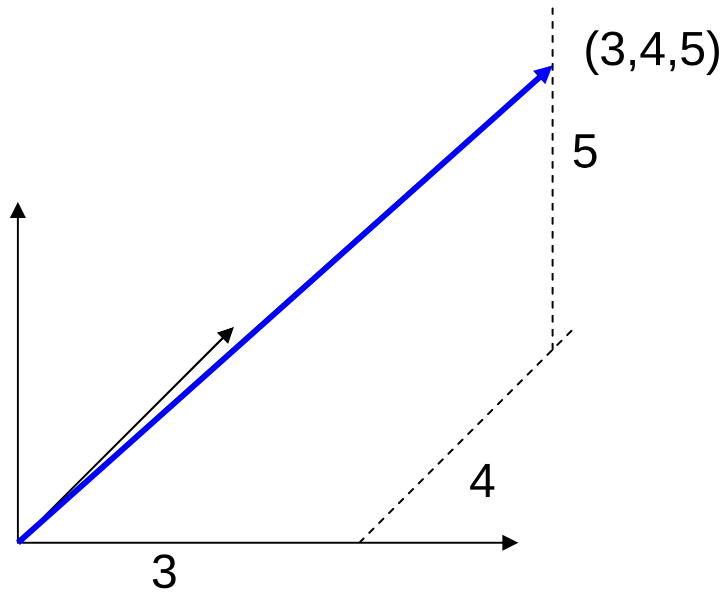
- Vectors usually hold sets of numerical attributes
  - X, Y, value
    - [1, 2, 0]
  - Earnings, losses, suicides
    - [$0 $1.000.000 3]
  - Etc …
- Consider a “relative Manhattan” vector
  - Provides a relative position by giving a number of avenues and streets to cross, e.g. [3av 33st]

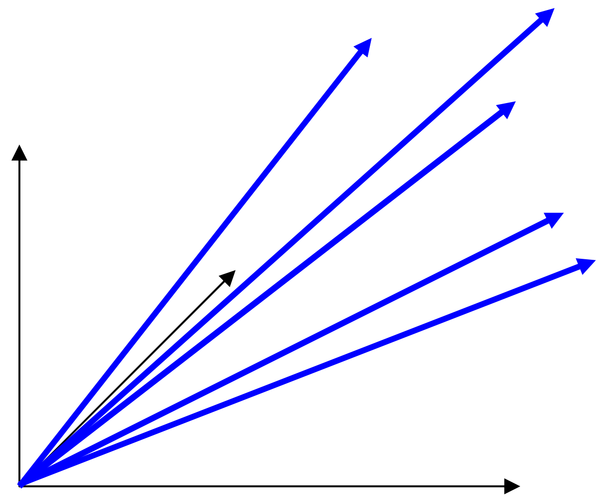
Vectors

- Ordered collection of numbers
  - Examples: [3 4 5], [a b c d], ..
  - [3 4 5] != [4 3 5] → **Order is important**
- Typically viewed as identifying (*the path from origin to*) a location in an N-dimensional space



Vectors vs. Matrices





- A vector is a geometric notation for how to get from (0,0) to some location in the space
- A matrix is simply a collection of destinations!
  - Properties of matrices are *average* properties of the traveller's path to these destinations

Basic arithmetic operations

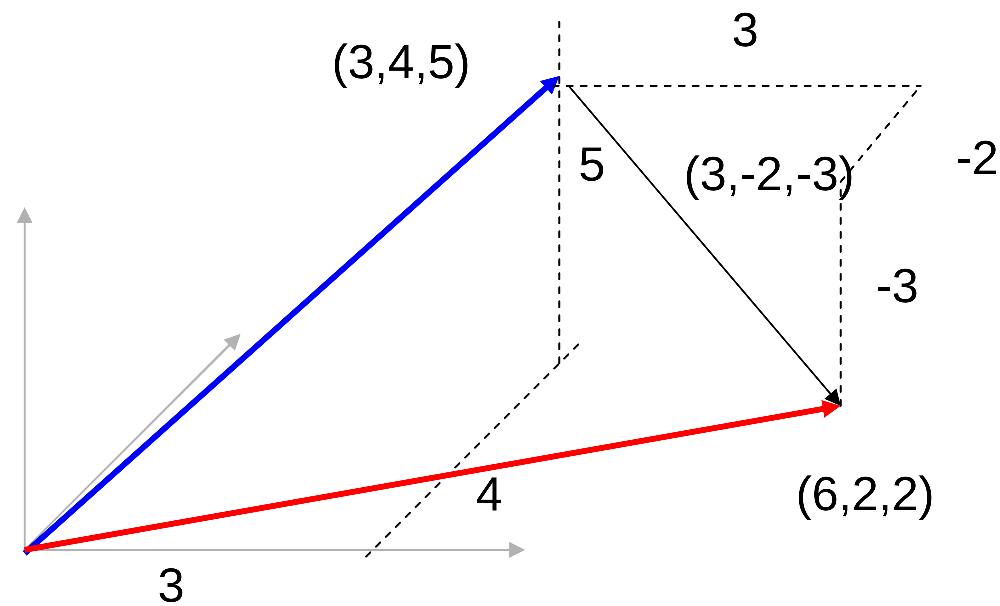
- Addition and subtraction
  - Element-wise operations

$$\mathbf{a}+\mathbf{b}=\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix}+\begin{bmatrix}b_1\\b_2\\b_3\end{bmatrix}=\begin{bmatrix}a_1+b_1\\a_2+b_2\\a_3+b_3\end{bmatrix}\qquad \mathbf{a}-\mathbf{b}=\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix}-\begin{bmatrix}b_1\\b_2\\b_3\end{bmatrix}=\begin{bmatrix}a_1-b_1\\a_2-b_2\\a_3-b_3\end{bmatrix}$$

$$\mathbf{A}+\mathbf{B}=\begin{bmatrix}a_{11} & a_{12}\\a_{21} & a_{22}\end{bmatrix}+\begin{bmatrix}b_{11} & b_{12}\\b_{21} & b_{22}\end{bmatrix}=\begin{bmatrix}a_{11}+b_{11} & a_{12}+b_{12}\\a_{21}+b_{21} & a_{22}+b_{22}\end{bmatrix}$$

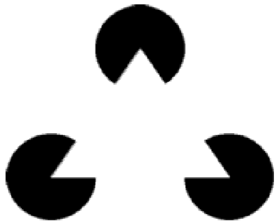
- MATLAB syntax: a+b and a - b

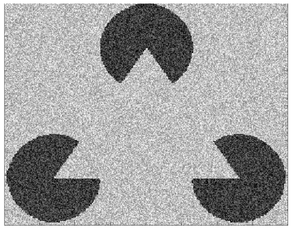
Vector Operations

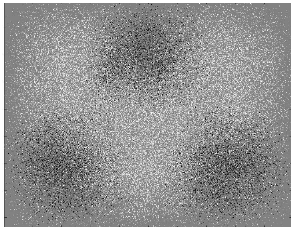


- Operations tell us how to get from ({0}) to the result of the vector operations
  - (3,4,5) + (3,-2,-3) = (6,2,2)

Operations example







```
>> X(1:32:end,1:40:end)

ans =

     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     0     1     1     1
     1     1     1     1     0     0     0     1     1     1
     1     1     1     1     0     1     0     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     0     1     1     1     1     1     0     1
     1     0     0     1     1     1     1     1     0     0
     1     0     0     0     1     1     1     0     0     0
     1     0     0     0     1     1     1     0     0     0
     1     1     1     1     1     1     1     1     1     1
```

$$[1\ \ 1\ \ .\ \ 1\ \ 1\ \ .\ \ 0\ \ 0\ \ 0\ \ .\ \ .\ \ 1]$$

$$\begin{bmatrix} 1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & 10 \\ 1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & 10 \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

```
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     0     1     1     1
     1     1     1     1     0     0     0     1     1     1
     1     1     1     1     0     1     0     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     0     1     1     1     1     1     0     1
     1     0     0     1     1     1     1     1     0     0
     1     0     0     0     1     1     1     0     0     0
     1     0     0     0     1     1     1     0     0     0
     1     1     1     1     1     1     1     1     1     1
```

+

0.6265	0.4839	0.7874	0.1749	0.3661	0.7716	0.9012	0.5111	0.1518	0.3528
0.2116	0.3506	0.4380	0.0707	0.2653	0.1263	0.6099	0.7610	0.8772	0.4465
0.1194	0.8242	0.2729	0.8895	0.0681	0.8501	0.4507	0.1967	0.6585	0.3526
0.9257	0.3736	0.5879	0.1068	0.9746	0.6336	0.1589	0.8425	0.0656	0.3100
0.6203	0.5834	0.5060	0.3234	0.5002	0.1915	0.2964	0.0938	0.4636	0.5114
0.5782	0.9409	0.5144	0.3392	0.4970	0.6307	0.4717	0.9053	0.4850	0.2467
0.8887	0.4086	0.7580	0.9547	0.7566	0.9898	0.5251	0.6518	0.8996	0.7946
0.0616	0.3867	0.4978	0.5169	0.0529	0.8565	0.5613	0.3270	0.8976	0.6088
0.6095	0.1383	0.6277	0.5475	0.6145	0.5146	0.1139	0.1981	0.3401	0.4395
0.3806	0.2752	0.5621	0.9369	0.6252	0.0422	0.6518	0.8614	0.5366	0.9572
0.1412	0.2130	0.6296	0.9251	0.6635	0.7859	0.2164	0.5335	0.1640	0.3909

$$\begin{bmatrix} 1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & 10 \\ 1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & 10 \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

+

Random(3,columns(M))

- Adding random values to different representations of the image

Vector norm

- Measure of how big a vector is:
  - Notated as $\|\mathbf{x}\|$

$$\left\|\begin{bmatrix} a & b & \ldots \end{bmatrix}\right\| = \sqrt{a^2 + b^2 + \ldots^2}$$

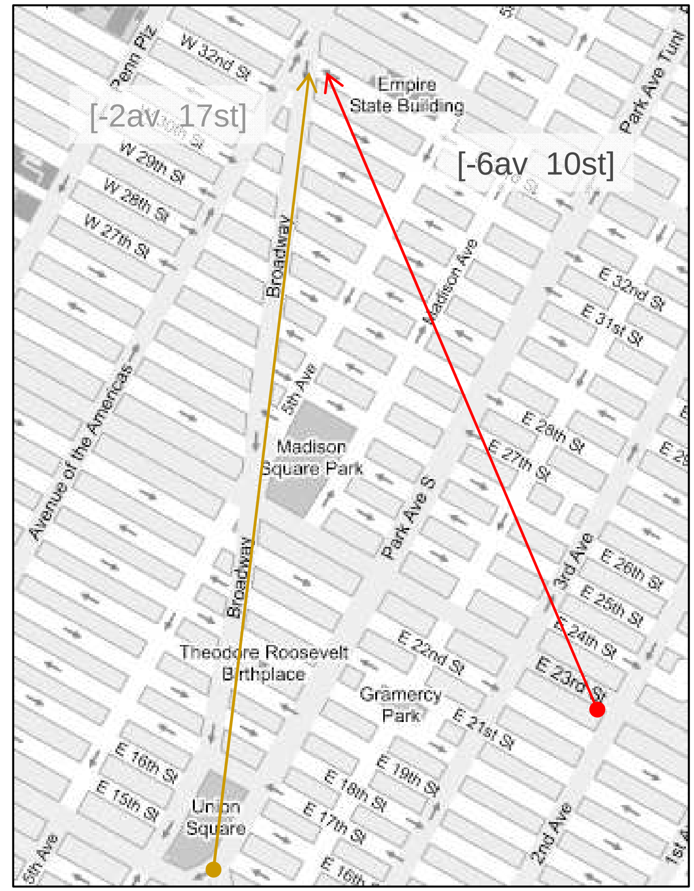
- In Manhattan vectors a measure of distance

$$\left\|\begin{bmatrix} -2 & 17 \end{bmatrix}\right\| = 17.11$$

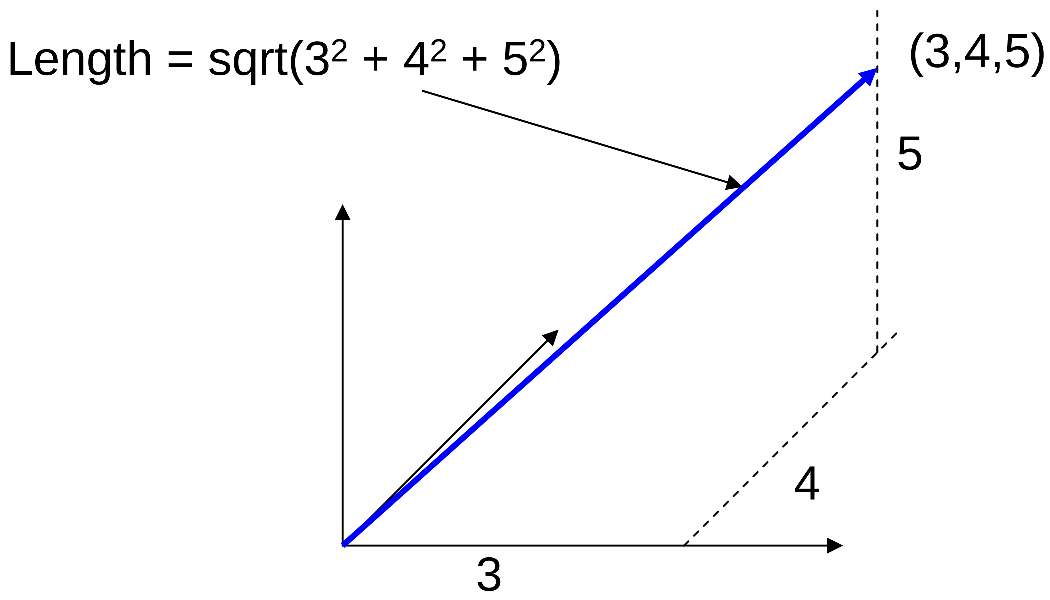
$$\left\|\begin{bmatrix} -6 & 10 \end{bmatrix}\right\| = 11.66$$

- MATLAB syntax:

```
norm(x)
```



Vector Norm



- Geometrically the shortest distance to travel from the origin to the destination
  - As the crow flies
  - Assuming Euclidean Geometry

Transposition

- A transposed row vector becomes a column
  (and vice versa)

$$\mathbf{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{x}^T = \begin{bmatrix} a & b & c \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} a & b & c \end{bmatrix}, \quad \mathbf{y}^T = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

- A transposed matrix gets all its row (or column) vectors transposed in order

$$\mathbf{X} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}, \quad \mathbf{X}^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$$

$\mathbf{M} =$ , $\mathbf{M}^T =$ 

- MATLAB syntax: a'

Vector multiplication

- Multiplication is not element-wise!
- Dot product, or inner product
  - Vectors must have the same number of elements
  - Row vector times column vector = scalar

$$\begin{bmatrix} a & b & c \end{bmatrix} \cdot \begin{bmatrix} d \\ e \\ f \end{bmatrix} = a \cdot d + b \cdot e + c \cdot f$$

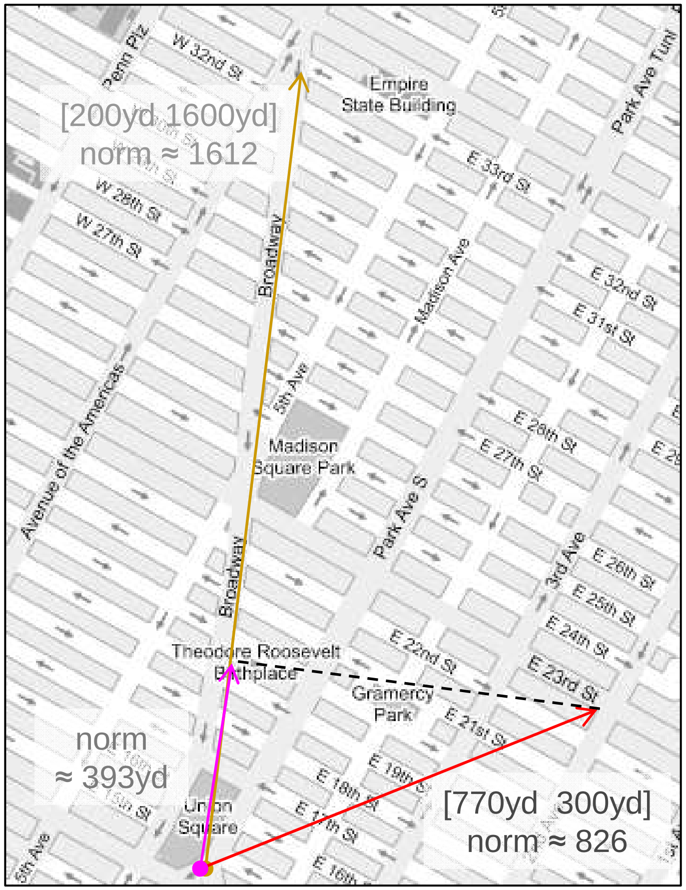
- Cross product, outer product or vector direct product
  - Column vector times row vector = matrix

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot \begin{bmatrix} d & e & f \end{bmatrix} = \begin{bmatrix} a \cdot d & a \cdot e & a \cdot f \\ b \cdot d & b \cdot e & b \cdot f \\ c \cdot d & c \cdot e & c \cdot f \end{bmatrix}$$

- MATLAB syntax: a*b

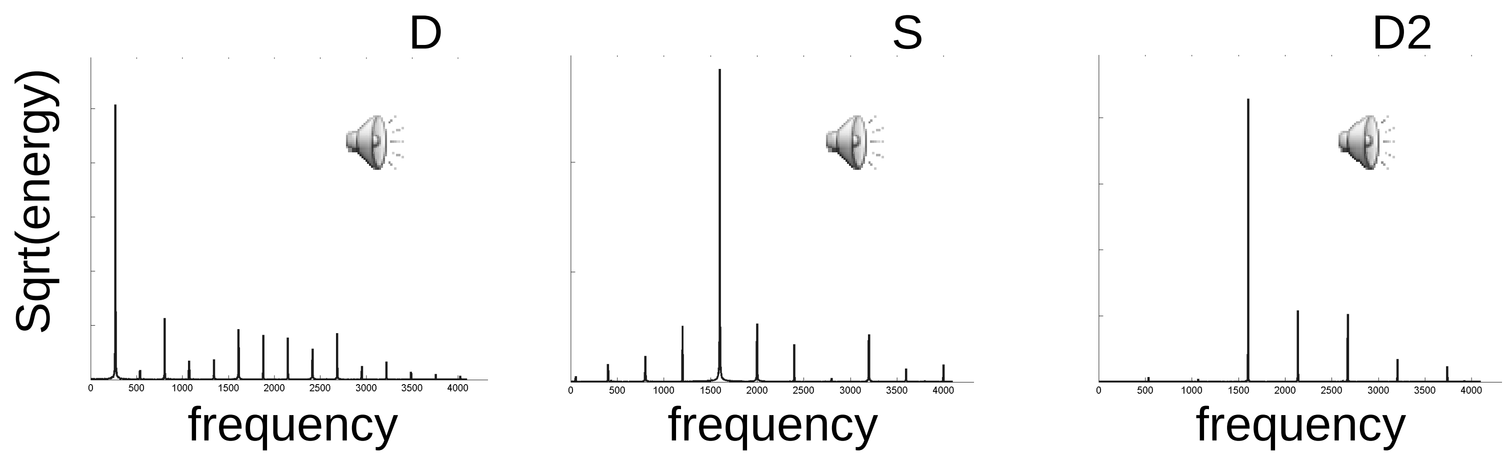
Vector ***dot product*** in Manhattan

- Multiplying the "yard" vectors
  - Instead of avenue/street we'll use yards
  - $\mathbf{a} = [200\ 1600]$, $\mathbf{b} = [770\ 300]$
- The dot product of the two vectors relates to the length of a *projection*
  - How much of the first vector have we covered by following the second one?
  - The answer comes back as a unit of the first vector so we divide by its length



$$\frac{\mathbf{a}\cdot\mathbf{b}^T}{\|\mathbf{a}\|} = \frac{[200 \quad 1600]\cdot\begin{bmatrix}770\\300\end{bmatrix}}{\|[200 \quad 1600]\|} \approx 393\text{yd}$$

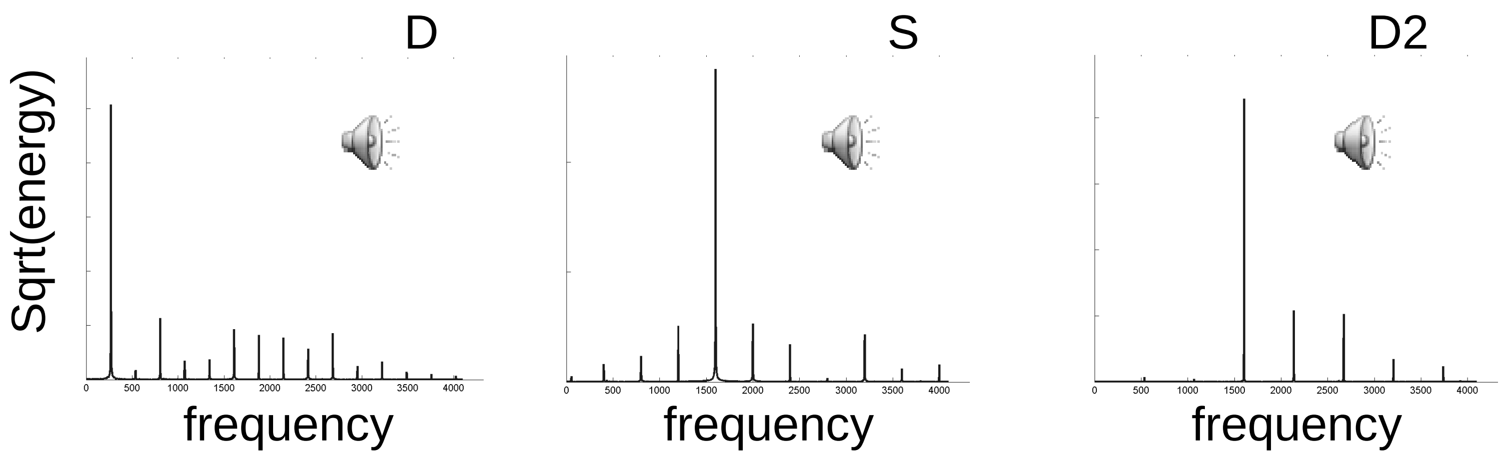
Vector dot product



$[11\ \ 9\ \ .\ \ 0\ \ 54\ \ 1\ \ .\ \ .\ \ .\ \ 1]$ $[3\ \ .\ \ 24\ \ .\ \ .\ \ 16\ \ .\ \ 14\ \ .\ \ 1]$ $[0\ \ .\ \ 0\ \ .\ \ 3\ \ 0\ \ .\ \ 13\ \ .\ \ 0]$

- Vectors are spectra
  - Energy at a discrete set of frequencies
  - Actually 1x4096
  - X axis is the *index* of the number in the vector
    - Represents frequency
  - Y axis is the value of the number in the vector
    - Represents magnitude

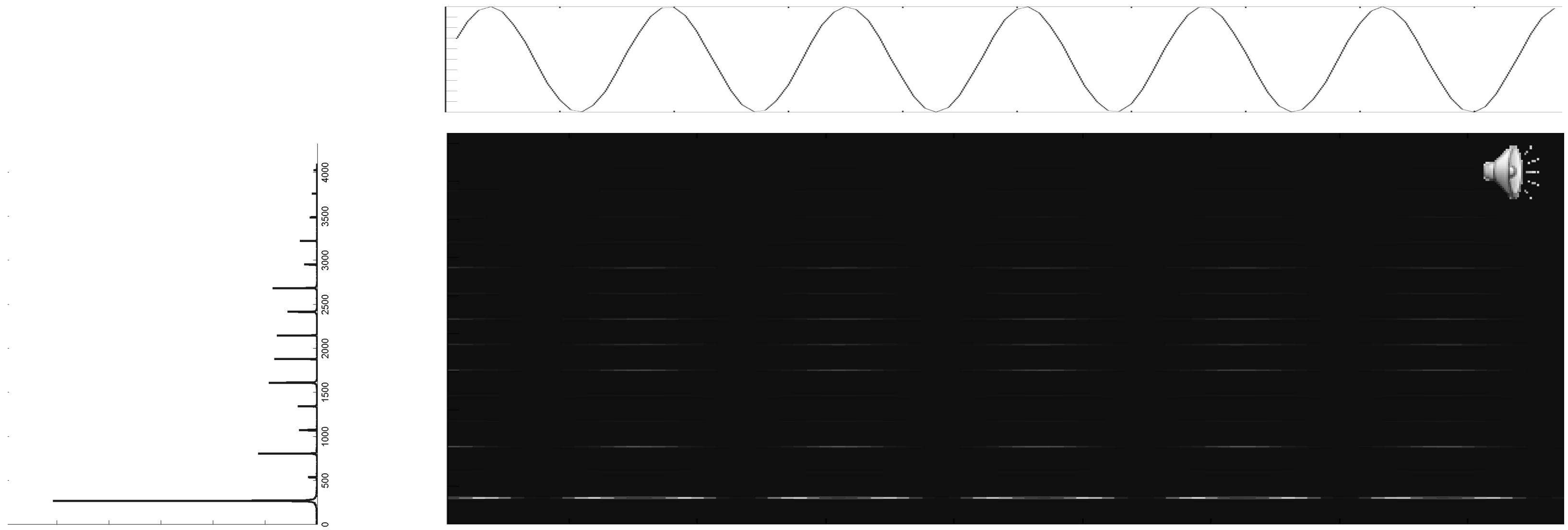
Vector dot product



$[11 \quad 9 \quad . \quad 0 \quad 54 \quad 1 \quad . \quad . \quad . \quad 1]$ $[3 \quad . \quad 24 \quad . \quad . \quad 16 \quad . \quad 14 \quad . \quad 1]$ $[0 \quad . \quad 0 \quad . \quad 3 \quad 0 \quad . \quad 13 \quad . \quad 0]$

- How much of D is also in S
  - How much can you fake a D by playing an S
  - D.S / |D||S| = 0.1
  - Not very much
- How much of D is in D2?
  - D.D2 / |D| /|D2| = 0.5
  - Not bad, you can fake it
- To do this, D, S, and D2 *must be the same size*

Vector cross product



- The column vector is the spectrum
- The row vector is an amplitude modulation
- The crossproduct is a spectrogram
  - Shows how the energy in each frequency varies with time
  - The pattern in each column is a scaled version of the spectrum
  - Each row is a scaled version of the modulation

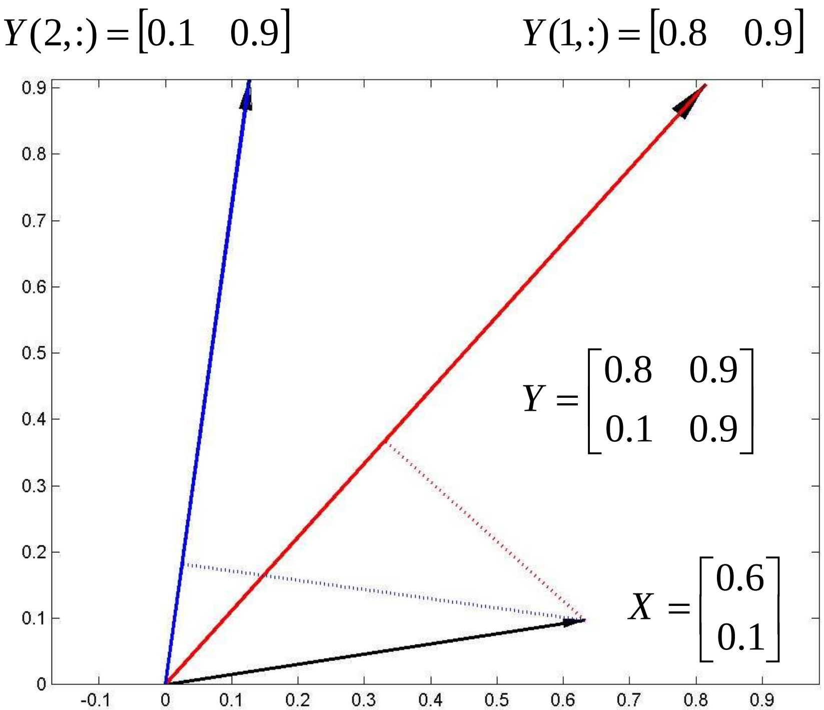
Matrix multiplication

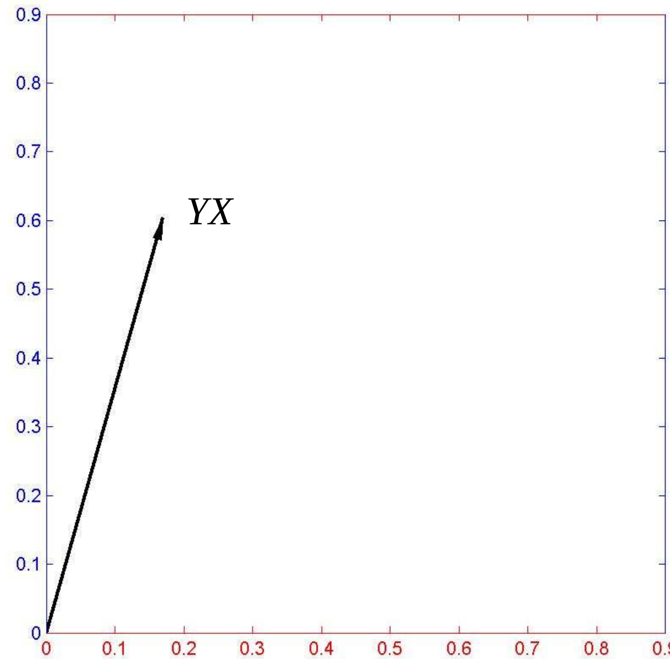
- Generalization of vector multiplication
  - Dot product of each vector pair

$$\mathbf{A}\cdot\mathbf{B} = \begin{bmatrix} \leftarrow & \mathbf{a}_1 & \rightarrow \\ \leftarrow & \mathbf{a}_2 & \rightarrow \end{bmatrix} \cdot \begin{bmatrix} \uparrow & \uparrow \\ \mathbf{b}_1 & \mathbf{b}_2 \\ \downarrow & \downarrow \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1\cdot\mathbf{b}_1 & \mathbf{a}_1\cdot\mathbf{b}_2 \\ \mathbf{a}_2\cdot\mathbf{b}_1 & \mathbf{a}_2\cdot\mathbf{b}_2 \end{bmatrix}$$

  - Dimensions must match!!
    - Columns of first matrix = rows of second
    - Result inherits the number of rows from the first matrix and the number of columns from the second matrix
- MATLAB syntax: `a*b`

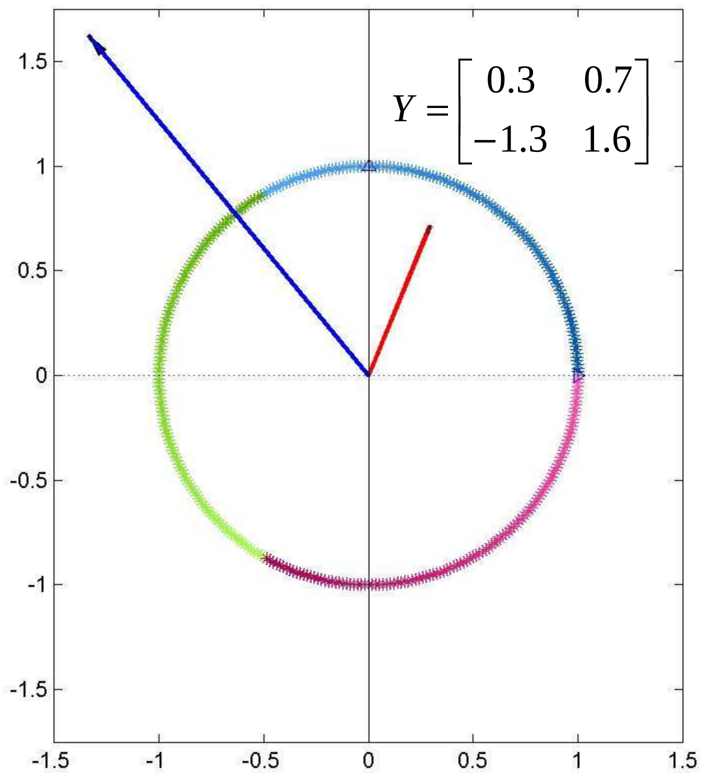
Multiplying a Vector by a Matrix

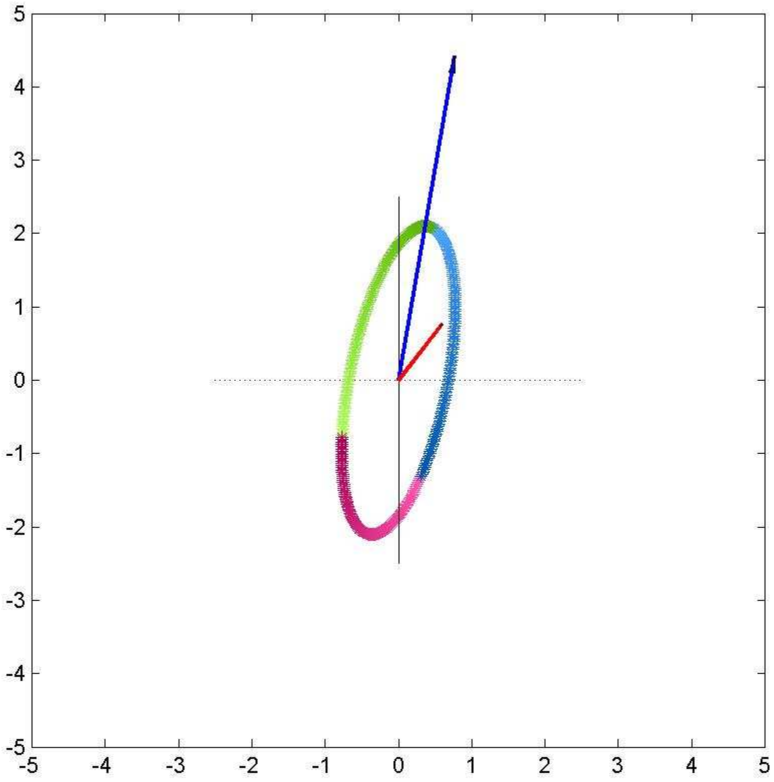




- Multiplication of a vector X by a matrix Y expresses the vector X in terms of projections of X on the row vectors of the matrix Y
  - It scales and rotates the vector
  - Alternately viewed, it scales and rotates the space – the underlying plane

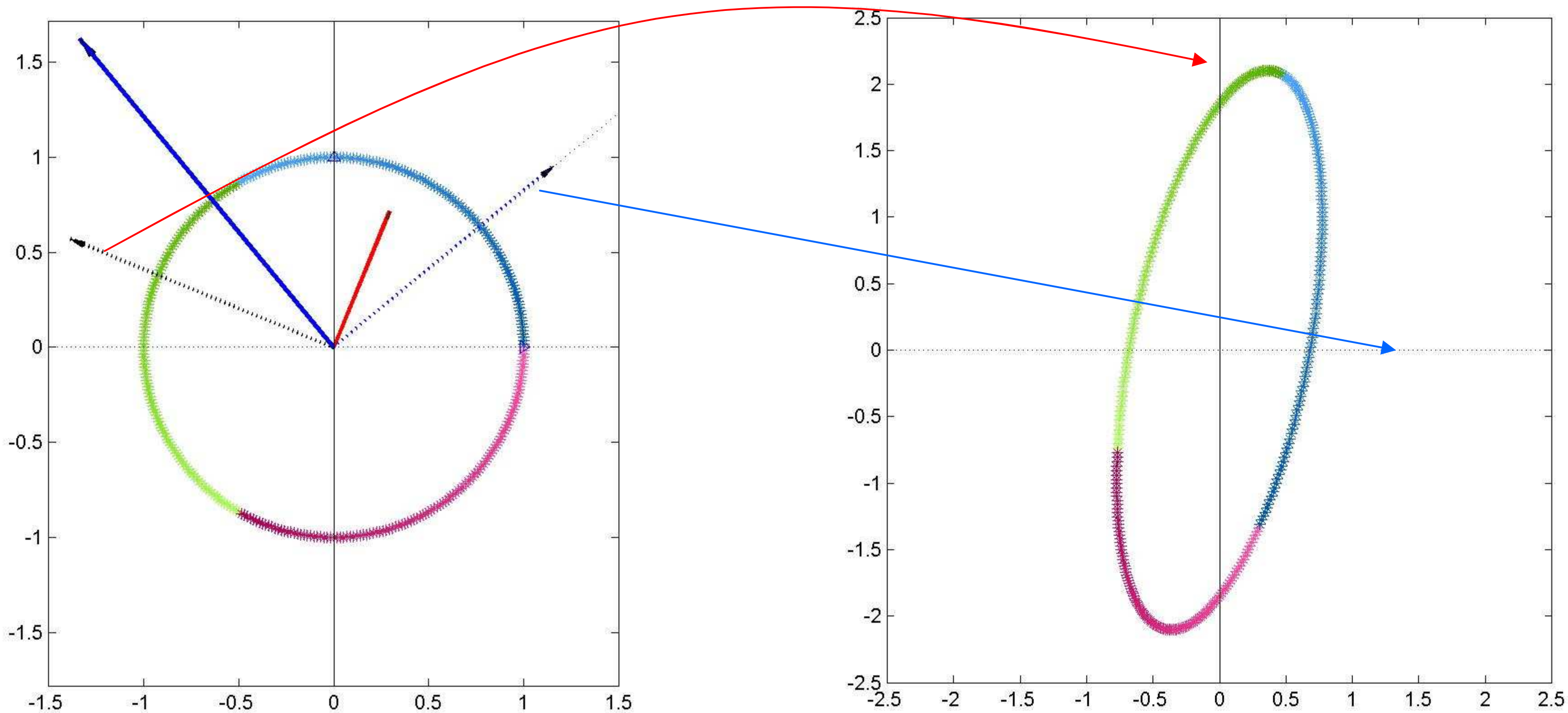
Matrix Multiplication





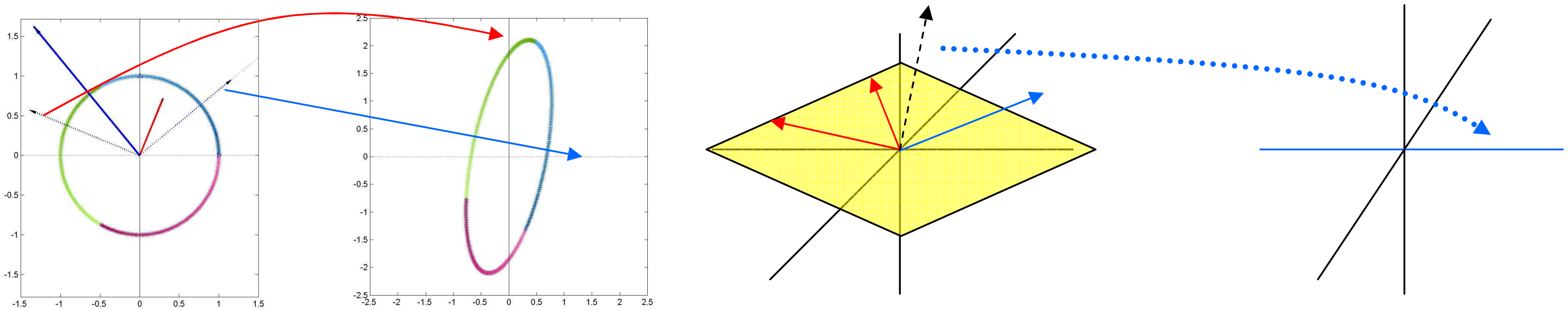
- The matrix rotates and scales the space
  - Including its own vectors

Matrix Multiplication



- The *normals* to the row vectors in the matrix become the new axes
  - X axis = normal to the *second* row vector
    - Scaled by the inverse of the length of the *first* row vector

Matrix Multiplication is projection



- The k-th axis corresponds to the normal to the hyperplane represented by the 1..k-1,k+1..N-th row vectors in the matrix
  - Any set of K-1 vectors represent a hyperplane of dimension K-1 or less
- The distance along the new axis equals the length of the projection on the k-th row vector
  - Expressed in inverse-lengths of the vector

Matrix Multiplication: Column space

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = x\begin{bmatrix} a \\ d \end{bmatrix} + y\begin{bmatrix} b \\ e \end{bmatrix} + z\begin{bmatrix} c \\ f \end{bmatrix}$$

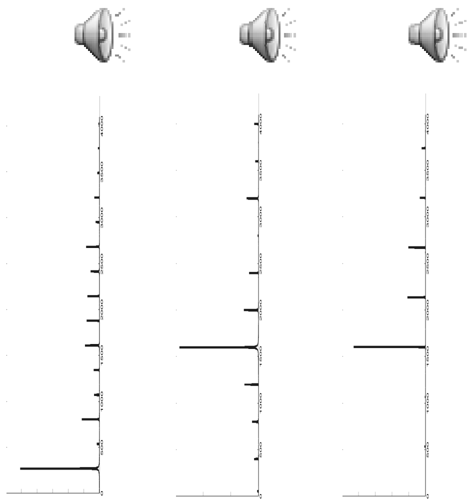
- So much for spaces .. what does multiplying a matrix by a vector really do?
- It *mixes* the column vectors of the matrix using the numbers in the vector
- The *column space* of the Matrix is the complete set of all vectors that can be formed by mixing its columns

Matrix Multiplication: Row space

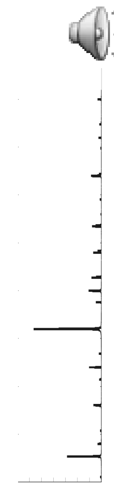
$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} = x\begin{bmatrix} a & b & c \end{bmatrix} + y\begin{bmatrix} d & e & f \end{bmatrix}$$

- Left multiplication mixes the *row vectors* of the matrix.
- The *row space* of the Matrix is the complete set of all vectors that can be formed by mixing its rows

Matrix multiplication: Mixing vectors

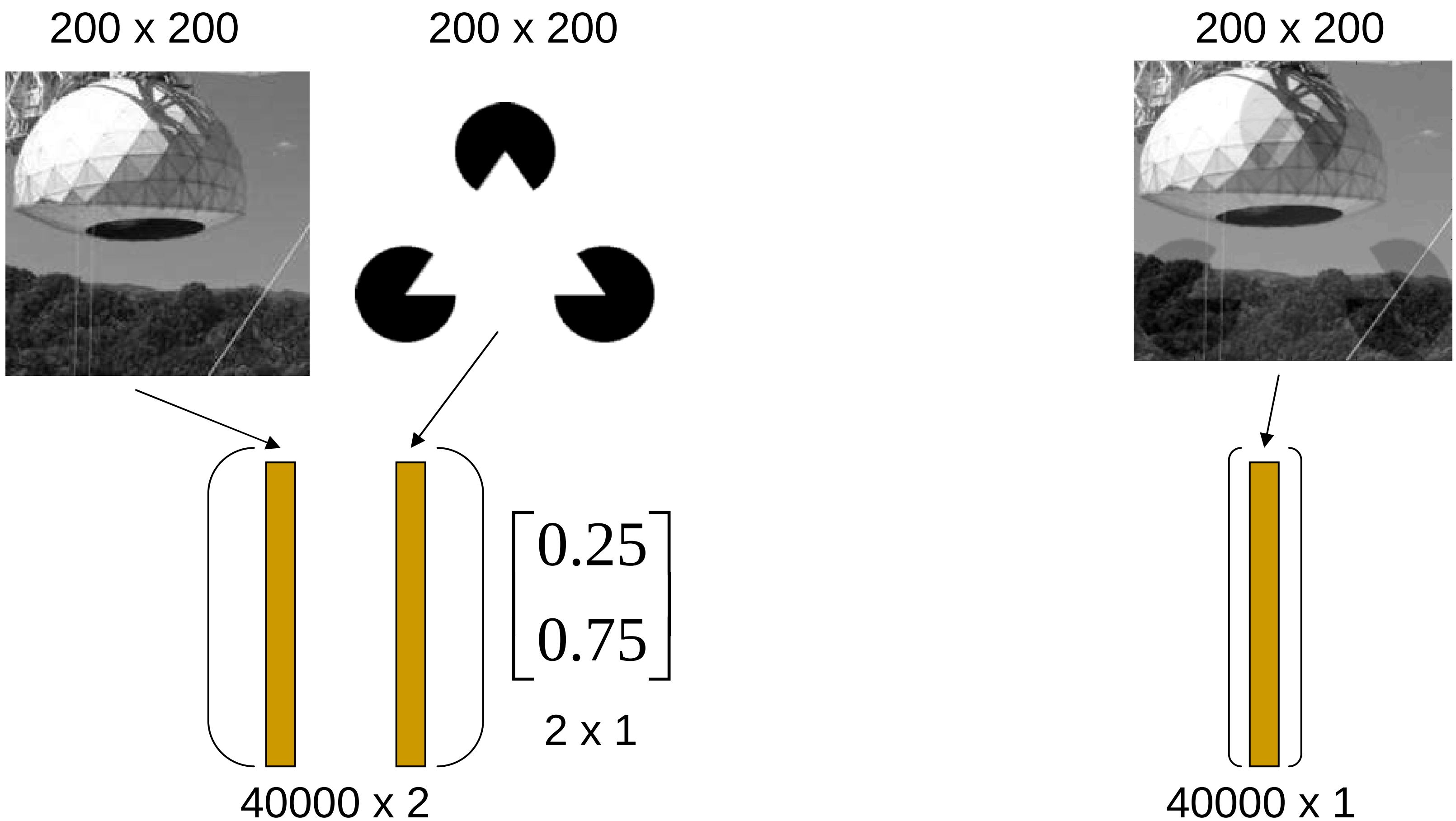


$$\overset{X}{\begin{bmatrix} 1 & 3 & 0 \\ . & . & 0 \\ 9 & 24 & . \\ . & . & 1 \end{bmatrix}} \overset{Y}{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}} = \begin{bmatrix} 7 \\ . \\ . \\ 2 \end{bmatrix}$$



- A physical example
  - The three column vectors of the matrix X are the spectra of three notes
  - The multiplying column vector Y is just a mixing vector
  - The result is a sound that is the mixture of the three notes

Matrix multiplication: Mixing vectors



- Mixing two images
  - The images are arranged as columns
    - position value not included
  - The result of the multiplication is rearranged as an image

Administrivia

- New classroom!!
  - PH 125C
    - Seats 70! Bring your friends.
- Registration: All students on waitlist are registered
- TA: Not yet :-/
- Homework: Against “class3” on mlsp.cs.cmu.edu
  - Transcribing music
  - Feel free to discuss amongst yourselves
  - Use the discussion lists on blackboard.andrew.cmu.edu
- No class next week
  - You will get email from me with updates
- Blackboard – if you are not registered on blackboard please register

Matrix multiplication: another view

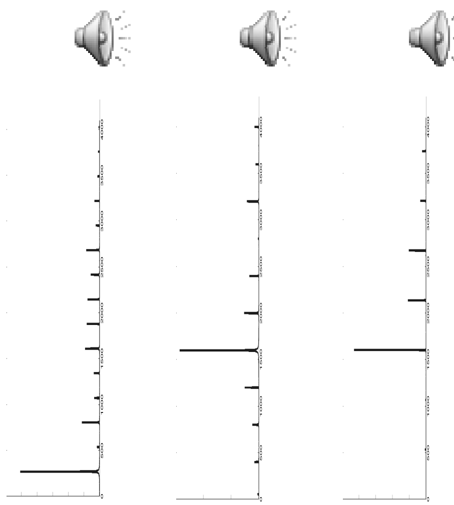
$$\mathbf{A}\cdot\mathbf{B} = \begin{bmatrix} a_{11} & . & . & a_{1N} \\ a_{21} & . & . & a_{2N} \\ . & . & . & . \\ a_{M1} & . & . & a_{MN} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & . & b_{NK} \\ . & . & . \\ b_{N1} & . & b_{NK} \end{bmatrix} = \begin{bmatrix} \sum_k a_{1k} b_{k1} & . & \sum_k a_{1k} b_{kK} \\ . & . & . \\ \sum_k a_{Mk} b_{k1} & . & \sum_k a_{Mk} b_{kK} \end{bmatrix}$$

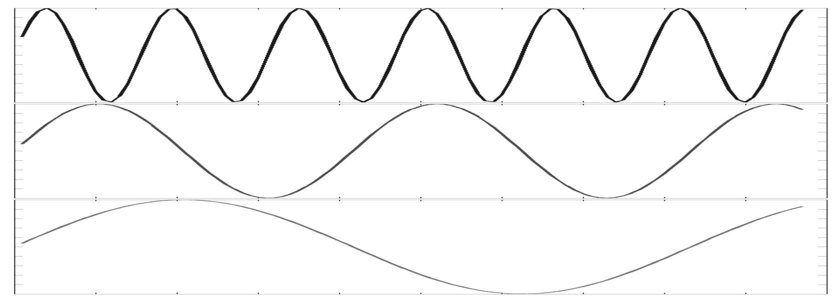
- What does this mean?

$$\begin{bmatrix} a_{11} & . & . & a_{1N} \\ a_{21} & . & . & a_{2N} \\ . & . & . & . \\ a_{M1} & . & . & a_{MN} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & . & b_{NK} \\ . & . & . \\ b_{N1} & . & b_{NK} \end{bmatrix} = \begin{bmatrix} a_{11} \\ . \\ . \\ a_{M1} \end{bmatrix} \begin{bmatrix} b_{11} & . & b_{1K} \end{bmatrix} + \begin{bmatrix} a_{12} \\ . \\ . \\ a_{M2} \end{bmatrix} \begin{bmatrix} b_{21} & . & b_{2K} \end{bmatrix} + \ldots + \begin{bmatrix} a_{1N} \\ . \\ . \\ a_{MN} \end{bmatrix} \begin{bmatrix} b_{N1} & . & b_{NK} \end{bmatrix}$$

- The outer product of the first column of A and the first row of B + outer product of the second column of A and the second row of B + ….

Why is that useful?





$$\begin{bmatrix} 1 & 3 & 0 \\ . & . & 0 \\ 9 & 24 & . \\ . & . & 1 \end{bmatrix}$$

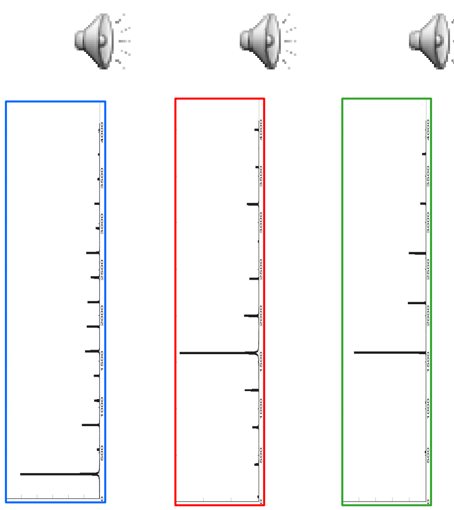
X

$$\begin{bmatrix} 0 & 0.5 & 0.75 & 1 & 0.75 & 0.5 & 0 & . & . & . & . & . \\ 1 & 0.9 & 0.7 & 0.5 & 0 & 0.5 & . & . & . & . & . & . \\ 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 0.95 & 1 & . & . & . & . & . \end{bmatrix}$$

Y

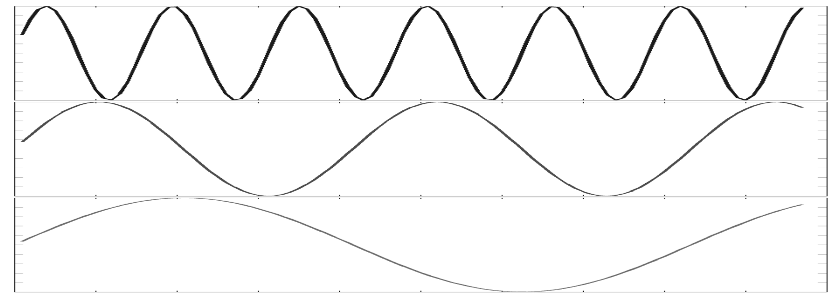
- Sounds: Three notes modulated independently

Matrix multiplication: Mixing modulated spectra



$$\begin{bmatrix} 1 & 3 & 0 \\ . & . & 0 \\ 9 & 24 & . \\ . & . & 1 \end{bmatrix}$$

X

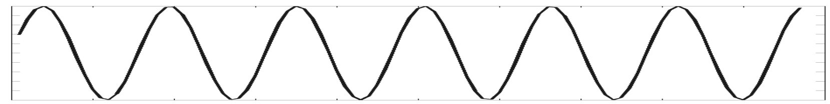


$$\begin{bmatrix} 0 & 0.5 & 0.75 & 1 & 0.75 & 0.5 & 0 & . & . & . & . & . \\ 1 & 0.9 & 0.7 & 0.5 & 0 & 0.5 & . & . & . & . & . & . \\ 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 0.95 & 1 & . & . & . & . & . \end{bmatrix}$$

Y

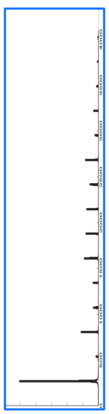
- Sounds: Three notes modulated independently

Matrix multiplication: Mixing modulated spectra



$$\begin{bmatrix} 0 & 0.5 & 0.75 & 1 & 0.75 & 0.5 & 0 & . & . & . & . & . \end{bmatrix}$$

Y

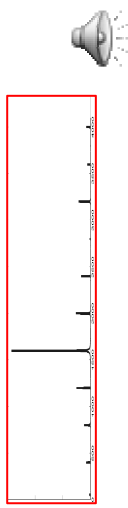


$$\begin{bmatrix} 1 \\ . \\ 9 \\ . \end{bmatrix}$$

X

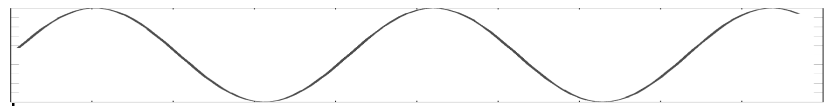
- Sounds: Three notes modulated independently

Matrix multiplication: Mixing modulated spectra

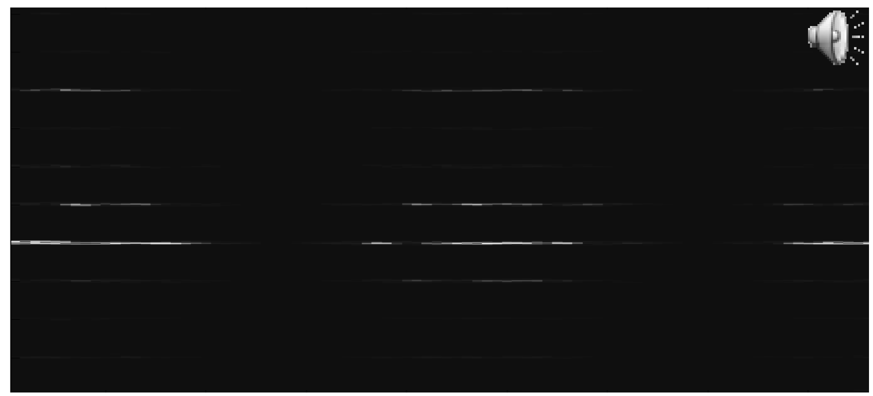


$$\begin{bmatrix} 3 \\ . \\ 24 \\ . \end{bmatrix}$$

X

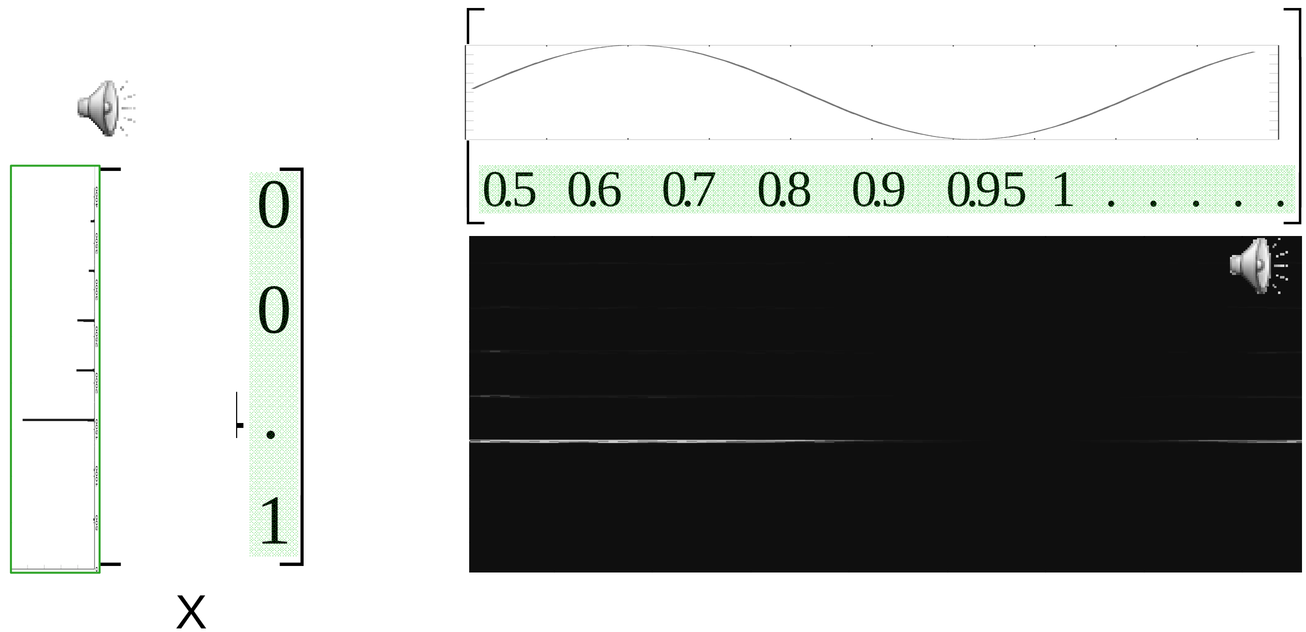


$$\begin{bmatrix} 1 & 0.9 & 0.7 & 0.5 & 0 & 0.5 & . & . & . & . & . & . \end{bmatrix}$$



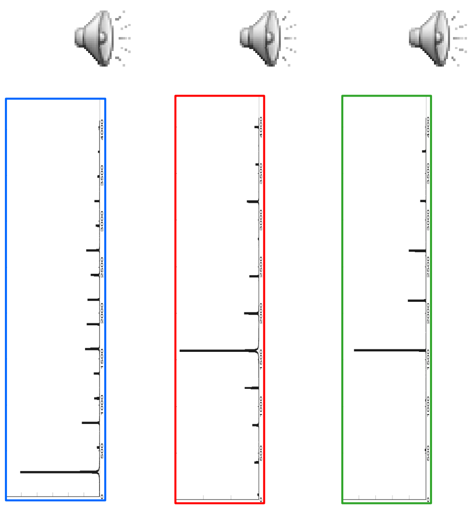
- Sounds: Three notes modulated independently

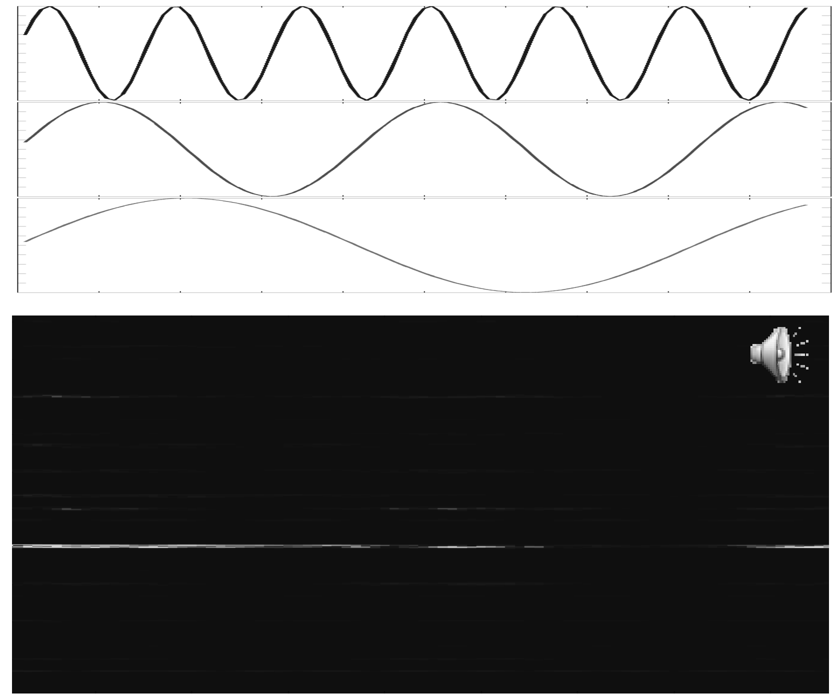
Matrix multiplication: Mixing modulated spectra



- Sounds: Three notes modulated independently

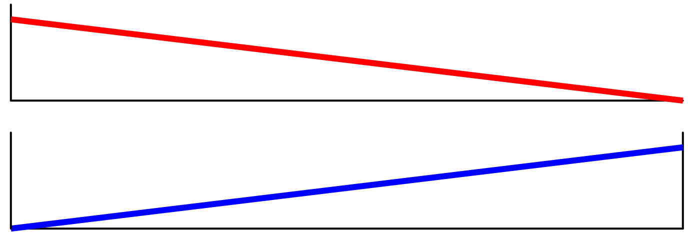
Matrix multiplication: Mixing modulated spectra



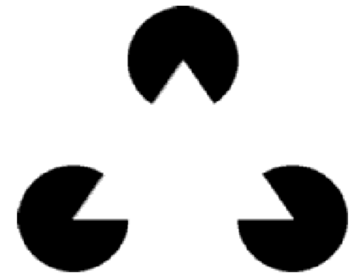


- Sounds: Three notes modulated independently

Matrix multiplication: Image transition

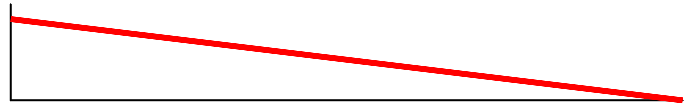


$$\begin{bmatrix} i_1 & j_1 \\ i_2 & j_2 \\ . & . \\ . & . \\ . & . \end{bmatrix} \begin{bmatrix} 1 & .9 & .8 & .7 & .6 & .5 & .4 & .3 & .2 & .1 & 0 \\ 0 & .1 & .2 & .3 & .4 & .5 & .6 & .7 & .8 & .9 & 1 \end{bmatrix}$$



- Image1 fades out linearly
- Image 2 fades in linearly

Matrix multiplication: Image transition

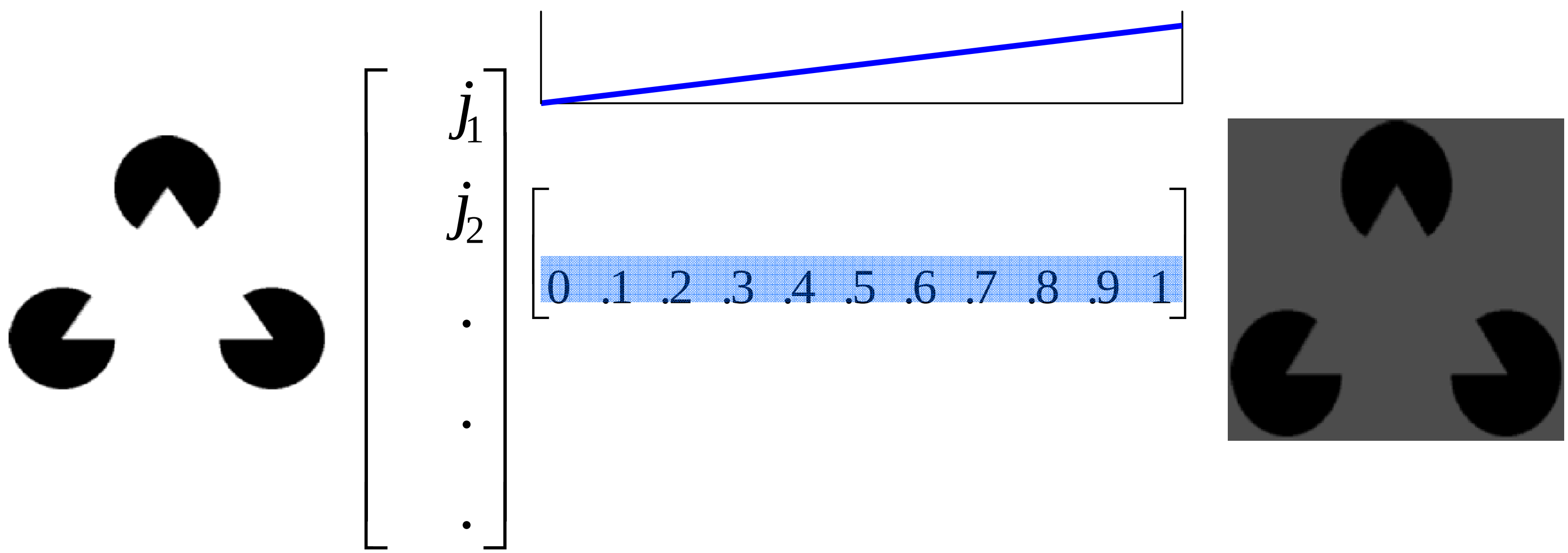


$$\begin{bmatrix} i_1 \\ i_2 \\ . \\ . \\ . \end{bmatrix} \begin{bmatrix} 1 & .9 & .8 & .7 & .6 & .5 & .4 & .3 & .2 & .1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} i_1 & 0.9i_1 & 0.8i_1 & . & . & . & . & . & . & 0 \\ i_2 & 0.9i_2 & 0.8i_2 & . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . & 0. \\ i_N & 0.9i_N & 0.8i_N & . & . & . & . & . & . & 0 \end{bmatrix}$$

- Each column is one image
  - The columns represent a sequence of images of decreasing intensity
- Image1 fades out linearly

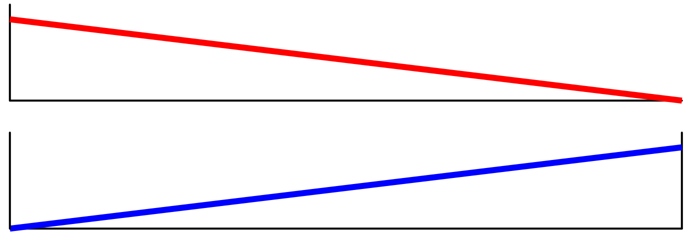
Matrix multiplication: Image transition



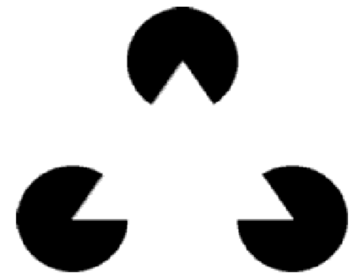
- Image 2 fades in linearly

Matrix multiplication: Image transition

$$\begin{bmatrix} i_1 & j_1 \\ i_2 & j_2 \\ . & . \\ . & . \\ . & . \end{bmatrix}$$



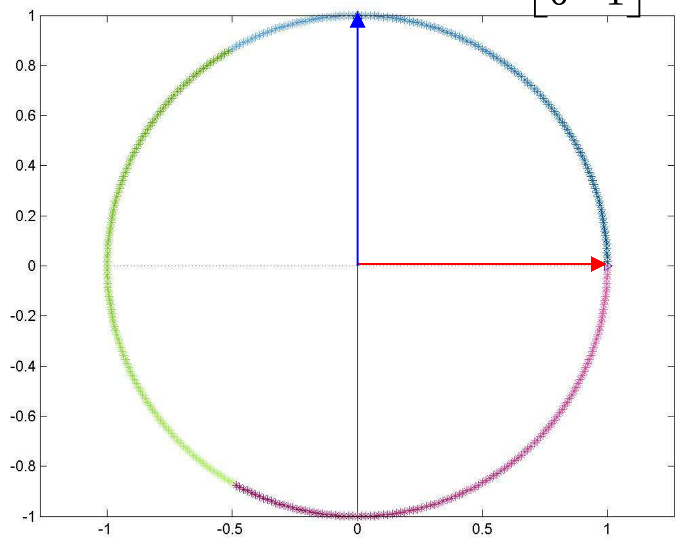
$$\begin{bmatrix} 1 & .9 & .8 & .7 & .6 & .5 & .4 & .3 & .2 & .1 & 0 \\ 0 & .1 & .2 & .3 & .4 & .5 & .6 & .7 & .8 & .9 & 1 \end{bmatrix}$$

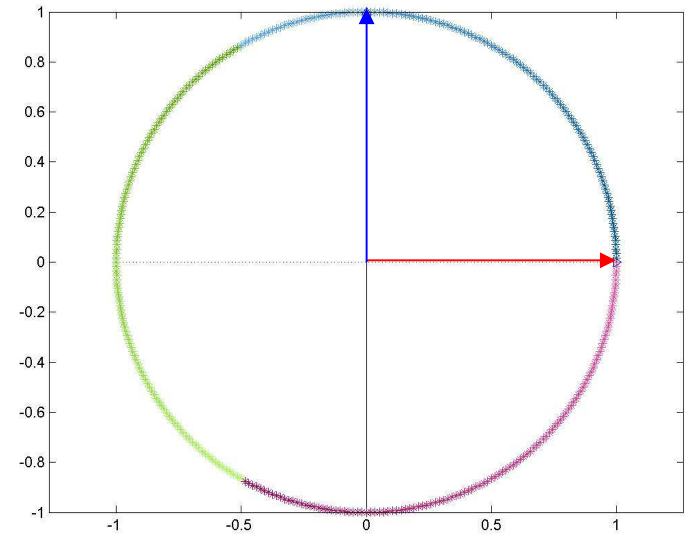


- Image1 fades out linearly
- Image 2 fades in linearly

The Identity Matrix

$$Y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

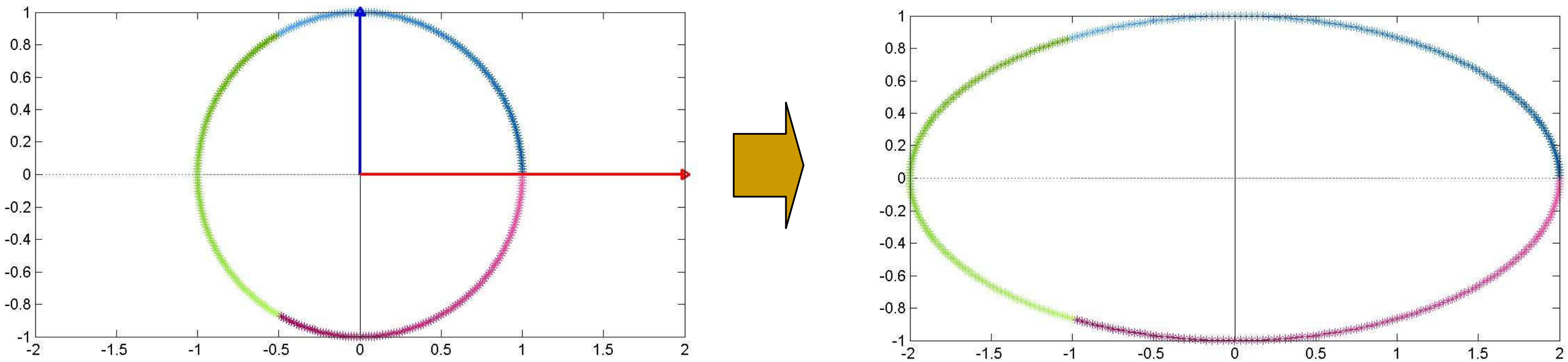




- An identity matrix is a square matrix where
  - All diagonal elements are 1.0
  - All off-diagonal elements are 0.0
- Multiplication by an identity matrix does not change vectors

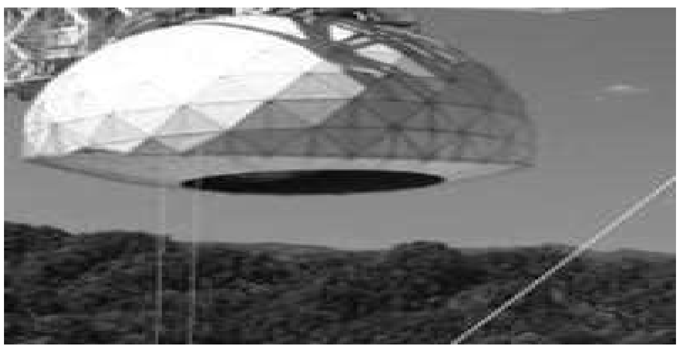
Diagonal Matrix

$$Y = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$



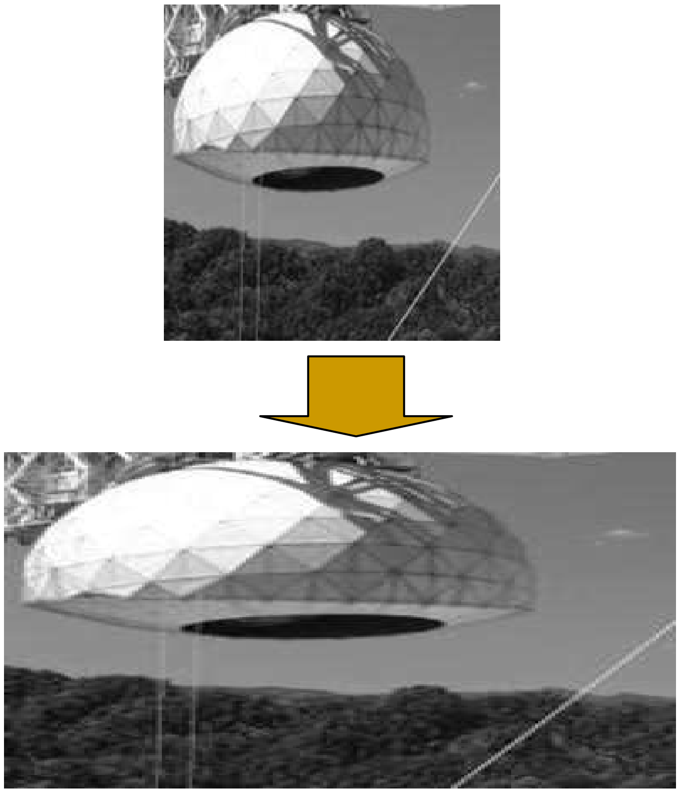
- All off-diagonal elements are zero
- Diagonal elements are non-zero
- Scales the axes
  - May flip axes

Diagonal matrix to transform images



- How?

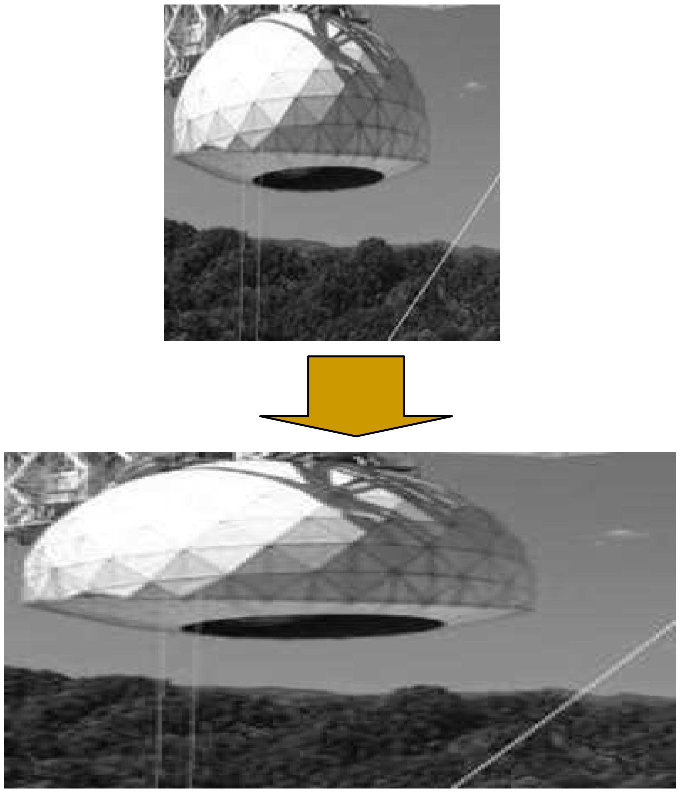
Stretching



$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & 10 \\ 1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & 10 \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

- Location-based representation
- Scaling matrix – only scales the X axis
  - The Y axis and pixel value are scaled by identity
- Not a good way of scaling.

Stretching



$$D = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & .5 & 0 & 0 & . \\ 0 & .5 & 1 & .5 & . \\ 0 & 0 & 0 & .5 & . \\ 0 & 0 & 0 & 0 & . \\ . & . & . & . & . \end{bmatrix} (N\mathrm{x}2N)$$

$$\mathrm{Newpic} = EA$$

- Better way

Modifying color

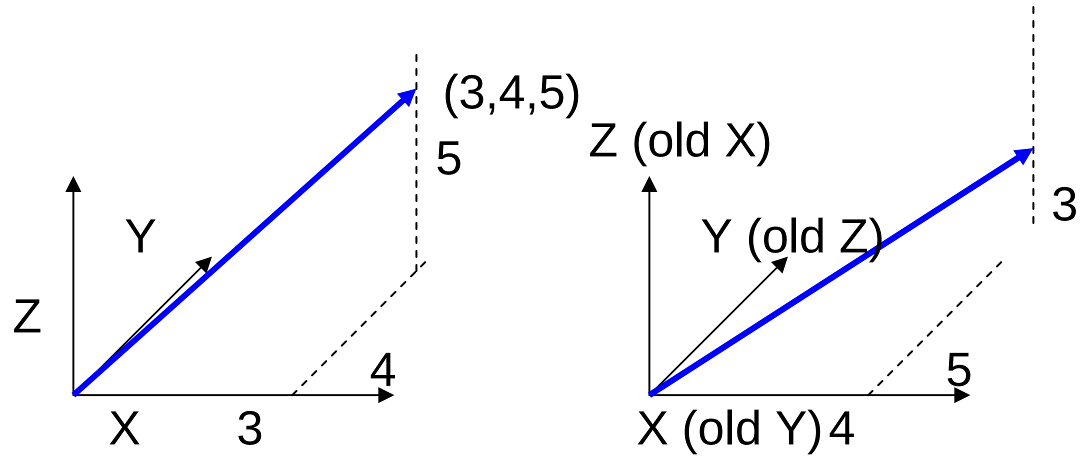
$$P = \begin{bmatrix} R & G & B \end{bmatrix}$$

$$Newpic = P\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Scale only Green

Permutation Matrix

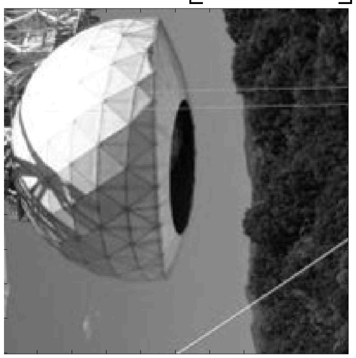
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} y \\ z \\ x \end{bmatrix}$$



- A permutation matrix simply rearranges the axes
  - The row entries are axis vectors in a different order
  - The result is a combination of rotations and reflections
- The permutation matrix effectively *permutes* the arrangement of the elements in a vector

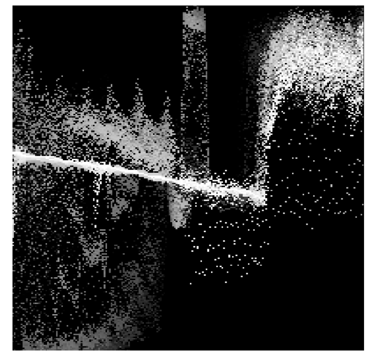
Permutation Matrix

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & 10 \\ 1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & 10 \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

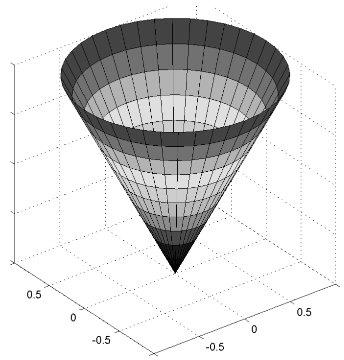
$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$



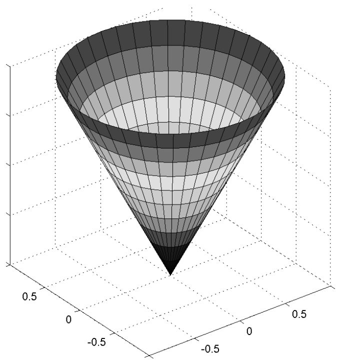
- Reflections and 90 degree rotations of images and objects

Permutation Matrix

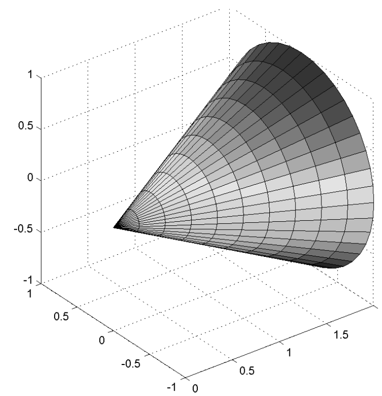
$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$



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



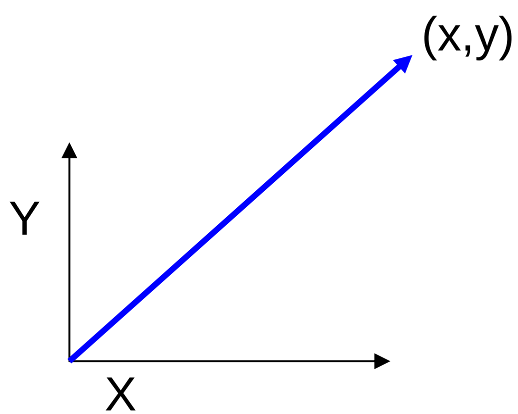
- Reflections and 90 degree rotations of images and objects
  - Object represented as a matrix of 3-Dimensional "position" vectors
  - Positions identify each point on the surface

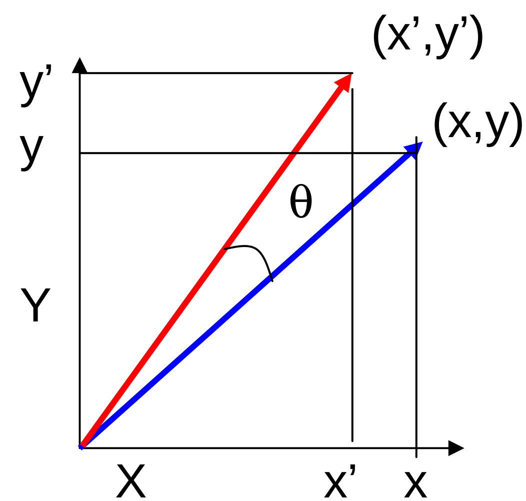
Rotation Matrix

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$
$$X = \begin{bmatrix} x \\ y \end{bmatrix}$$
$$X_{new} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

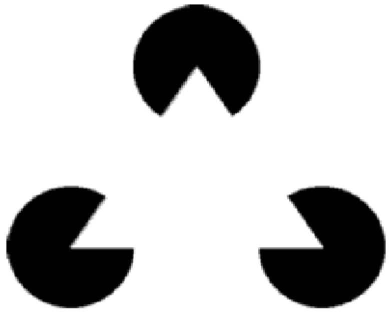
$$R_\theta X = X_{new}$$



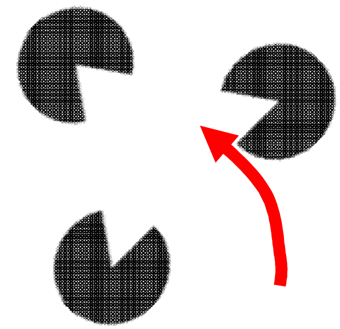


- A rotation matrix *rotates* the vector by some angle θ
- Alternately viewed, it rotates the axes
  - The new axes are at an angle θ to the old one

Rotating a picture



$$R = \begin{bmatrix} \cos 45 & -\sin 45 & 0 \\ \sin 45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

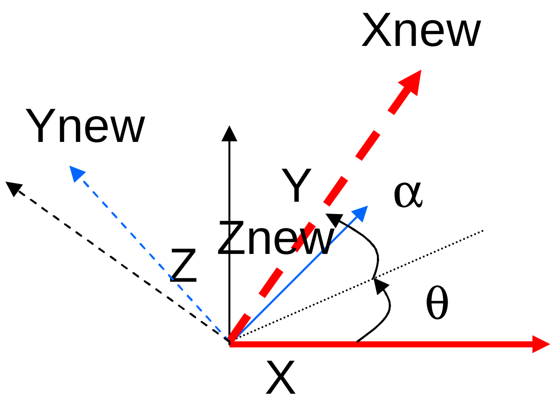


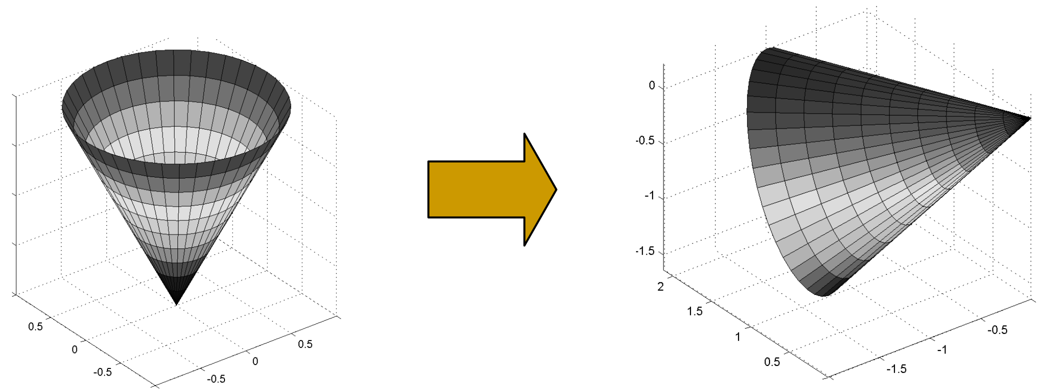
$$\begin{bmatrix} 1 & 1 & . & 2 & . & 2 & 2 & . & 2 & . & . \\ 1 & 2 & . & 1 & . & 5 & 6 & . & 10 & . & . \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & -\sqrt{2} & . & \sqrt{2} & . & -3\sqrt{2} & -4\sqrt{2} & . & -8\sqrt{2} & . & . \\ \sqrt{2} & 3\sqrt{2} & . & 3\sqrt{2} & . & 7\sqrt{2} & 8\sqrt{2} & . & 12\sqrt{2} & . & . \\ 1 & 1 & . & 1 & . & 0 & 0 & . & 1 & . & 1 \end{bmatrix}$$

- Note the representation: 3-row matrix
  - Rotation only applies on the “coordinate” rows
  - The value does not change
  - Why is pacman grainy?

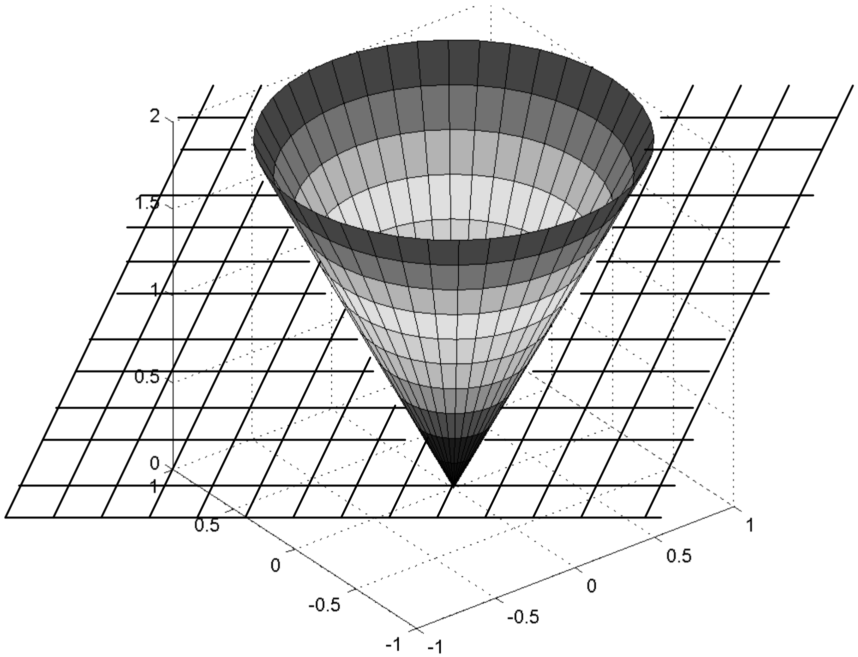
3-D Rotation

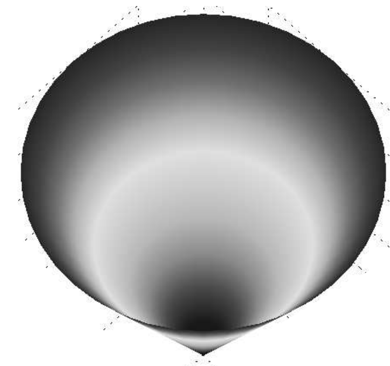




- 2 degrees of freedom
  - 2 separate angles
- What will the rotation matrix be?

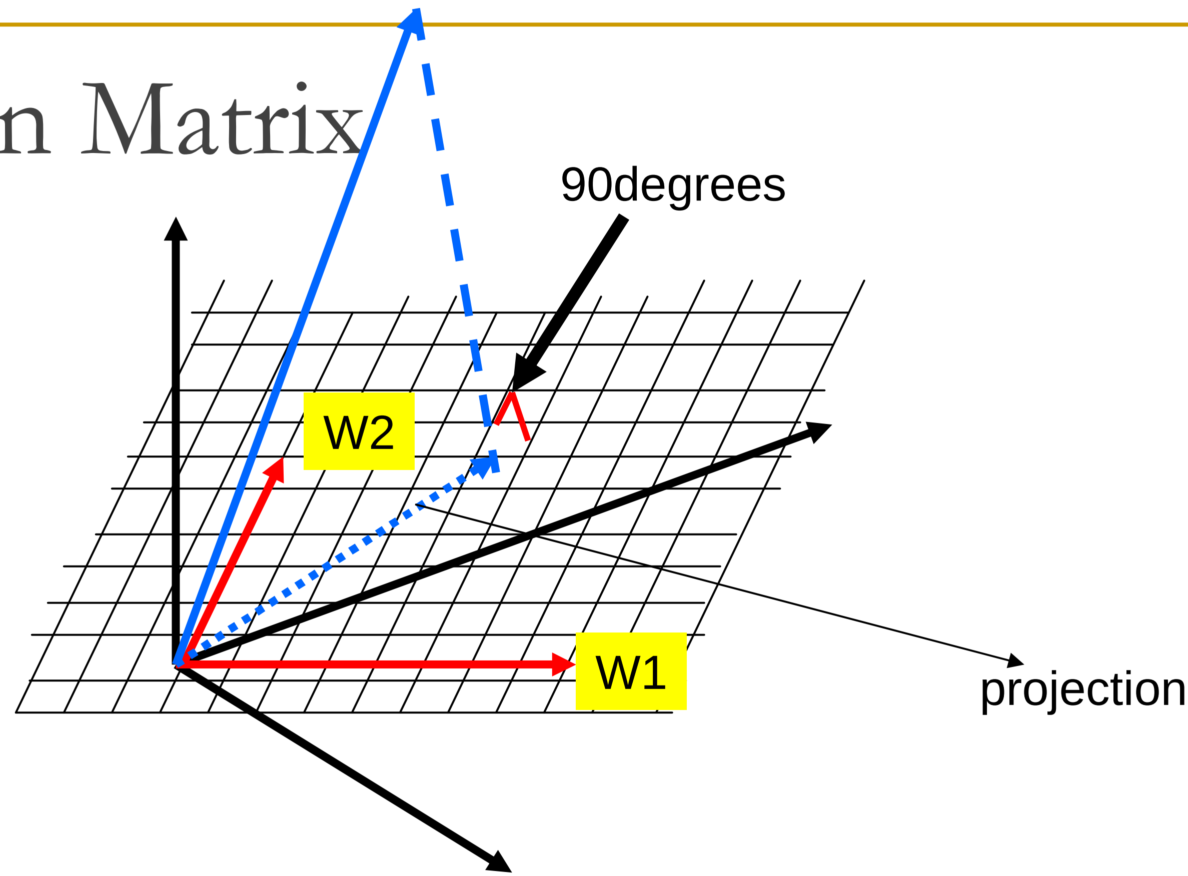
Projections





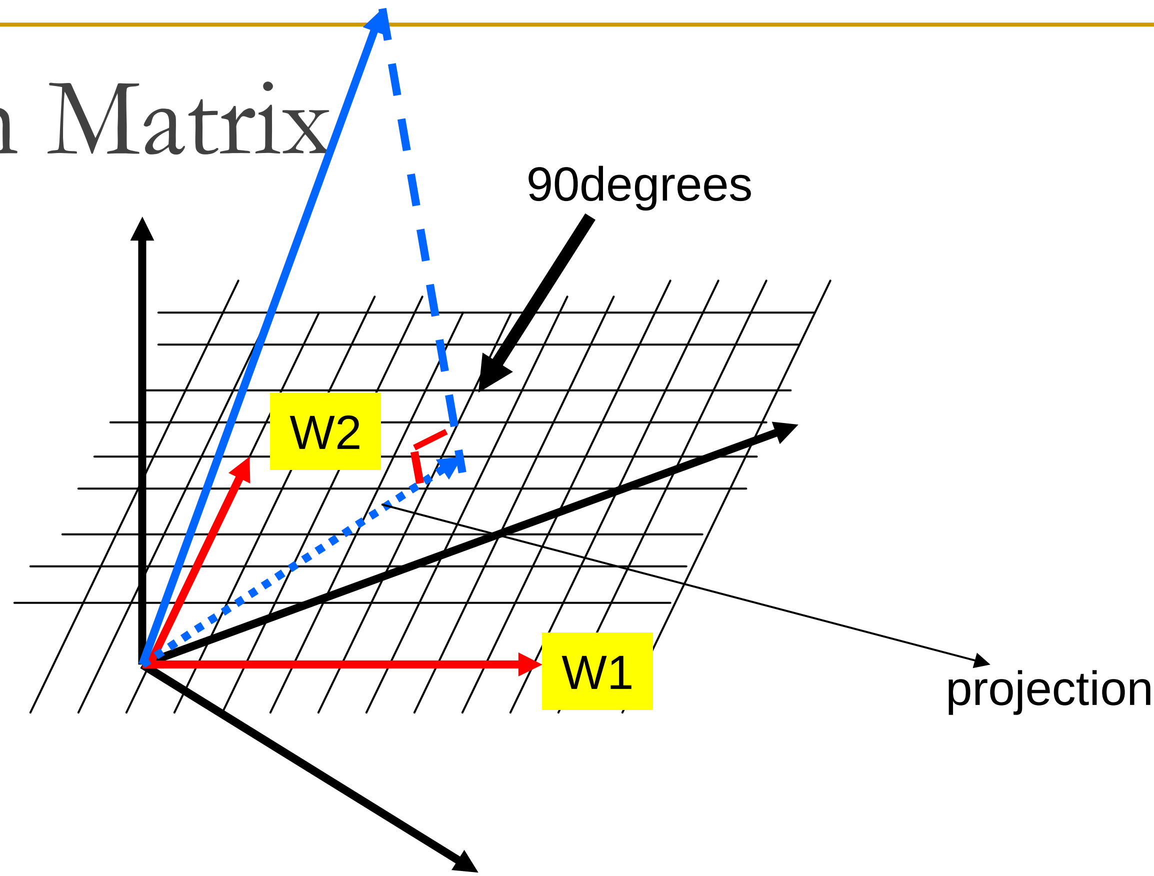
- What would we see if the cone to the left were transparent if we looked at it along the normal to the plane
  - The plane goes through the origin
  - Answer: the figure to the right
- How do we get this? Projection

Projection Matrix



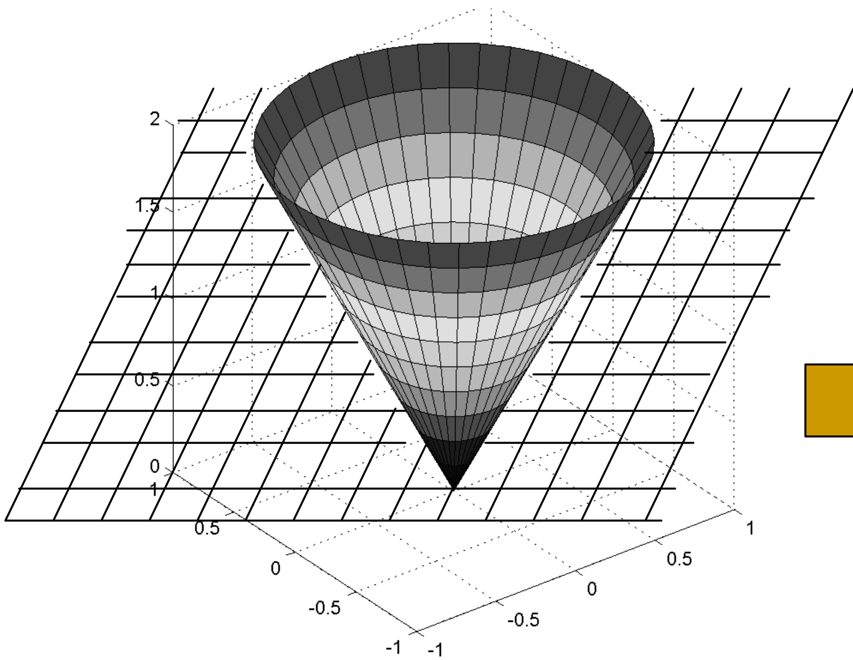
- Consider any plane specified by a set of vectors W_1, W_2..
  - Or matrix [W_1 W_2 ..]
  - Any vector can be projected onto this plane
  - The matrix A that rotates and scales the vector so that it becomes its projection is a projection matrix

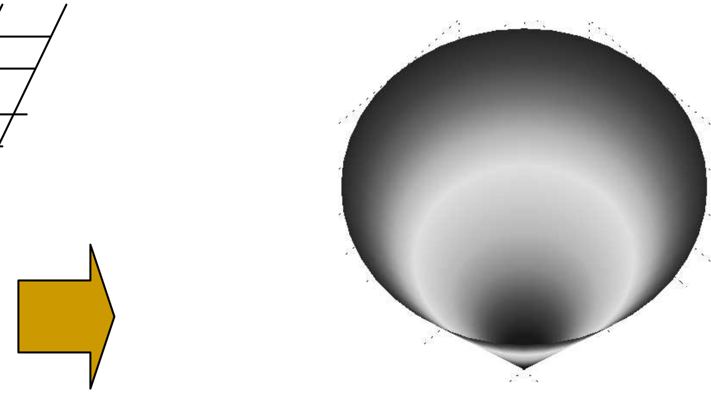
Projection Matrix



- Given a set of vectors W1, W2, which form a matrix W = [W1 W2..]
- The projection matrix that transforms any vector X to its projection on the plane is
  - $P = W (W^T W)^{-1} W^T$
    - We will visit matrix inversion shortly
- Magic – any set of vectors from the same plane that are expressed as a matrix will give you the same projection matrix
  - $P = V (V^T V)^{-1} V^T$

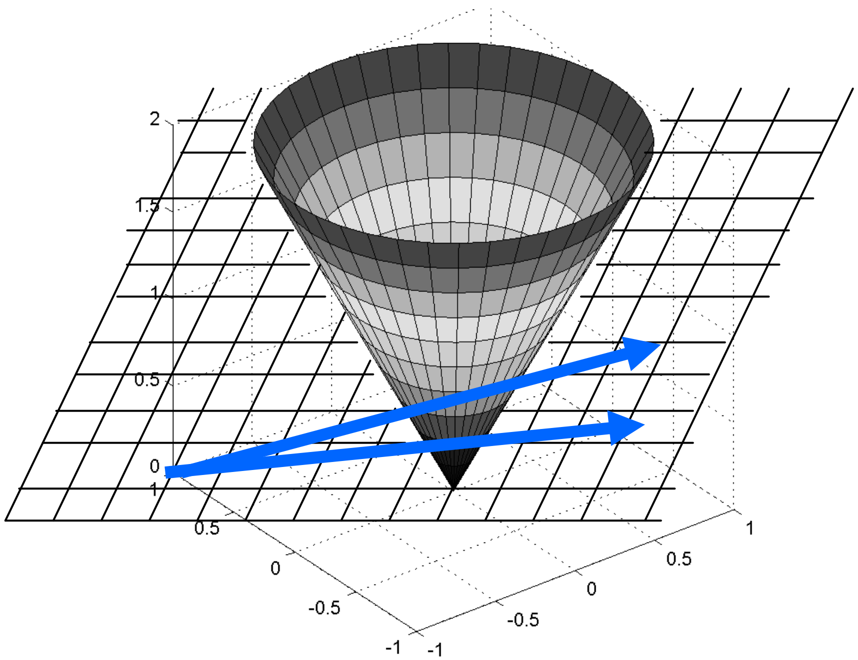
Projections

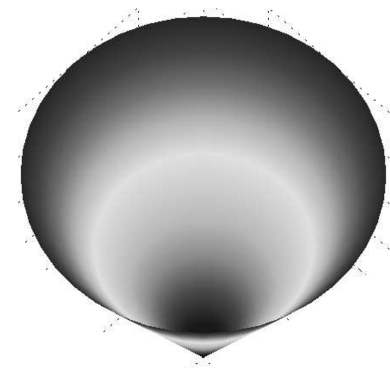




- HOW?

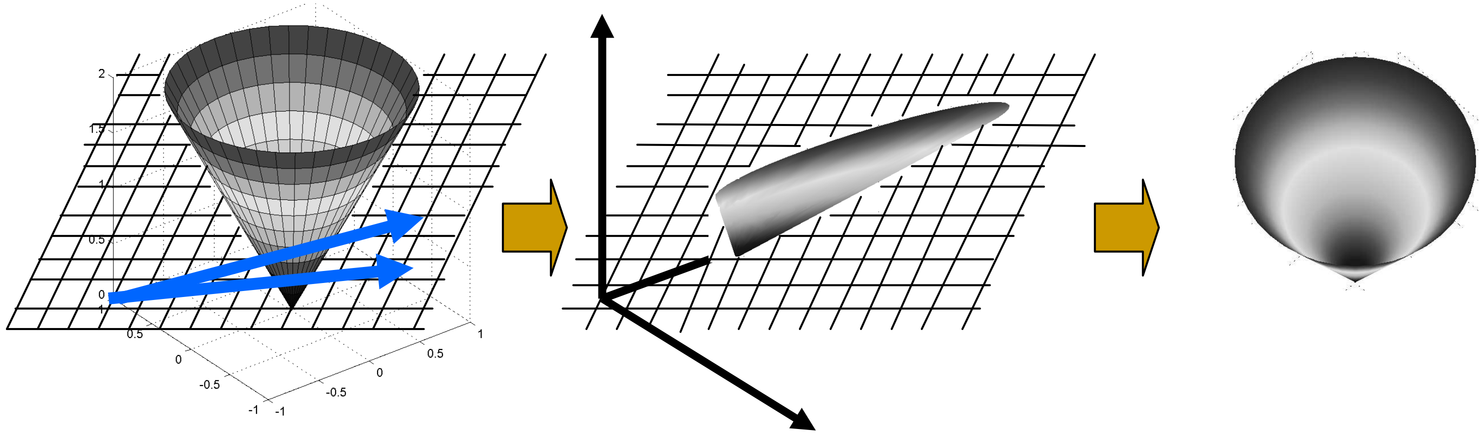
Projections





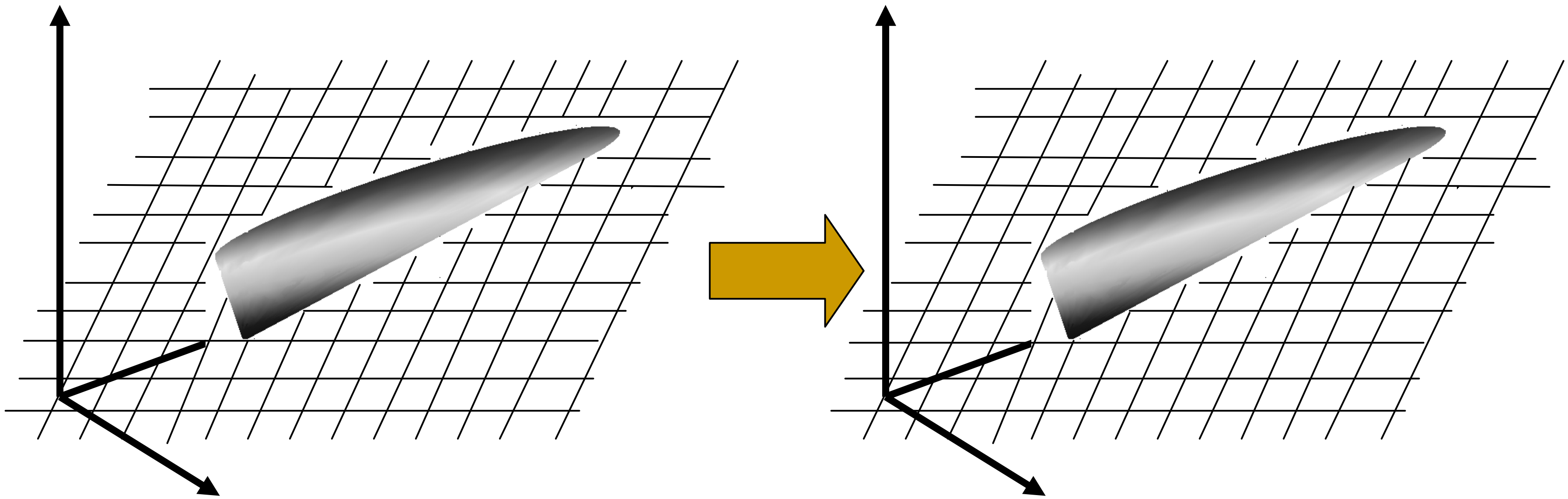
- Draw any two vectors W1 and W2 that lie on the plane
  - ***ANY two*** **so long as they have different angles**
- Compose a matrix W = [W1 W2]
- Compose the projection matrix $P = W (W^T W)^{-1} W^T$
- Multiply every point on the cone by P to get its projection
- View it ☺
  - I'm missing a step here – what is it?

Projections



- The projection actually projects it onto the plane, but you're still seeing the plane in 3D
  - The result of the projection is a 3-D vector
  - $P = W (W^TW)^{-1} W^T = 3x3$, $P*Vector = 3x1$
  - The image must be rotated till the plane is in the plane of the paper
    - The Z axis in this case will always be zero and can be ignored
    - How will you rotate it? (remember you know W1 and W2)

Projection matrix properties



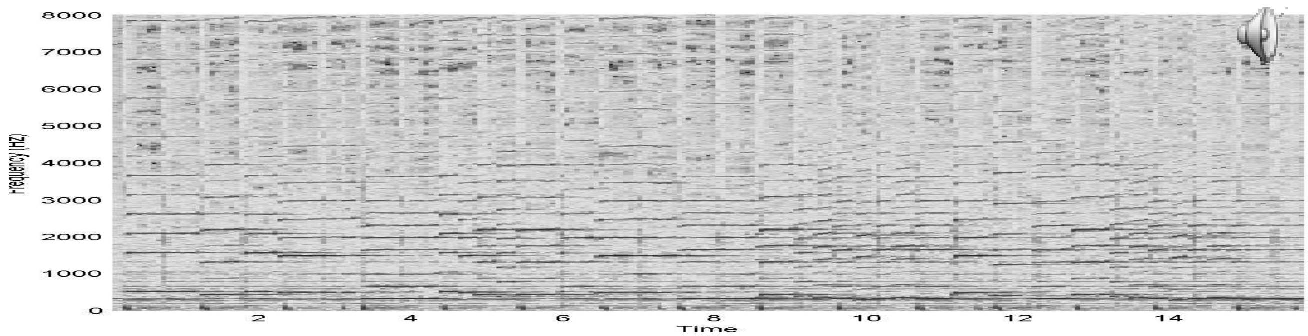
- The projection of any vector that is already on the plane is the vector itself
  - Px = x if x is on the plane
  - If the object is already on the plane, there is no further projection to be performed
- The projection of a projection is the projection
  - P (Px) = Px
  - That is because Px is already on the plane
- Projection matrices are *idempotent*
  - $P^2 = P$
    - Follows from the above

Projections: A more physical meaning

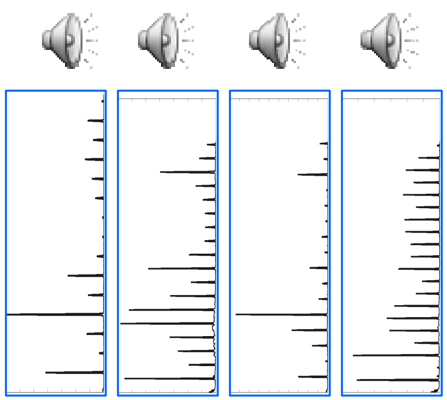
- Let W_1, W_2 .. W_k be “bases”
- We want to explain our data in terms of these “bases”
  - We often cannot do so
  - But we can explain a significant portion of it

- The portion of the data that can be expressed in terms of our vectors W_1, W_2, .. W_k, is the projection of the data on the W_1 .. W_k (hyper) plane
  - In our previous example, the “data” were all the points on a cone
  - The interpretation for volumetric data is obvious

Projection : an example with sounds

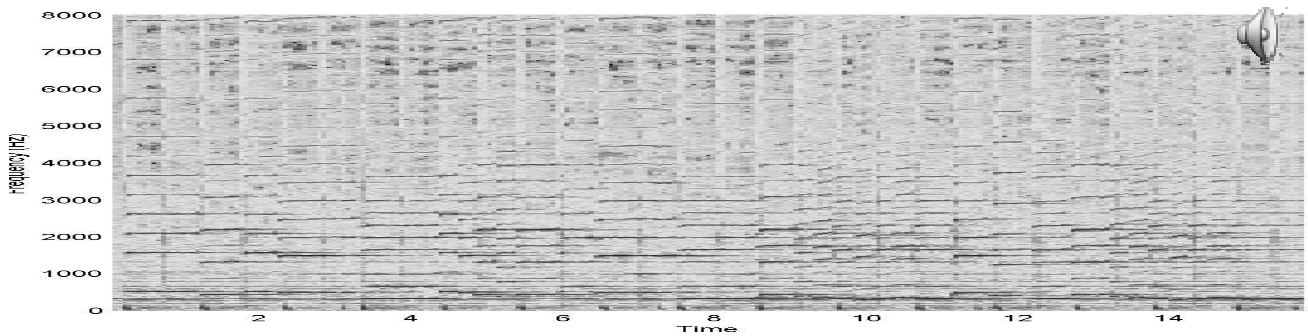


- The spectrogram (matrix) of a piece of music

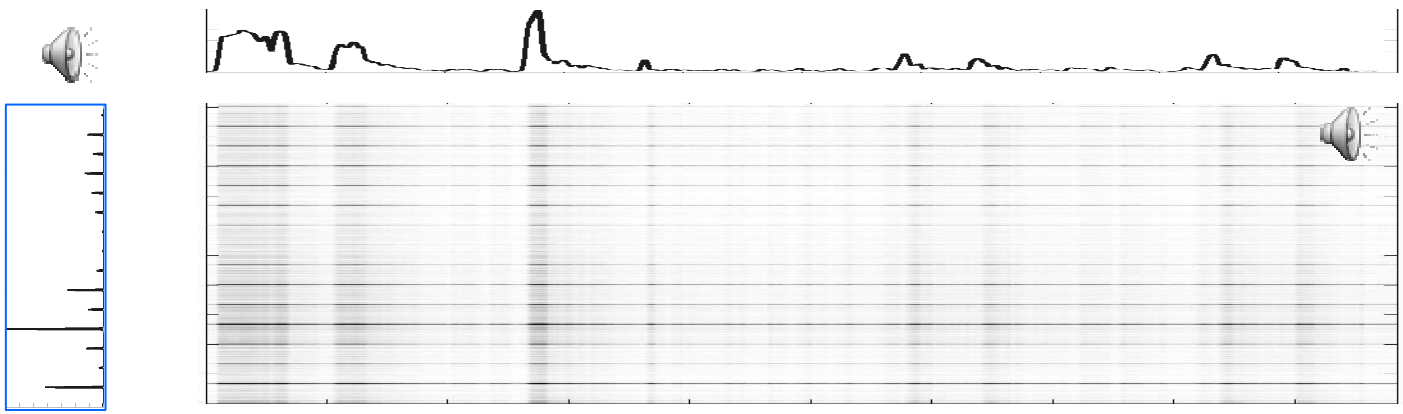


- How much of the above music was composed of the above notes
  - I.e. how much can it be explained by the notes

Projection: one note

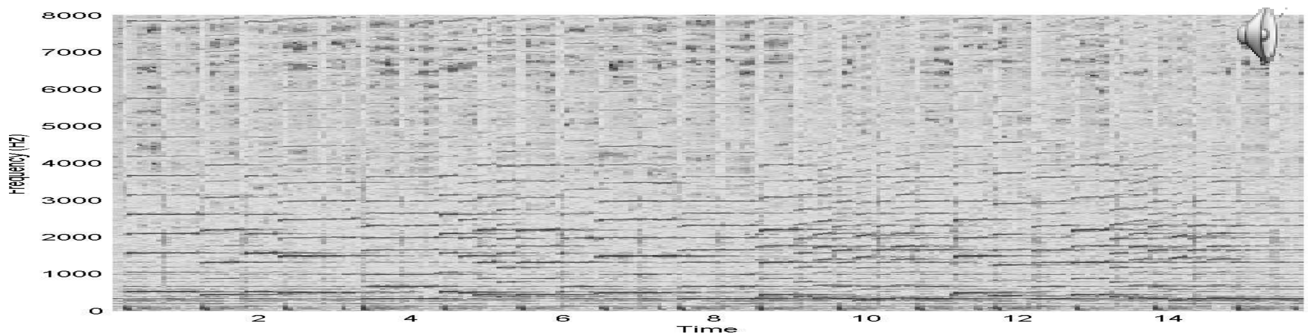
M = 

- The spectrogram (matrix) of a piece of music

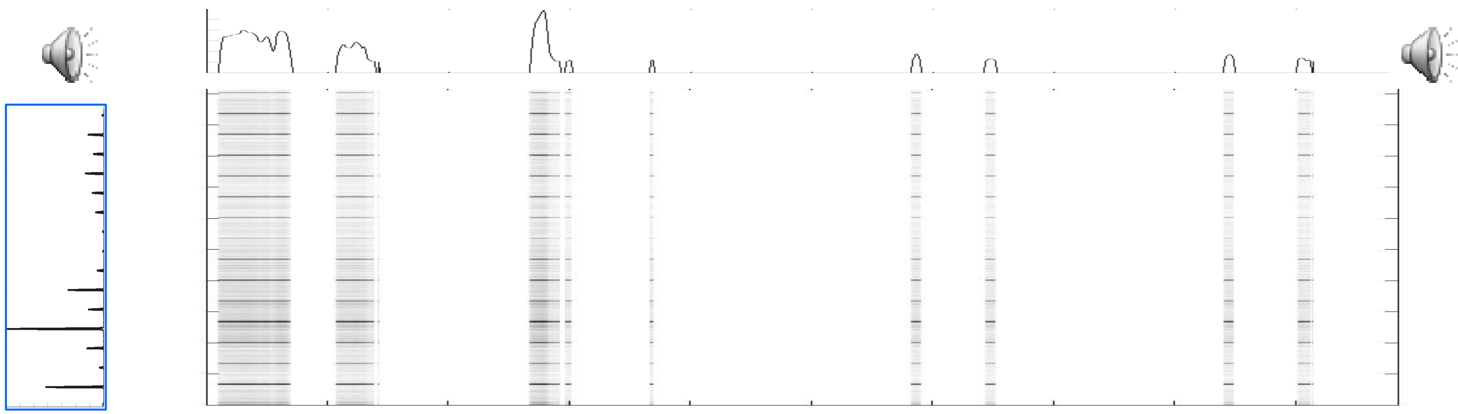
W = 

- M = spectrogram;   W = note
- $P = W (W^T W)^{-1} W^T$
- Projected Spectrogram = P * M

Projection: one note – cleaned up

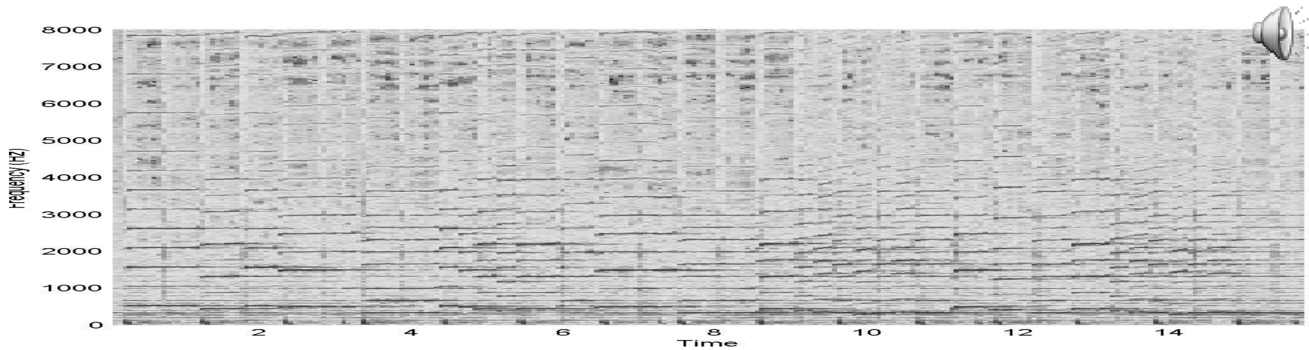
M = 

- The spectrogram (matrix) of a piece of music

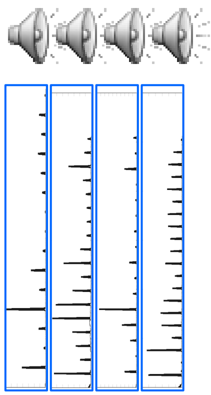
W = 

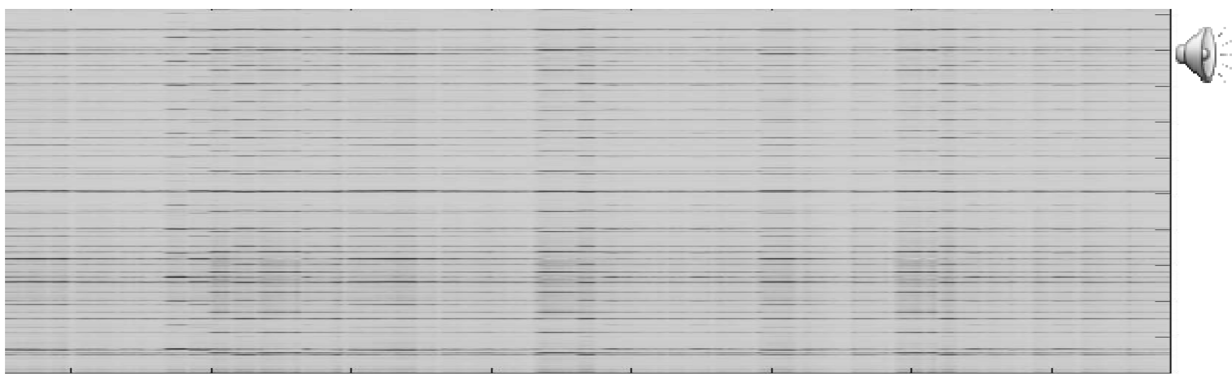
- Floored all matrix values below a threshold to zero

Projection: multiple notes

M = 

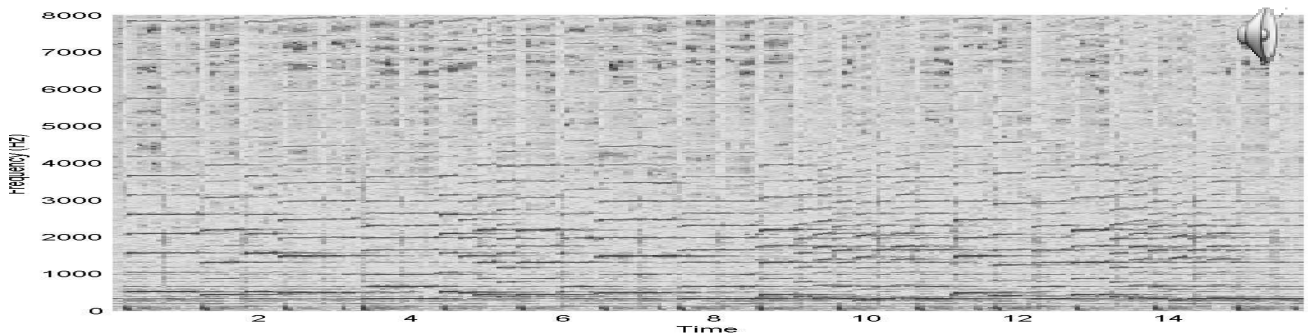
- The spectrogram (matrix) of a piece of music

W = 

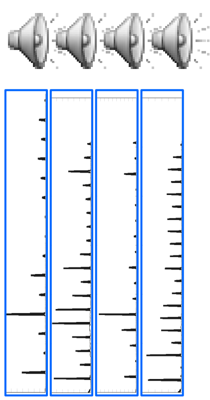


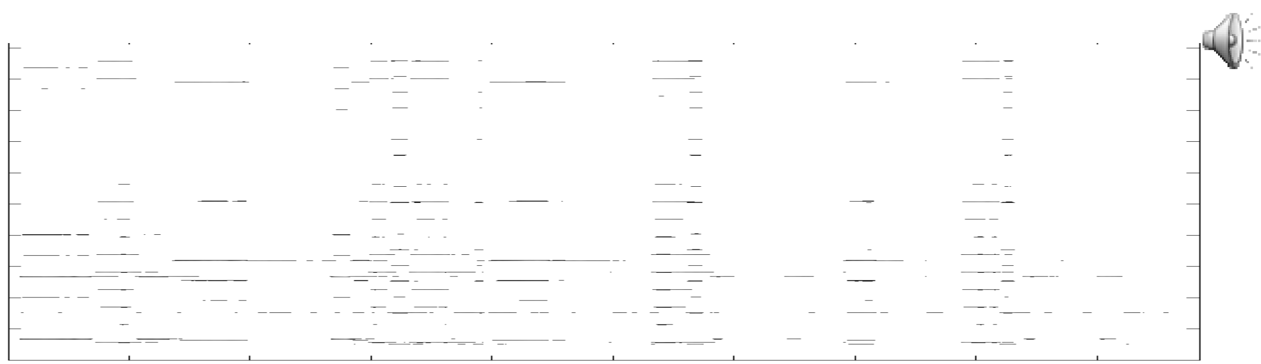
- $P = W (W^T W)^{-1} W^T$
- Projected Spectrogram = P * M

Projection: multiple notes, cleaned up

M = 

- The spectrogram (matrix) of a piece of music

W = 



- $P = W (W^T W)^{-1} W^T$
- Projected Spectrogram = P * M

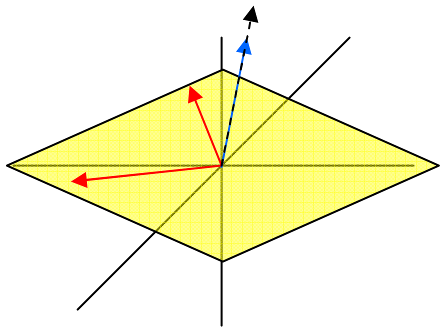
Projection and Least Squares

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

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

  - Error vector E = $V - V_{approx}$
  - Squared error energy for V $e(V) = \text{norm}(E)^2$
  - Total error = sum_over_all_V { e(V) } = Σ_V e(V)
- Projection computes V_{approx} for all vectors such that Total error is minimized
  - It does not give you "a", "b", "c".. Though
    - That needs a different operation – the inverse / pseudo inverse

Orthogonal and Orthonormal matrices



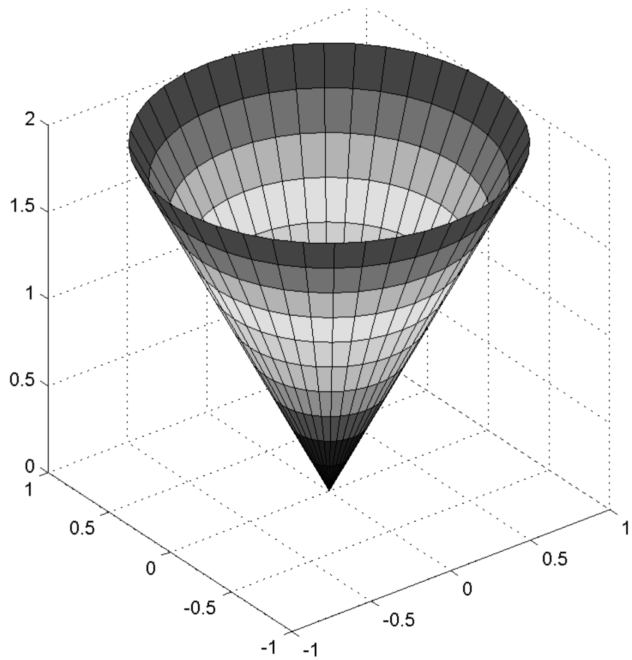
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0.707 & -0.354 & 0.612 \\ 0.707 & 0.354 & -0.612 \\ 0 & 0.866 & 0.5 \end{bmatrix}$$

- Orthogonal Matrix : AA^T = diagonal
  - Each row vector lies exactly along the normal to the plane specified by the rest of the vectors in the matrix
- Orthonormal Matrix: $AA^T = A^TA = I$
  - In additional to be orthogonal, each vector has length exactly = 1.0
  - Interesting observation: In a square matrix if the length of the row vectors is 1.0, the length of the column vectors is also 1.0

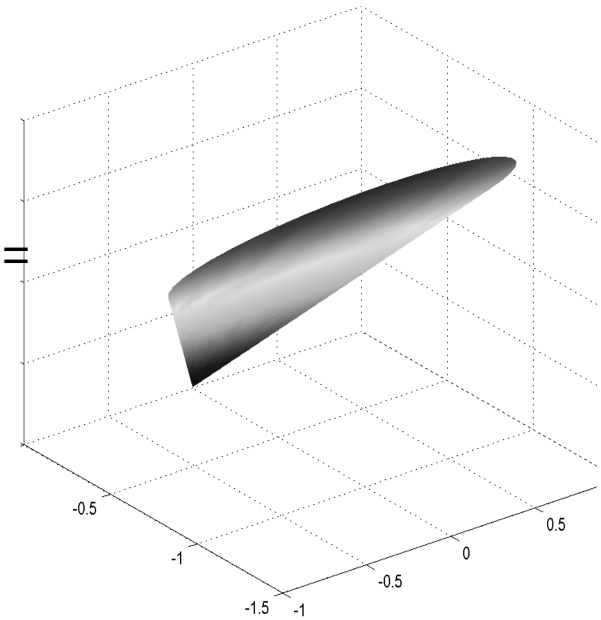
Orthogonal and Orthonormal Matrices

- Orthonormal matrices will retain the relative angles between transformed vectors
  - Essentially, they are combinations of rotations, reflections and permutations
  - Rotation matrices and permutation matrices are all orthonormal matrices
  - The vectors in an orthonormal matrix are at 90degrees to one another.
- Orthogonal matrices are like Orthonormal matrices with stretching
  - The product of a diagonal matrix and an orthonormal matrix

Matrix Rank and Rank-Deficient Matrices



P * Cone =

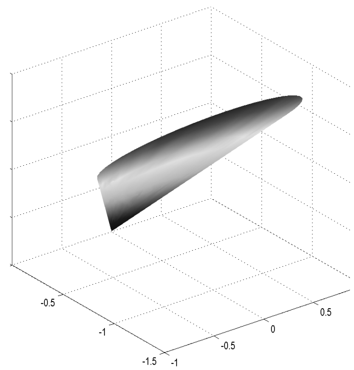


- Some matrices will eliminate one or more dimensions during transformation
  - These are *rank deficient* matrices
  - The rank of the matrix is the dimensionality of the trasnsformed version of a full-dimensional object

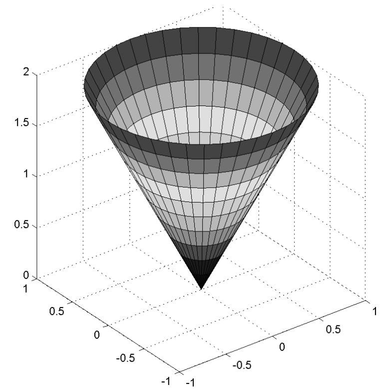
Matrix Rank and Rank-Deficient Matrices

P =

$$\begin{matrix} 1.0000 & 0 & 0 \\ 0 & 0.2500 & -0.4330 \\ 0 & -0.4330 & 0.7500 \end{matrix}$$

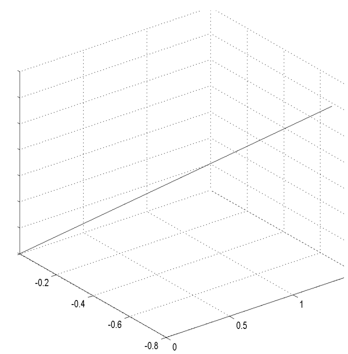


Rank = 2



P2 =

$$\begin{matrix} 0.5000 & -0.2500 & 0.4330 \\ -0.2500 & 0.1250 & -0.2165 \\ 0.4330 & -0.2165 & 0.3750 \end{matrix}$$

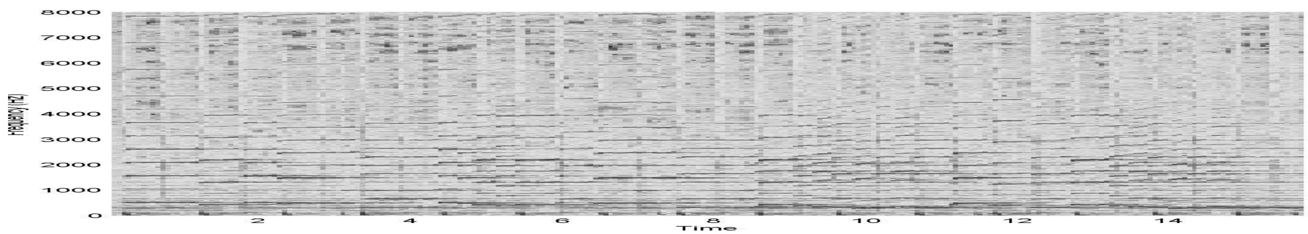


Rank = 1

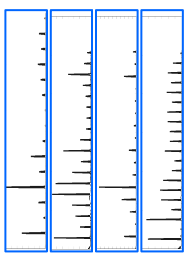
- Some matrices will eliminate one or more dimensions during transformation
  - These are *rank deficient* matrices
  - The rank of the matrix is the dimensionality of the transformed version of a full-dimensional object

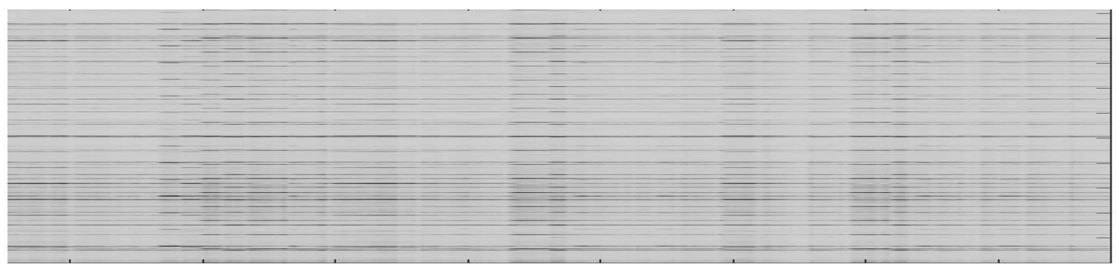
Projections are often examples of rank-deficient transforms

M =



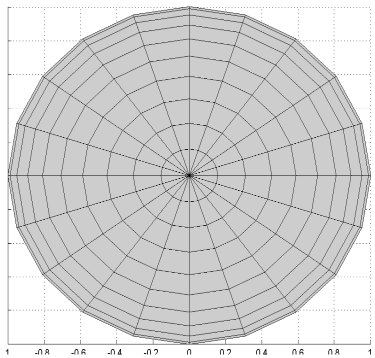
W =

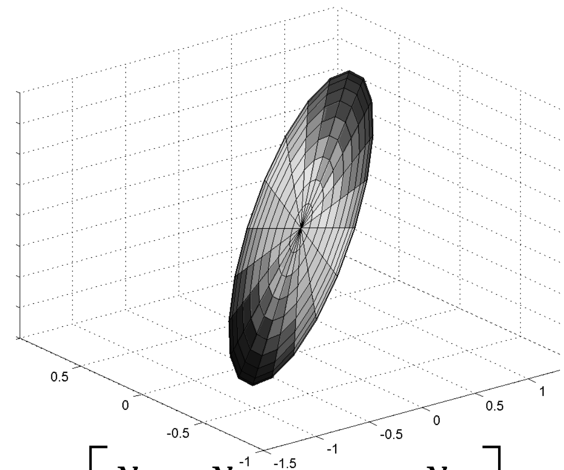




- $P = W (W^TW)^{-1} W^T$; Projected Spectrogram = P * M
- The original spectrogram can never be recovered
  - P is rank deficient
- P explains all vectors in the new spectrogram as a mixture of only the 4 vectors in W
  - There are only 4 ***independent*** bases
  - Rank of P is 4

Non-square Matrices



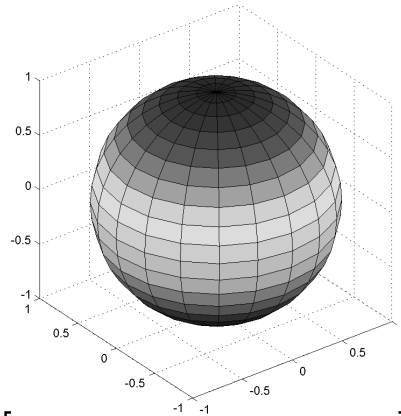


$$X = \begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \end{bmatrix} \qquad P = \begin{bmatrix} .8 & .9 \\ .1 & .9 \\ .6 & 0 \end{bmatrix} \qquad PX = \begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \\ z_1 & z_2 & . & . & z_N \end{bmatrix}$$

X = 2D data P = transform 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

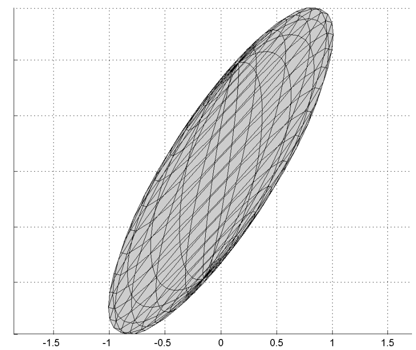


$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \\ z_1 & z_2 & . & . & z_N \end{bmatrix}$$

X = 3D data, rank 3

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

P = transform

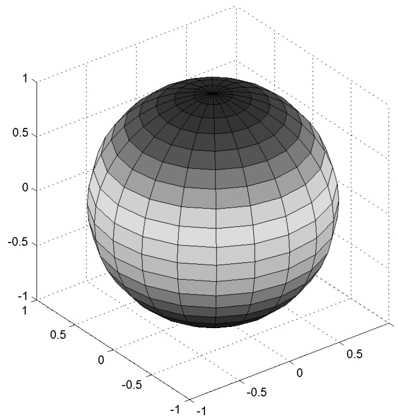


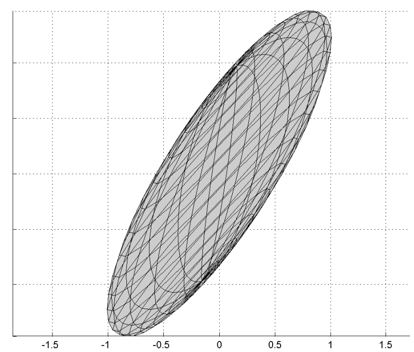
$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \end{bmatrix}$$

PX = 2D, rank 2

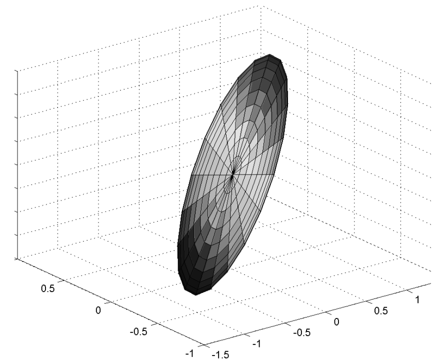
- Non-square matrices add or subtract axes
  - Fewer rows than columns → reduce axes
    - May reduce dimensionality of the data

The Rank of a Matrix





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

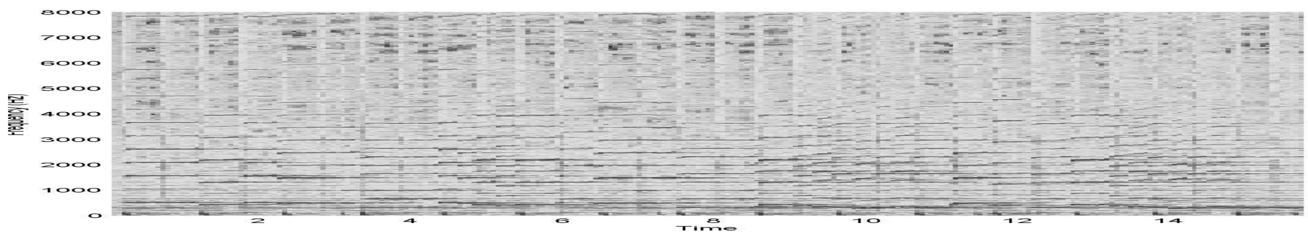


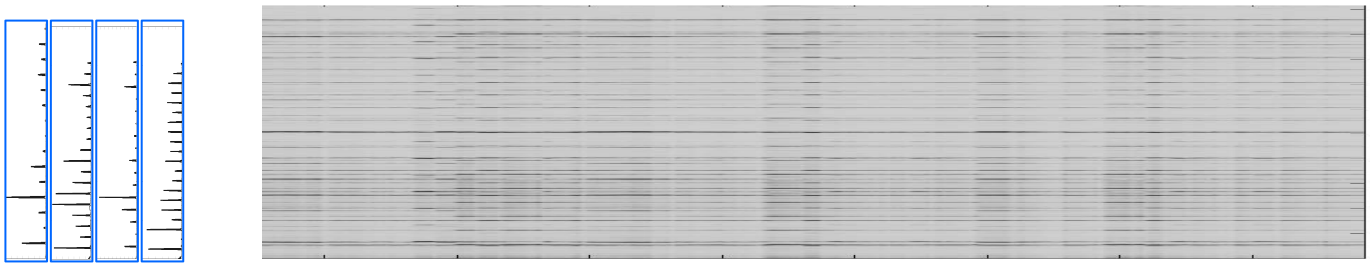
$$\begin{bmatrix} .8 & .9 \\ .1 & .9 \\ .6 & 0 \end{bmatrix}$$

- 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

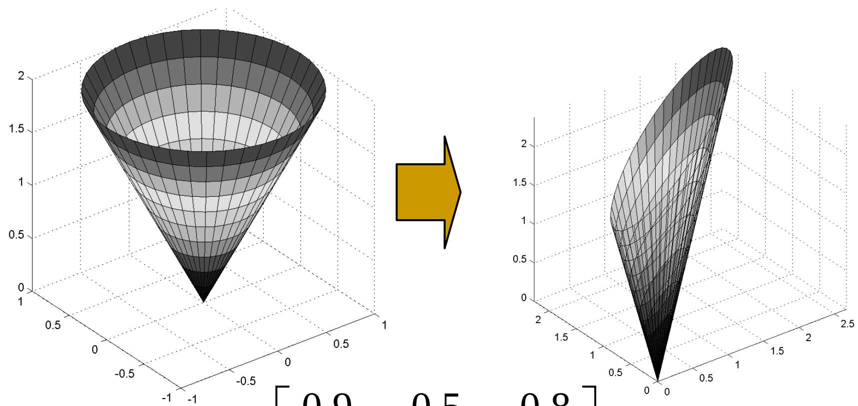
M =



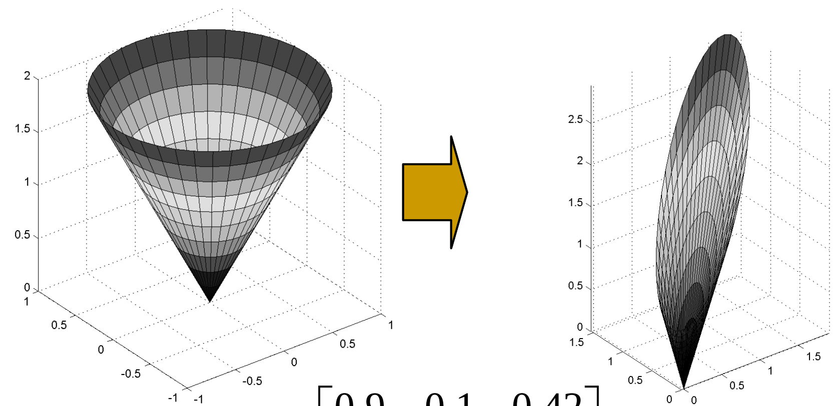


- Projected Spectrogram = P * M
  - Every vector in it is a combination of only 4 bases
- The rank of the matrix is the *smallest* no. of bases required to describe the output
  - E.g. if note no. 4 in P could be expressed as a combination of notes 1,2 and 3, it provides no additional information
  -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

Matrix rank is unchanged by transposition



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

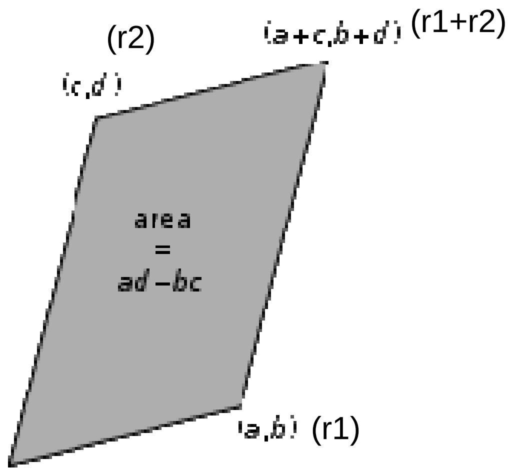


$$\begin{bmatrix} 0.9 & 0.1 & 0.42 \\ 0.5 & 0.4 & 0.44 \\ 0.8 & 0.9 & 0.86 \end{bmatrix}$$

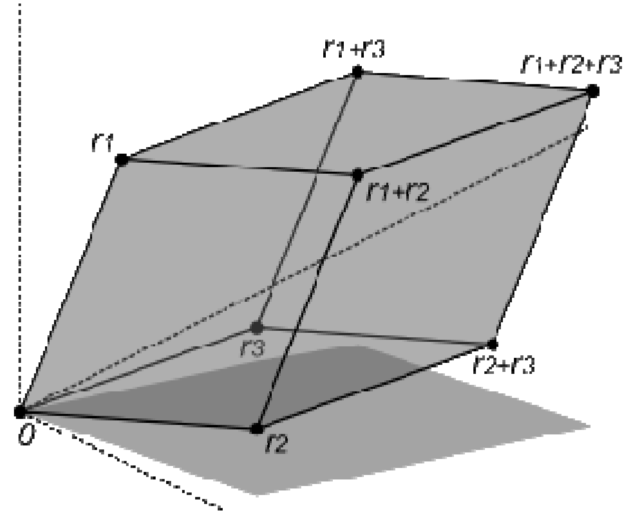
- If an N-D object is compressed to a K-D object by a matrix, it will also be compressed to a K-D object by the transpose of the matrix

Matrix Determinant

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



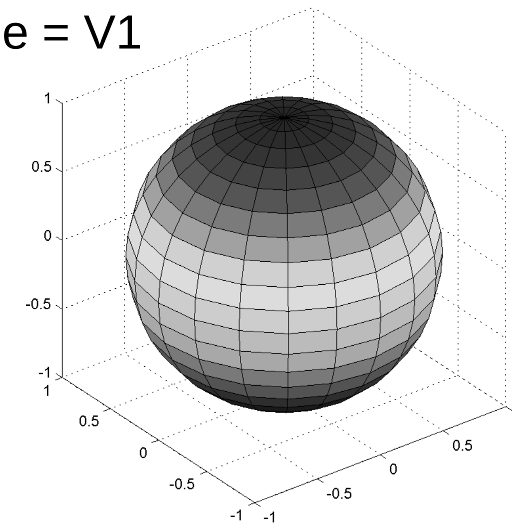
$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$



- 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

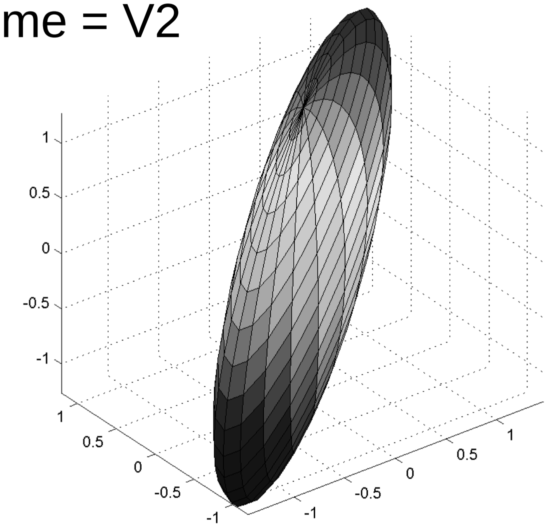
Matrix Determinant: Another Perspective

Volume = V1



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

Volume = V2



- 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-D objects into zero-volume N-D 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-D 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

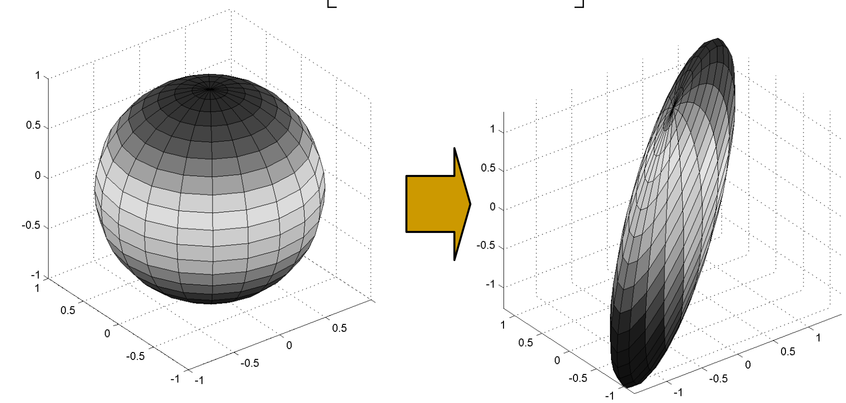
- Associative for square matrices $$|\mathbf{A}\cdot\mathbf{B}\cdot\mathbf{C}| = |\mathbf{A}|\cdot|\mathbf{B}|\cdot|\mathbf{C}|$$
  - Scaling volume sequentially by several matrices is equal to scaling once by the product of the matrices
- Volume of sum != sum of Volumes $$|(\mathbf{B}+\mathbf{C})| \neq |\mathbf{B}| + |\mathbf{C}|$$
  - The volume of the parallelepiped formed by row vectors of the sum of two matrices is not the sum of the volumes of the parallelepipeds formed by the original matrices
- Commutative for square matrices!!!

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

  - The order in which you scale the volume of an object is irrelevant

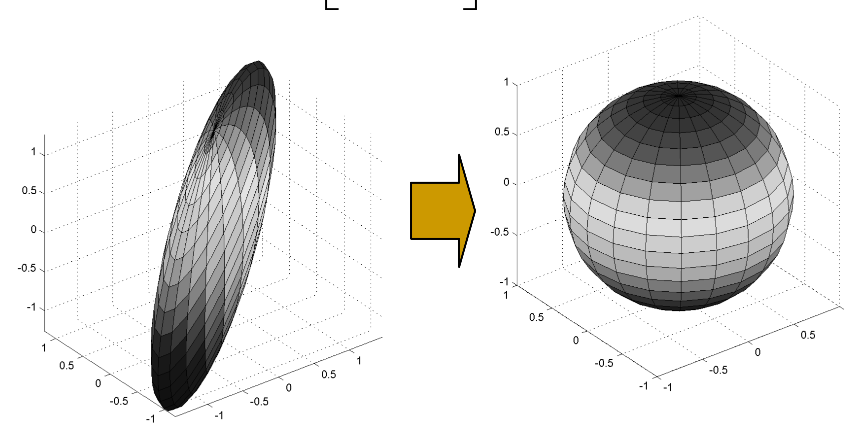
Matrix Inversion

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

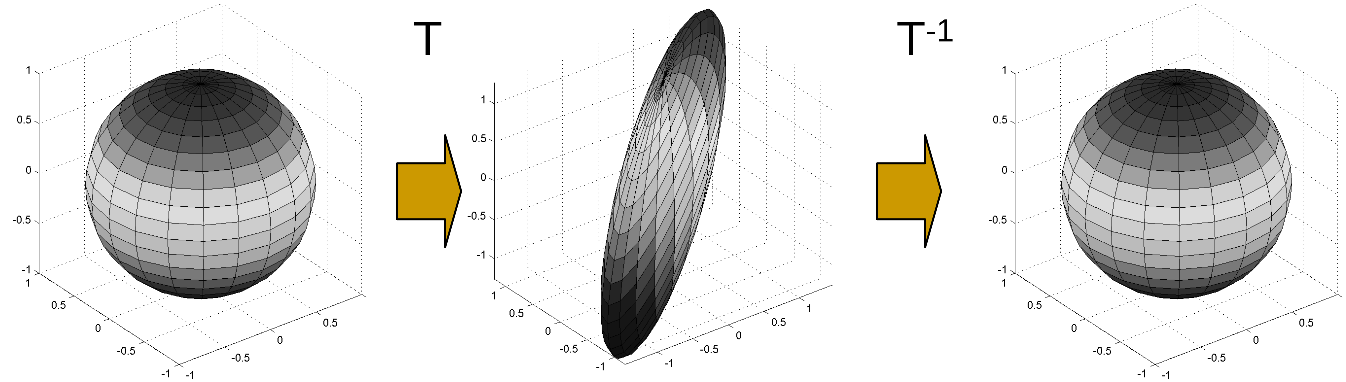


- A matrix transforms an N-D object to a different N-D object
- What transforms the new object back to the original?
  - The *inverse transformation*
- The inverse transformation is called the matrix inverse

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



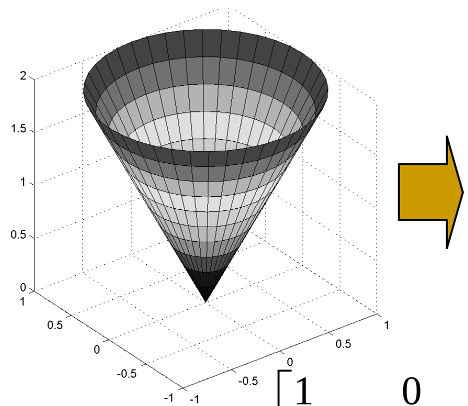
Matrix Inversion

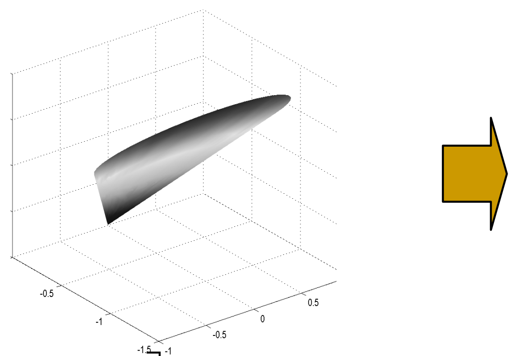


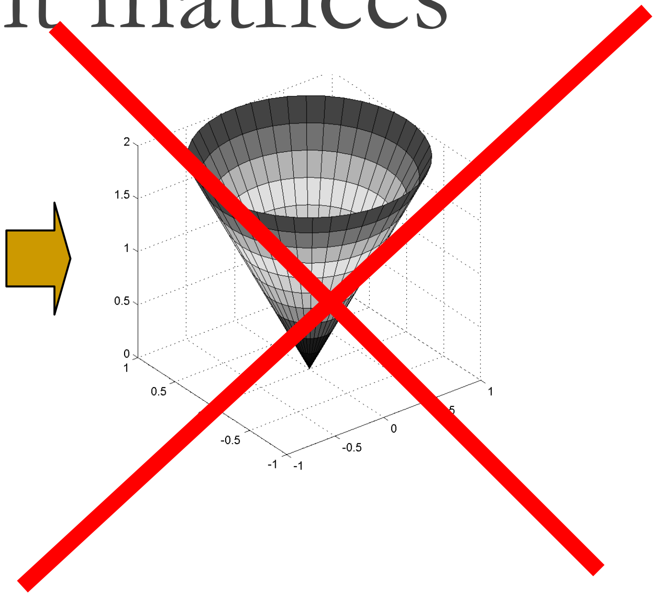
$T^{-1}T = I$

- 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

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 $e(V) = \text{norm}(E)^2$
  - Total error = Total error + e(V)
- Projection computes V_{approx} for all vectors such that Total error is minimized
- **But WHAT ARE "a" "b" and "c"?**

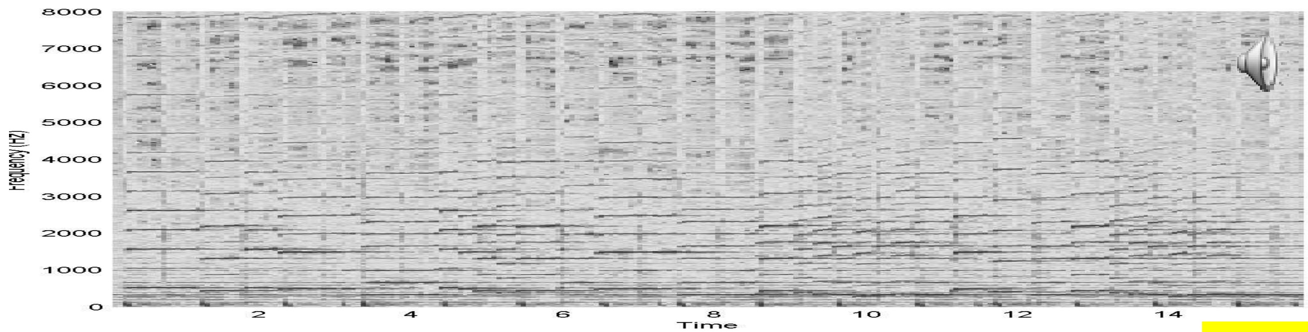
The Pseudo Inverse (PINV)

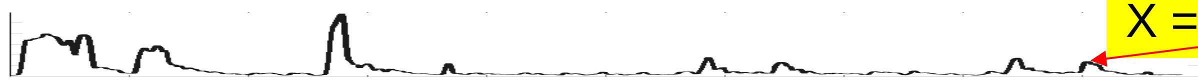
$$V_{approx} = T\begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow V \approx T\begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow \boxed{\begin{bmatrix} a \\ b \\ c \end{bmatrix} = PINV(T) * V}$$

- 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

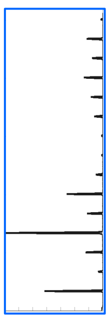
M =

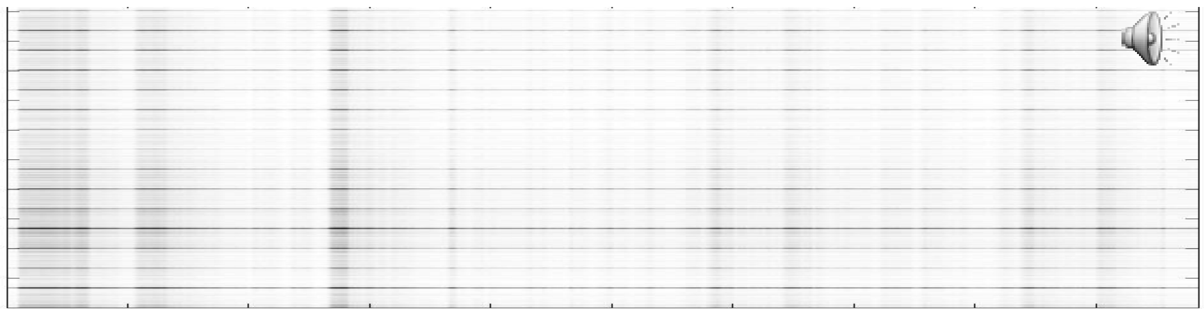




X =PINV(W)*M

W =



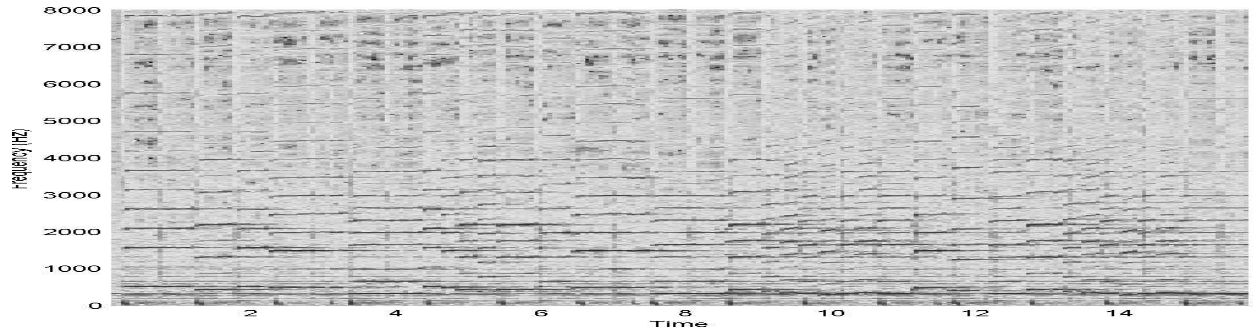


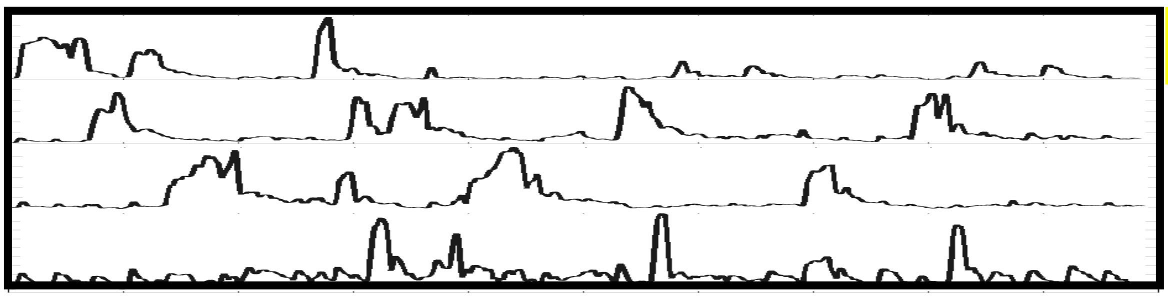
- Recap: $P = W (W^T W)^{-1} W^T$, Projected Spectrogram = P*M
- **Approximation: M = W*X**
- The amount of W in each vector = X = PINV(W)*M
- W*Pinv(W)*M = Projected Spectrogram
  - W*Pinv(W) = Projection matrix!!

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

Explanation with multiple notes

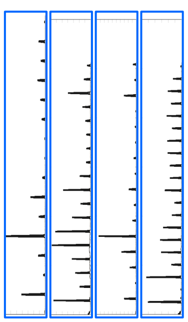
M =

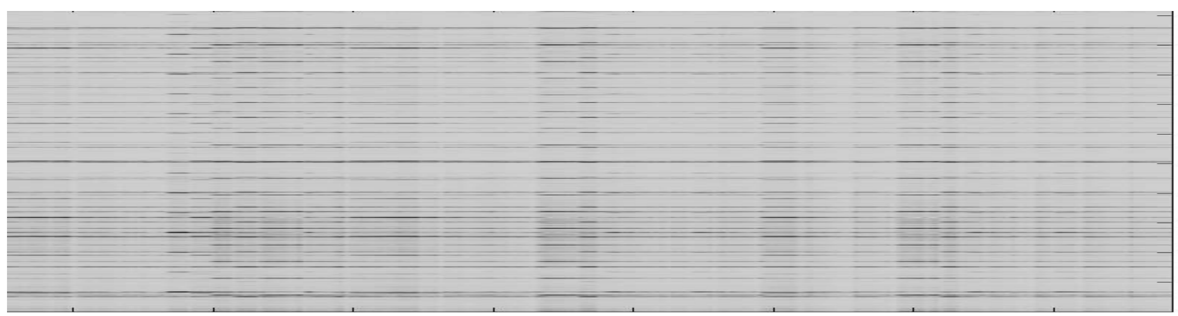




X=PINV(W)*M

W =

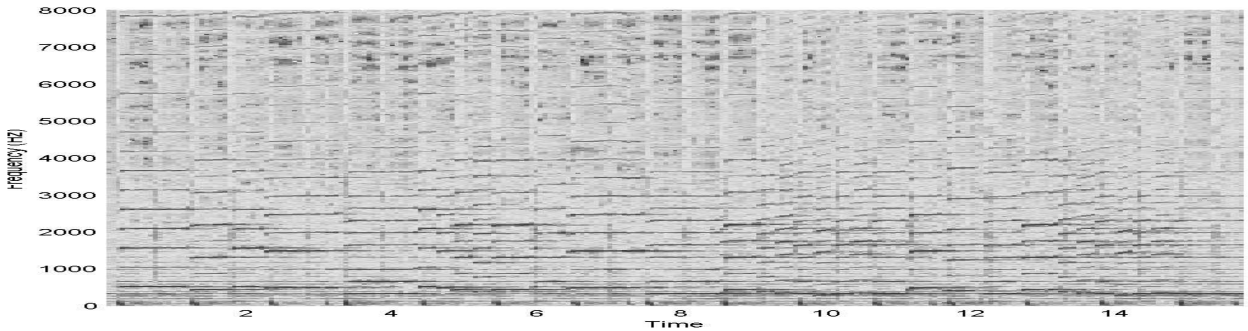




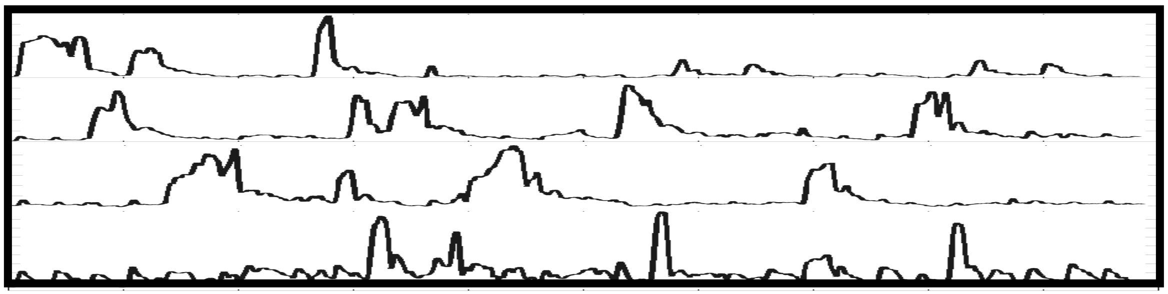
- X = Pinv(W) * M; Projected matrix = W*X = W*Pinv(W)*M

How about the other way?

M =



V =



W = ?

U = ?

- WV \approx M   W = M * Pinv(V)   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)*A = I in this case

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
- Pay attention to multiplication side!

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

- Matrix inverses defined for square matrices only
  - If matrix not square use a matrix pseudoinverse:

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

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

What is the Matrix ?

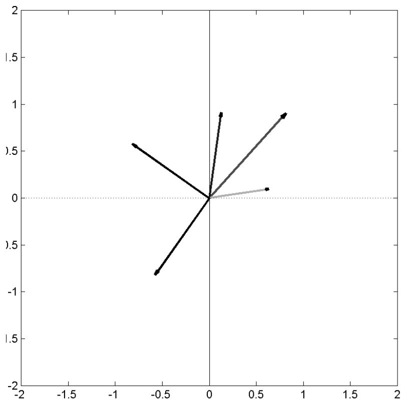
- Duality in terms of the matrix identity
  - Can be a container of data
    - An image, a set of vectors, a table, etc …
  - Can be a <u>linear</u> transformation
    - A process by which to transform data in another matrix
- We'll usually start with the first definition and then apply the second one on it
  - Very frequent operation
  - Room reverberations, mirror reflections, etc …
- Most of signal processing and machine learning are a matrix multiplication!

Eigenanalysis

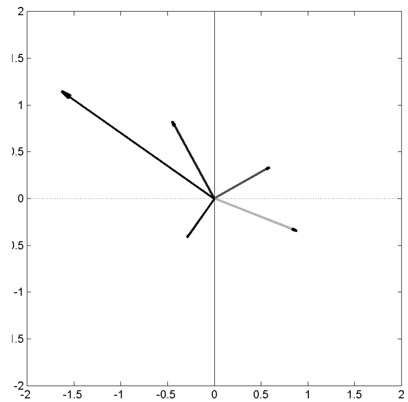
- If something can go through a process mostly unscathed in character it is an *eigen*-something
  - Sound example:
- A vector that can undergo a matrix multiplication and keep pointing the same way is an *eigenvector*
  - Its length can change though
- How much its length changes is expressed by its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

EigenVectors and EigenValues

Black vectors are eigen vectors



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



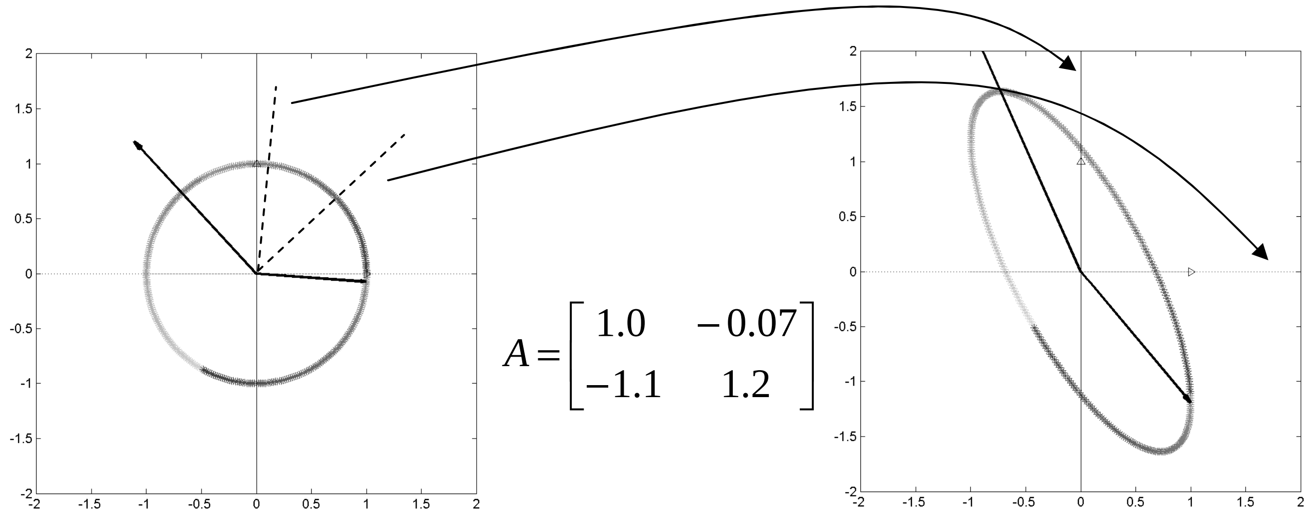
- Vectors that do not change angle upon transformation
  - They may change length

$$MV = \lambda V$$

  - V = eigen vector
  - λ = eigen value
  - `Matlab:  [V, L] = eig(M)`
    - L is a diagonal matrix whose entries are the eigen values
    - V is a maxtrix whose columns are the eigen vectors

Eigen vector example

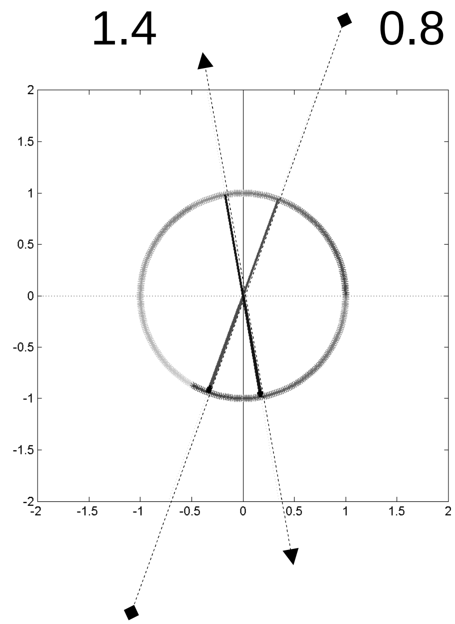
Matrix multiplication revisited



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

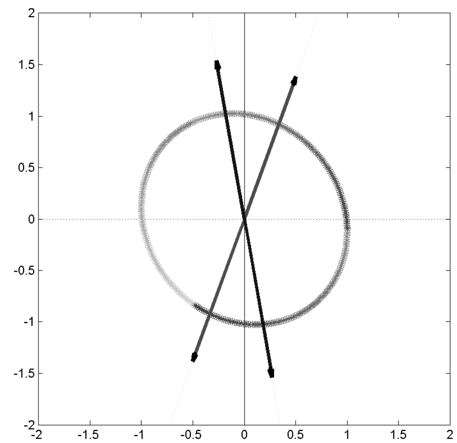
- Matrix transformation “transforms” the space
  - Warps the paper so that the normals to the two vectors now lie along the axes

A stretching operation



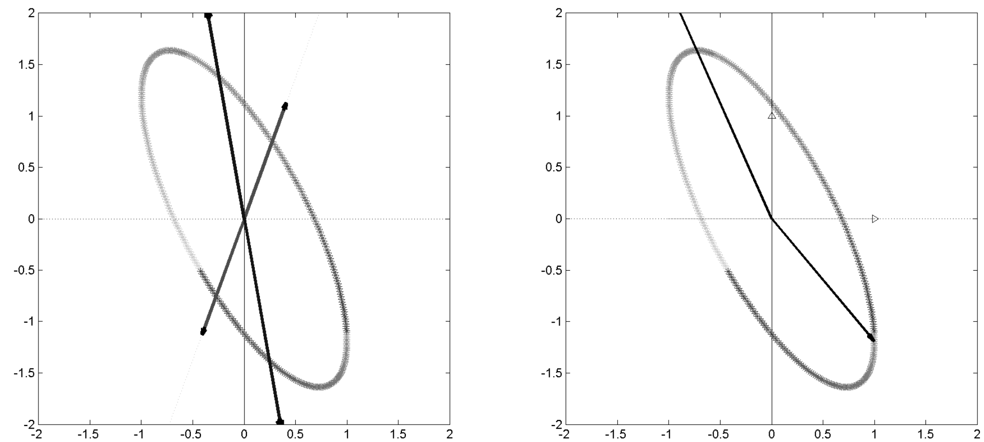
- Draw two lines
- Stretch / shrink the paper along these lines by factors d_1 and d_2
  - The factors could be negative – implies flipping the paper
- The result is a transformation of the space

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



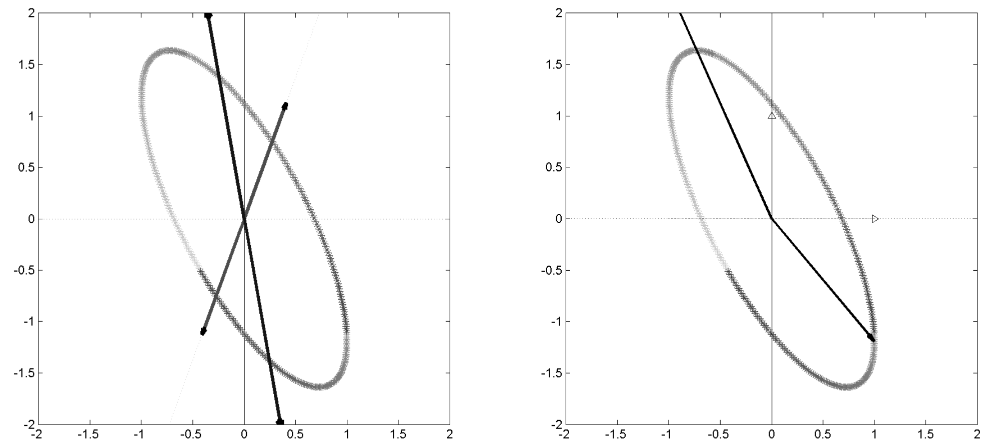
- The result of the stretching is exactly the same as transformation by a matrix
- The axes of stretching/shrinking are the eigenvectors
  - The degree of stretching/shrinking are the corresponding eigenvalues
- The EigenVectors and EigenValues convey all the information about the matrix

Physical interpretation of eigen vector

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

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

$$M = VLV^{-1}$$



- 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

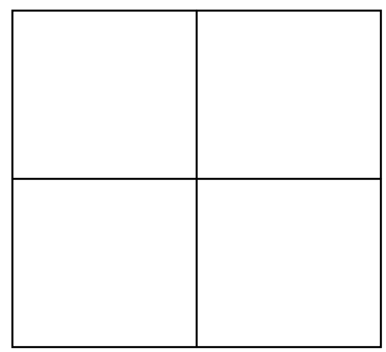
Eigen Analysis

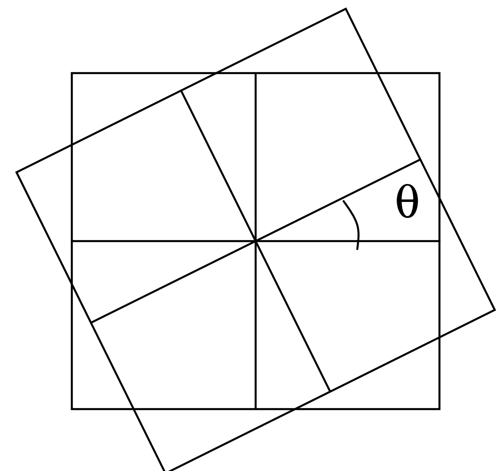
- Not all square matrices have nice eigen values and vectors
  - E.g. consider a rotation matrix

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

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

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

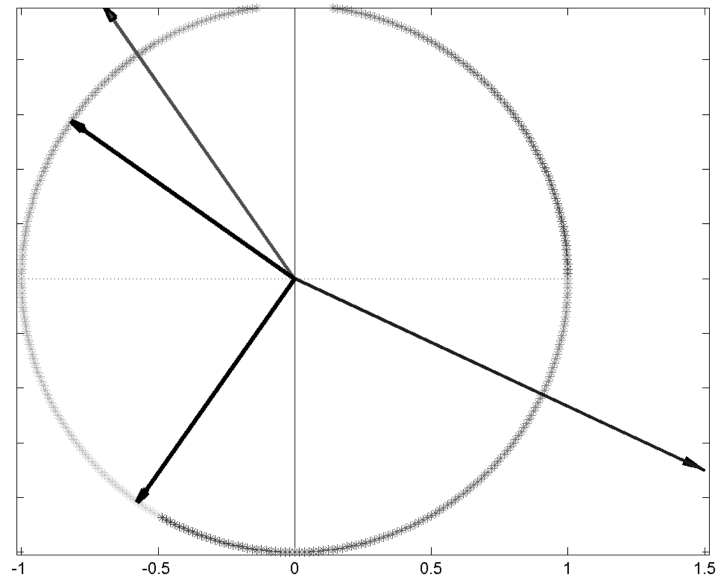




  - This rotates every vector in the plane
    - No vector that remains unchanged
- In these cases the Eigen vectors and values are complex
- Some matrices are special however..

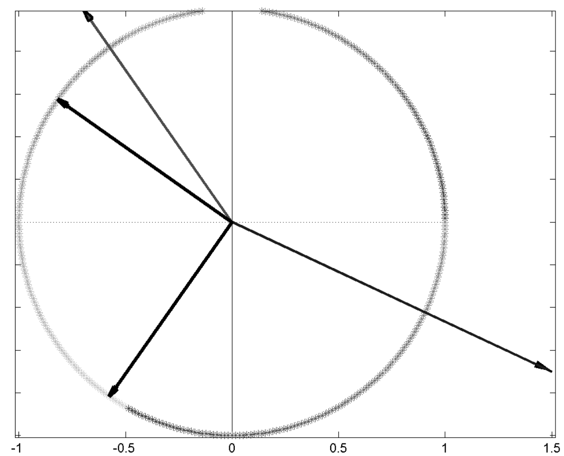
Symmetric Matrices

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

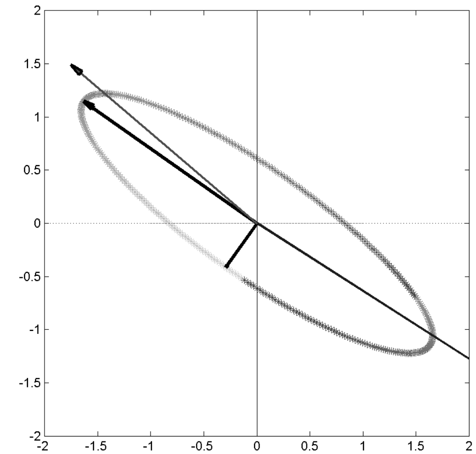


- Matrices that do not change on transposition
  - Row and column vectors are identical
- Symmetric matrix: Eigen vectors and Eigen values are always real
- Eigen vectors are always orthogonal
  - At 90 degrees to one another

Symmetric Matrices



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

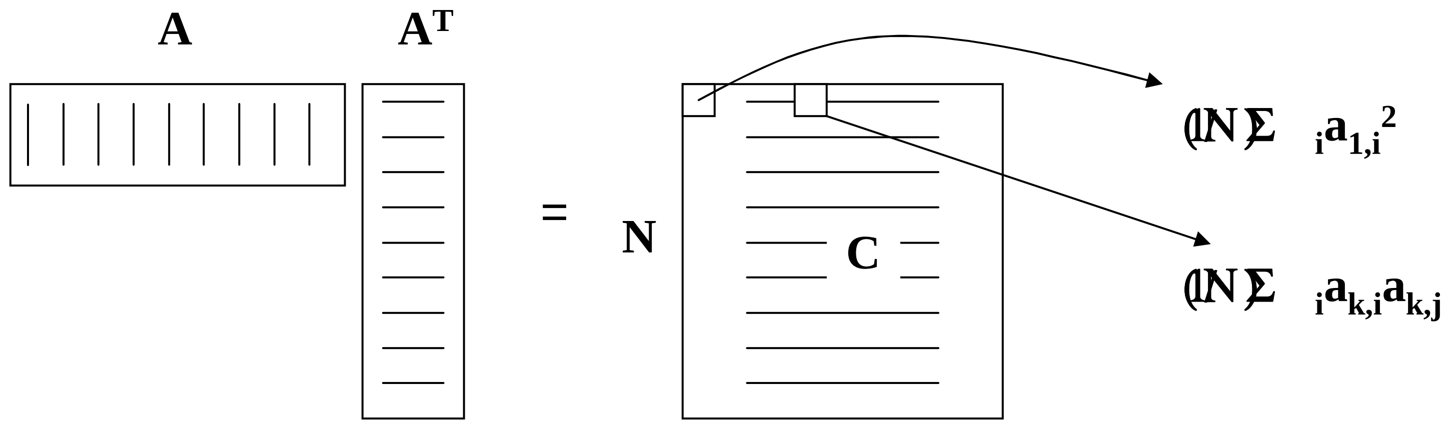


- Eigen vectors point in the direction of the major and minor axes of the ellipsoid resulting from the transformation of a spheroid
  - The eigen values are the lengths of the axes

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
  - $V^T = V^{-1}$
  - $V^T V = I$
  - $V V^T = I$
- $C V_i = \lambda V_i$
- In matrix form : $C V = V L$
  - L is a diagonal matrix with all eigen values
- $C = V L V^T$

The Correlation and Covariance Matrices



- Consider a set of column vectors represented as a DxN matrix M
- The correlation matrix is
  - C = (1/N) MM^T
    - If the average value (mean) of the vectors in M is 0, C is called the ***covariance*** matrix
    - **covariance = correlation + mean * mean^T**
- Diagonal elements represent average value 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

Correlation / Covariance Matrix

$$C = VLV^T$$

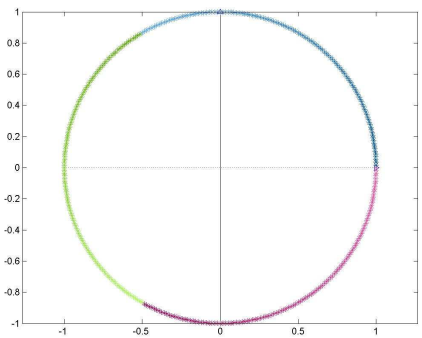
$$\boxed{Sqrt(C) = V.Sqrt(L).V^T}$$

$$\boxed{Sqrt(C).Sqrt(C)} = V.Sqrt(L).V^T V.Sqrt(L).V^T$$

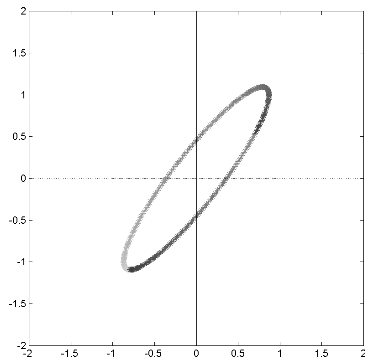
$$= V.Sqrt(L).Sqrt(L)V^T = VLV^T = \boxed{C}$$

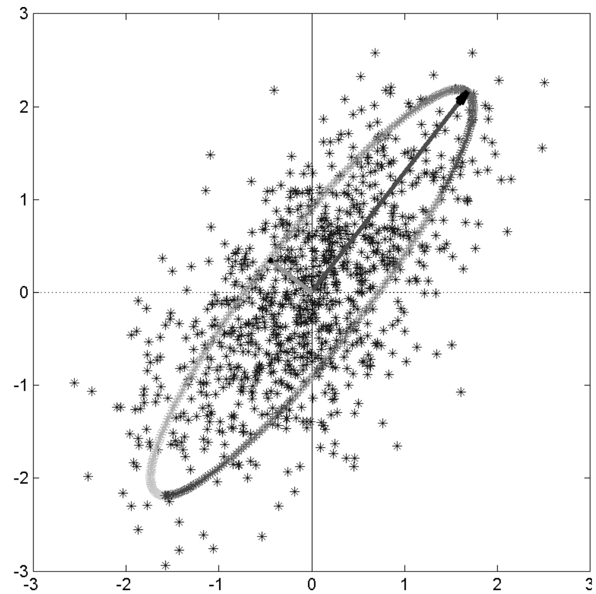
- The correlation / covariance matrix is symmetric
  - Has orthonormal eigen vectors and real, non-negative eigen values
- The *square root* of a correlation or covariance matrix is easily derived from the eigen vectors and eigen values
  - The eigen values of the *square root* of the covariance matrix are the square roots of the eigen values of the covariance matrix
  - These are also the “singular values” of the data set

Square root of the Covariance Matrix



$$\sqrt{C}$$



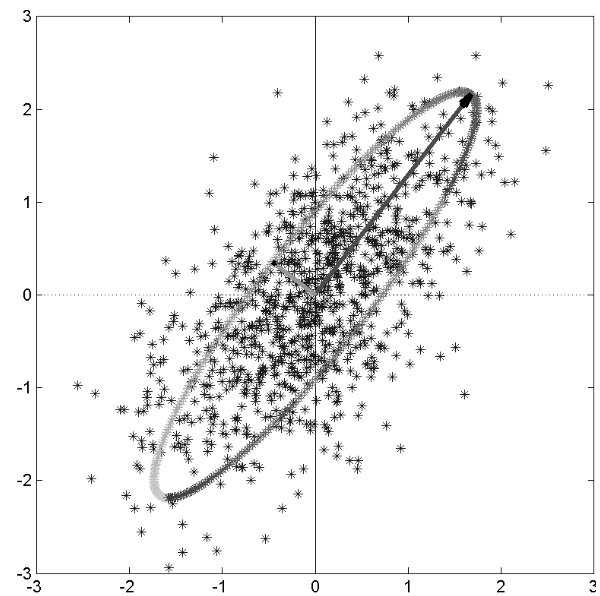


- The square root of the covariance matrix represents the elliptical scatter of the data
- The eigenvectors of the matrix represent the major and minor axes

The Covariance Matrix

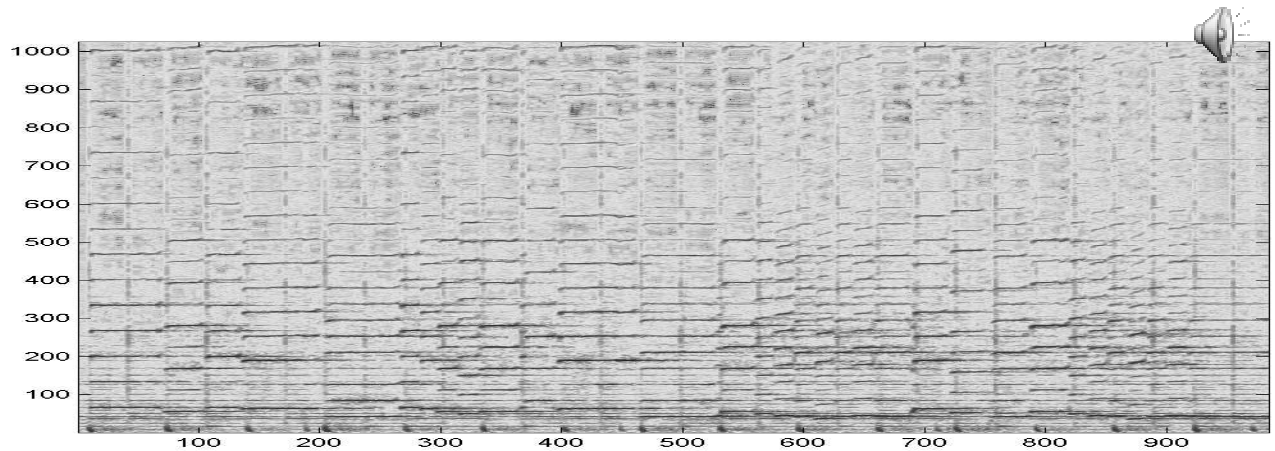
Any vector V = $a_{V,1}$ * eigenvec1 + $a_{V,2}$ *eigenvec2 + ..

$\Sigma_V \, a_{V,i}$ = 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

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

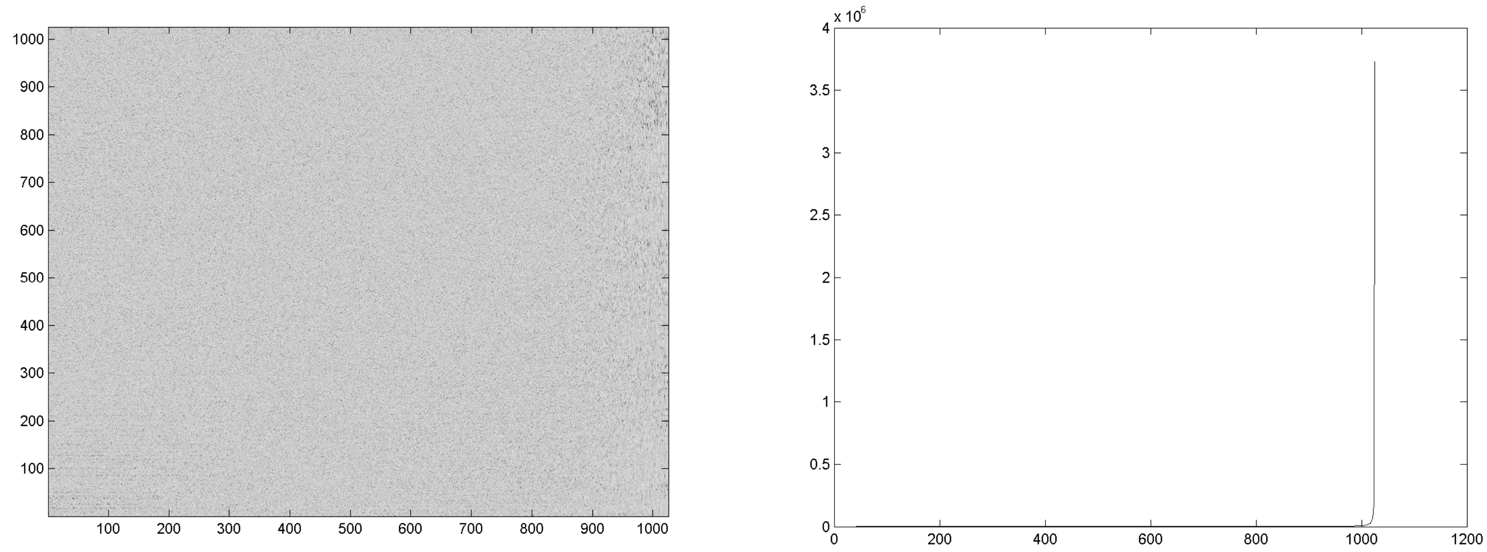
$$V_{reduced} = [V_1 \quad . \quad . \quad V_{25}]$$ 1025x25

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

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

- Compute the Covariance/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



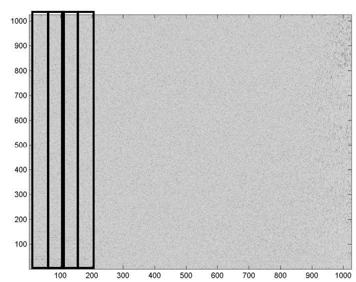
- Left panel: Matrix with 1025 eigen vectors
- Right panel: Corresponding eigen values
  - Most eigen values are close to zero
    - The corresponding eigenvectors are "unimportant"

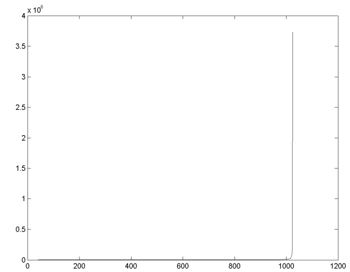
$$M = spectrogram$$

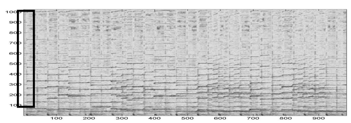
$$C = M.M^T$$

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

Eigenvalues and Eigenvectors





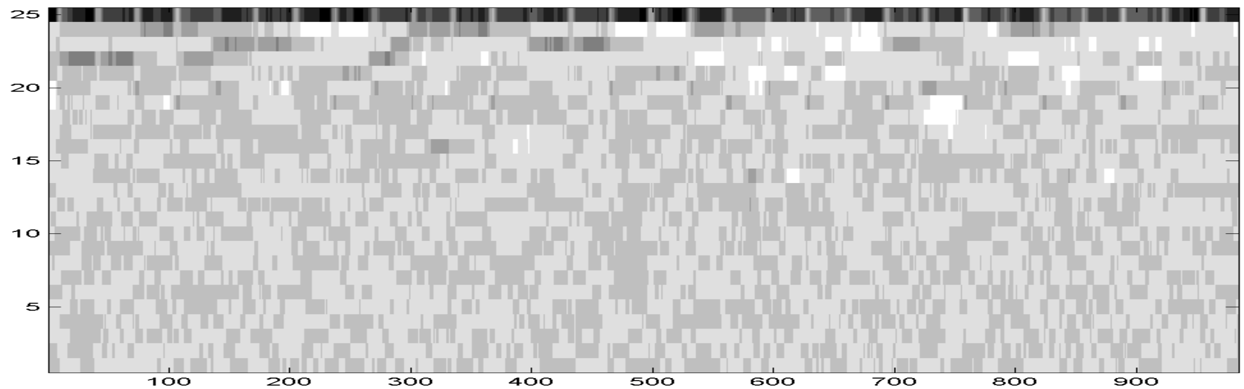


Vec = a1 *eigenvec1 + a2 * eigenvec2 + a3 * eigenvec3 …

- 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

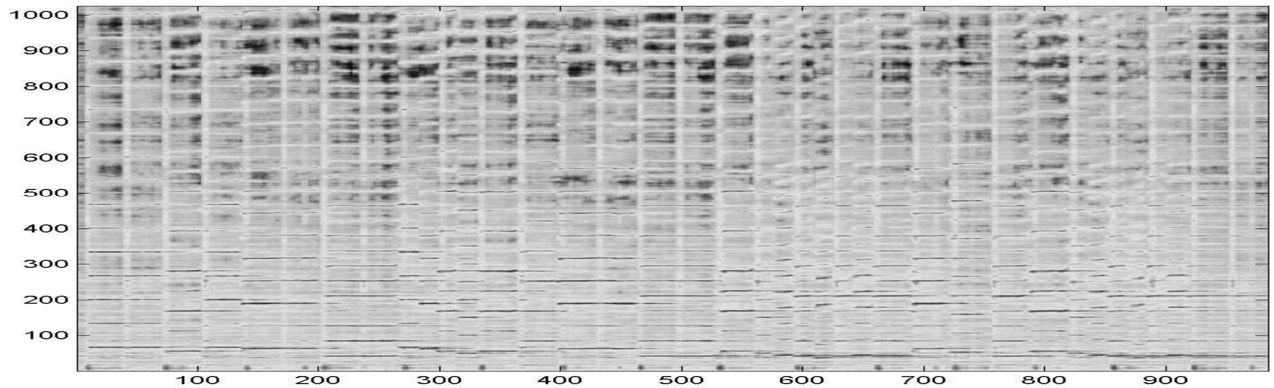
An audio example

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



- 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

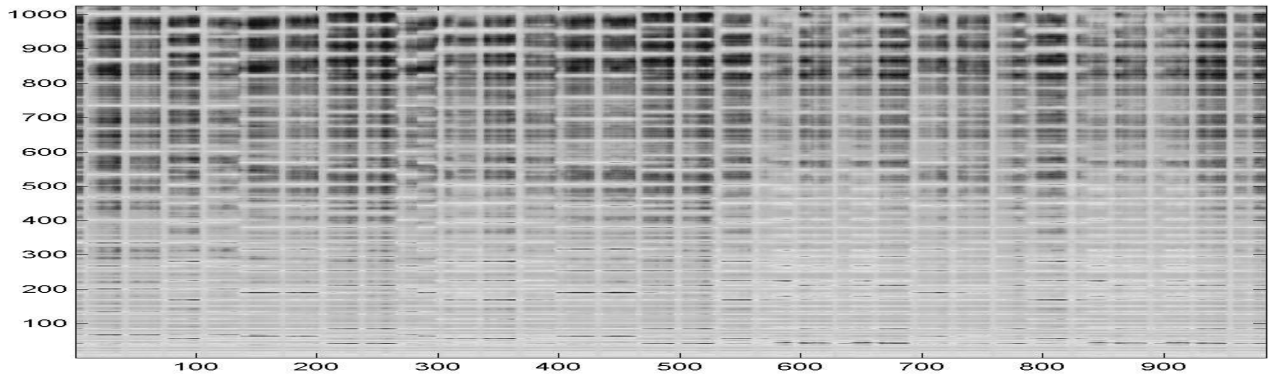
An audio example



$$M_{reconstructed} = V_{reduced} M_{low\dim}$$

- 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

Eigenvectors, Eigenvalues and Covariances

- 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

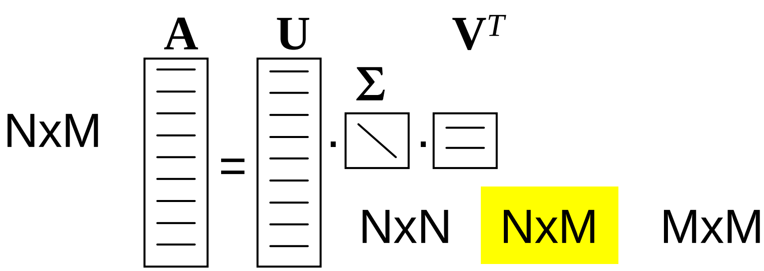
Singular Value Decomposition

- A matrix decomposition method

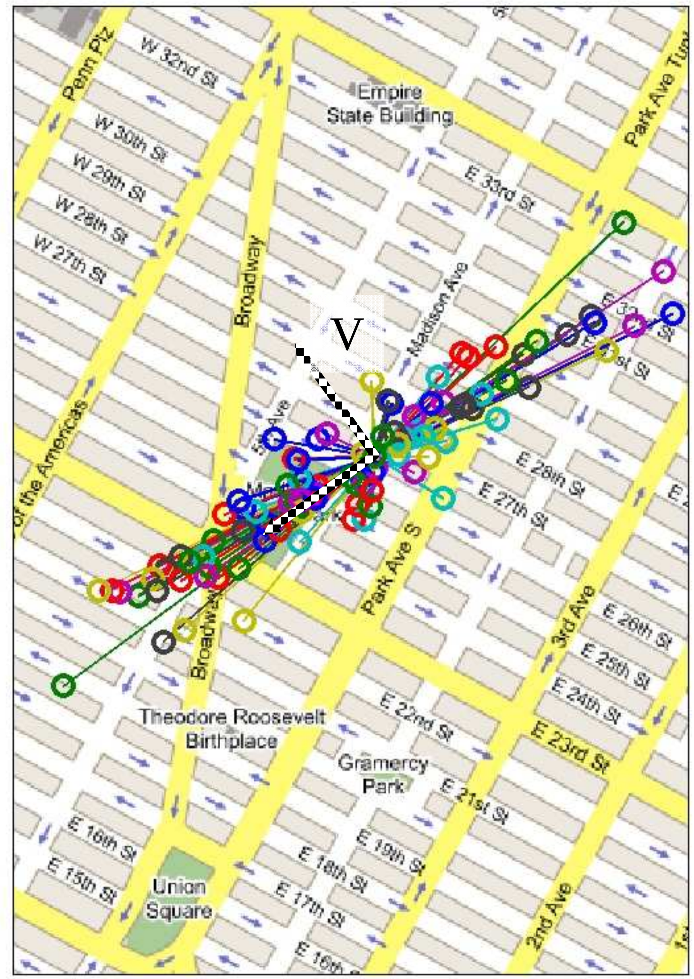
$$\mathbf{A} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T$$

$$\mathbf{U} \cdot \mathbf{U}^T = \mathbf{I}, \;\; \mathbf{V} \cdot \mathbf{V}^T = \mathbf{I}, \;\; \Sigma \text{ is diagonal}$$

- Breaks up the input into a product of three matrices, two orthogonal and one diagonal



- The right matrix will point towards two perpendicular directions on which the greater vector lengths are
- The diagonal will represent how much spread is in each direction and contains the *singular values*
- The left matrix will tell us how the two major directions can be combined to generate the input



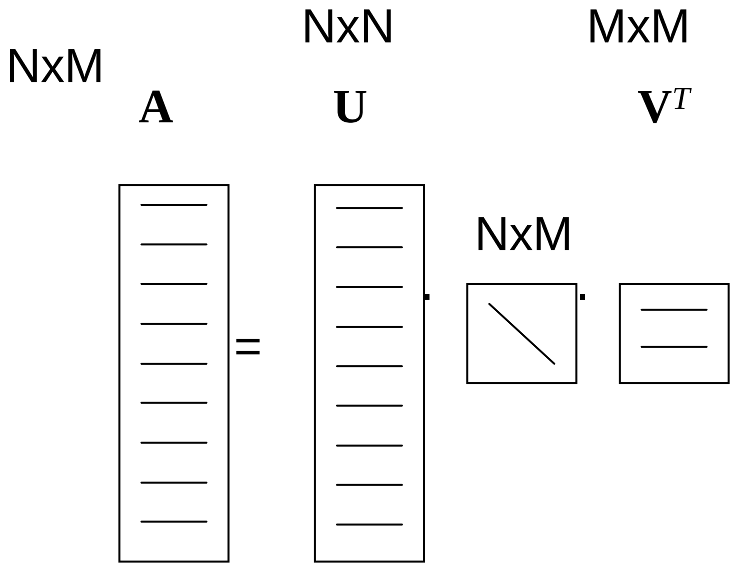
MATLAB syntax:

```
[u,s,v]=svd(x)
```

SVD vs. Eigen decomposition

- Singluar value decomposition is analogous to the eigen decomposition of the correlation matrix of the data
- The “right” singluar 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

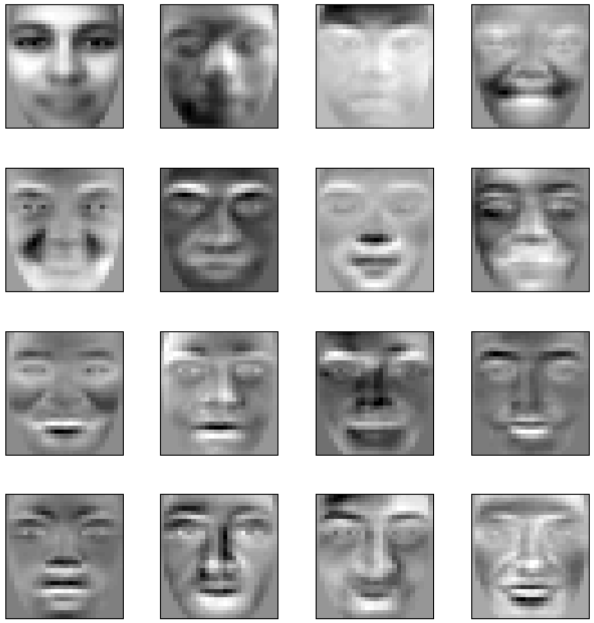
Thin SVD, compact SVD, reduced SVD



- Thin SVD: Only compute the first N columns of U
  - All that is required if N < M
- Compact SVD: Only the left and right eigen vectors corresponding to non-zero singular values are computed
- Reduced SVD: Only compute the columns of U corresponding to the K highest singular values

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

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)`, …

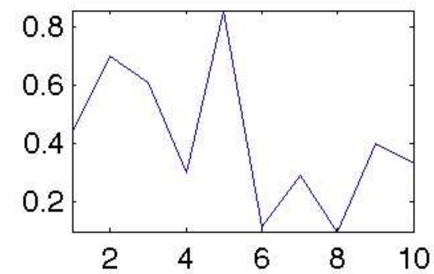
Linear algebra operations

- 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

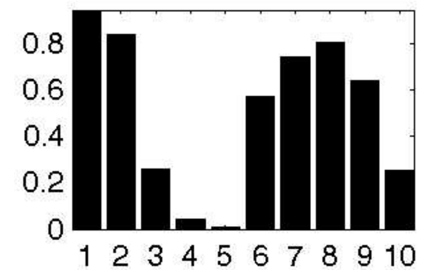
- 1-d plots

  `plot(x)`

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

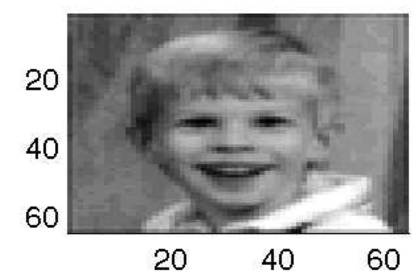
  `bar(x)`

  - Ditto but makes a bar plot

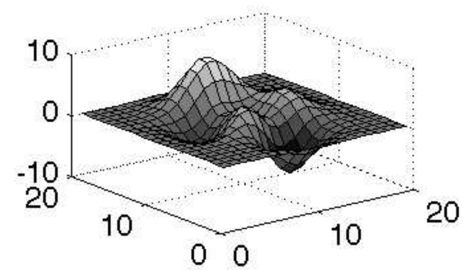
- 2-d plots

  `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