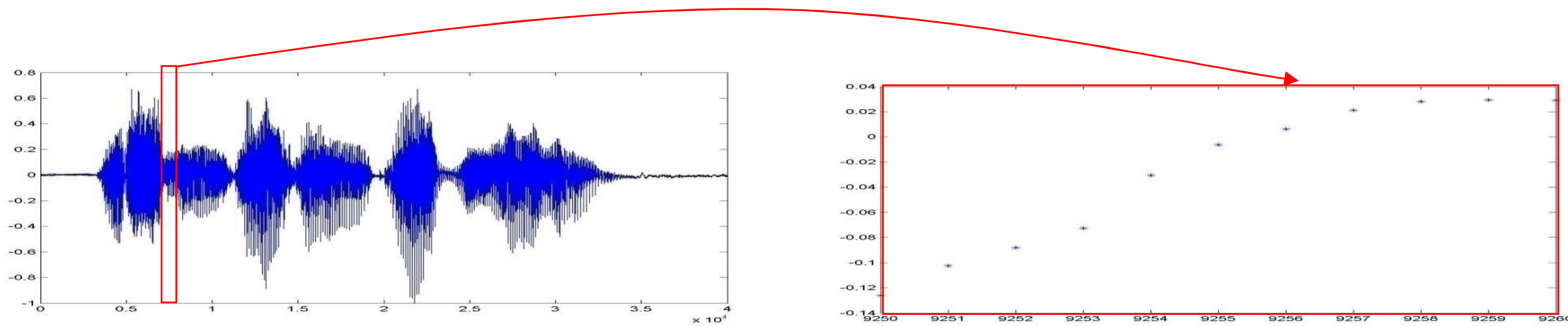
MLSP linear algebra refresher

I learned something old today!

Recap: Representing signals as vectors

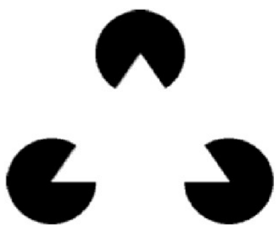
- Signals are frequently represented as vectors for manipulation
- E.g. A segment of an audio signal



- Represented as a vector of sample values

$$[s_1 \;\; s_2 \;\; s_3 \;\; s_4 \; \ldots \; s_N]$$

Representing an image as a vector



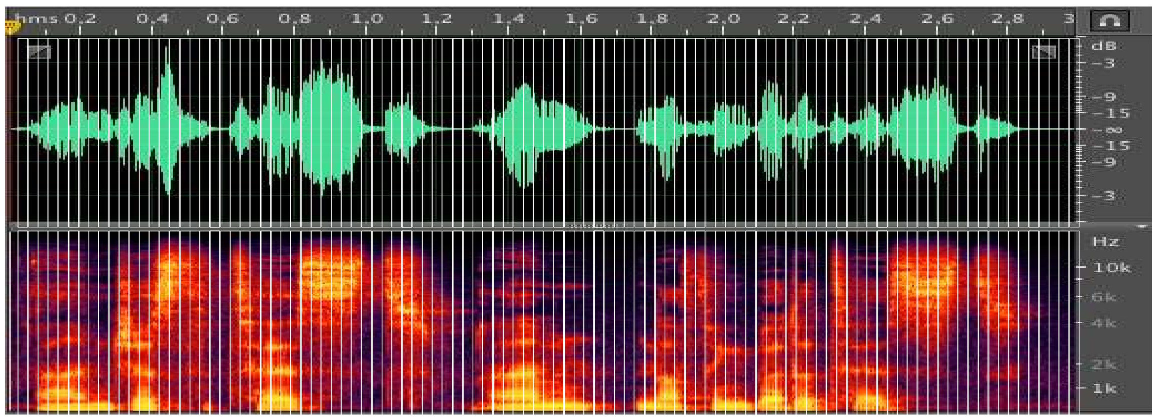
- 3 pacmen
- A 321 x 399 grid of pixel values
  - Row and Column = position
- A 1 x 128079 vector
  - "Unraveling" the matrix

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

  - Note: This can be recast as the grid that forms the image

Representing a signal as a matrix

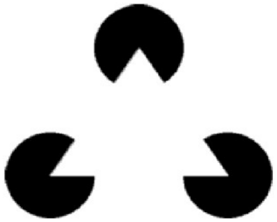
- Time series data like audio signals are often represented as spectrographic matrices



- Each column is the spectrum of a short segment of the audio signal

Representing a signal as a matrix

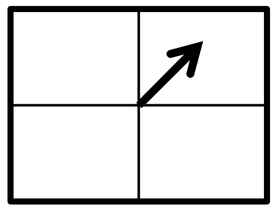
- Images are often just represented as matrices



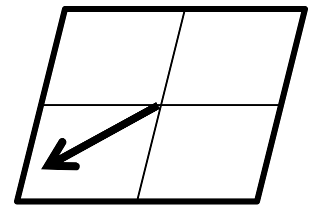
```
>> X(1:32:end,1:40:end)

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
```

Interpretations of a matrix

- As a ***transform*** that modifies vectors and vector spaces



$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

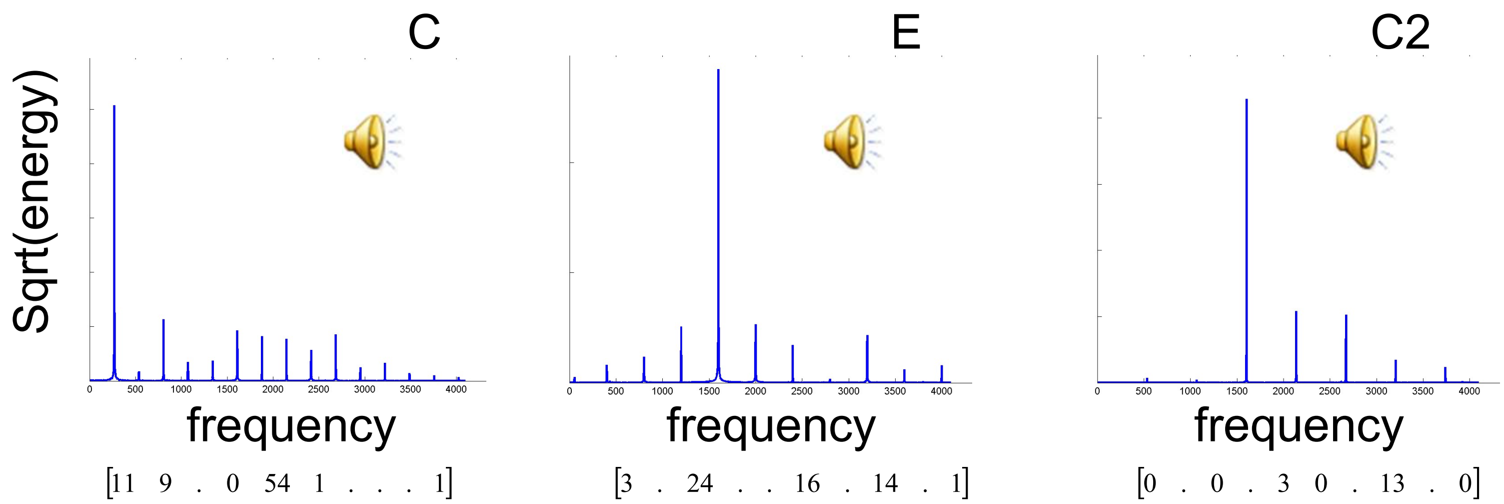


- As a ***container*** for data (vectors)

$$\begin{bmatrix} a & b & c & d & e \\ f & g & h & i & j \\ k & l & m & n & o \end{bmatrix}$$

- As a generator of vector spaces..

Revise.. Vector dot product



- How much of C is also in E
  - How much can you fake a C by playing an E
  - C.E / |C||E| = 0.1
  - Not very much
- How much of C is in C2?
  - C.C2 / |C| /|C2| = 0.5
  - Not bad, you can fake it

Poll 1

Poll 1

- Which of the following is true of a matrix?
  - **It can be a transform**
  - **It can be a data container**
  - **It can be a vector space constructor**
  - **It can be an element of a vector space**

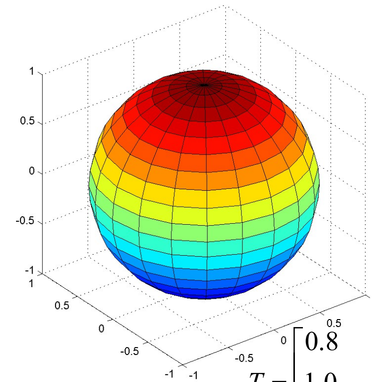
- Which of the following is true of matrix multiplication
  - **It can be expressed as the sum of the outer products of columns of the first matrix with corresponding rows of the second matrix**
  - **It can be expressed the matrix of inner products of the rows of the first matrix and the columns of the second matrix**
  - **The rank of the product is not greater than the lower of the ranks of the two matrices**
  - **If the matrices are both square, the determinant of the product is the product of the determinants**

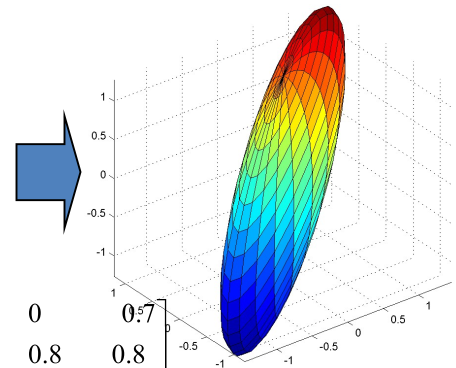
Overview

- Vectors and matrices
- Basic vector/matrix operations
- Various matrix types
- Matrix properties
  - Determinant
  - Inverse
  - Rank
- **Solving simultaneous equations**
- Projections
- Eigen decomposition
- SVD

Matrix Inversion

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



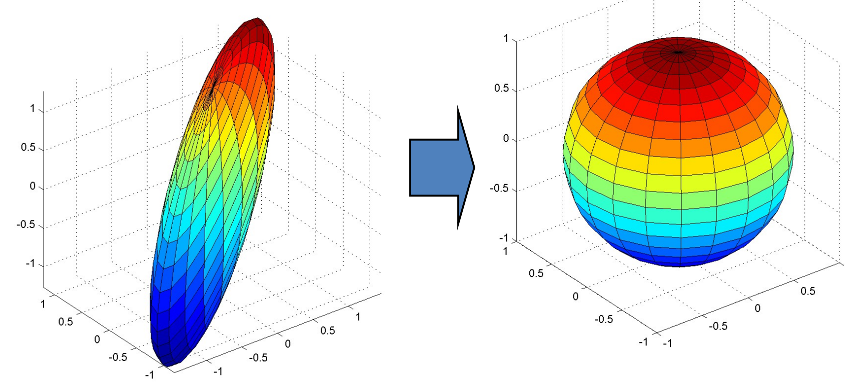


$$T = \begin{bmatrix} 0.8 & 0 & 0.7 \\ 1.0 & 0.8 & 0.8 \\ 0.7 & 0.9 & 0.7 \end{bmatrix}$$

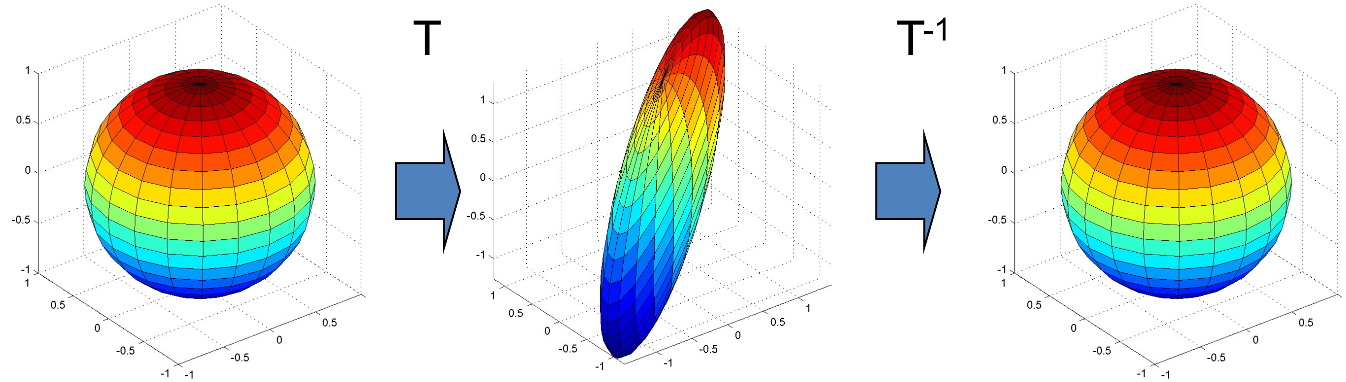
- What transforms the new object back to the original?
  - The *inverse transformation*

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

- The inverse transformation is called the matrix inverse



Matrix Inversion

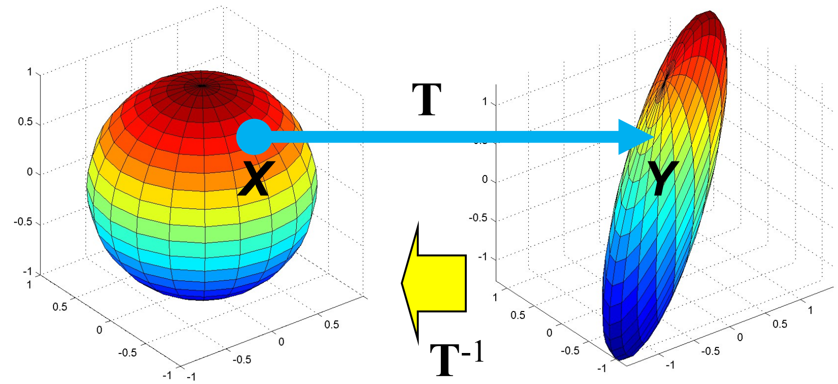


$$\mathbf{T}^{-1}\mathbf{T}\mathbf{D} = \mathbf{D} \Rightarrow \mathbf{T}^{-1}\mathbf{T} = \mathbf{I}$$

- 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

$$\mathbf{T}\mathbf{T}^{-1}\mathbf{D} = \mathbf{D} \Rightarrow \mathbf{T}\mathbf{T}^{-1} = \mathbf{I}$$

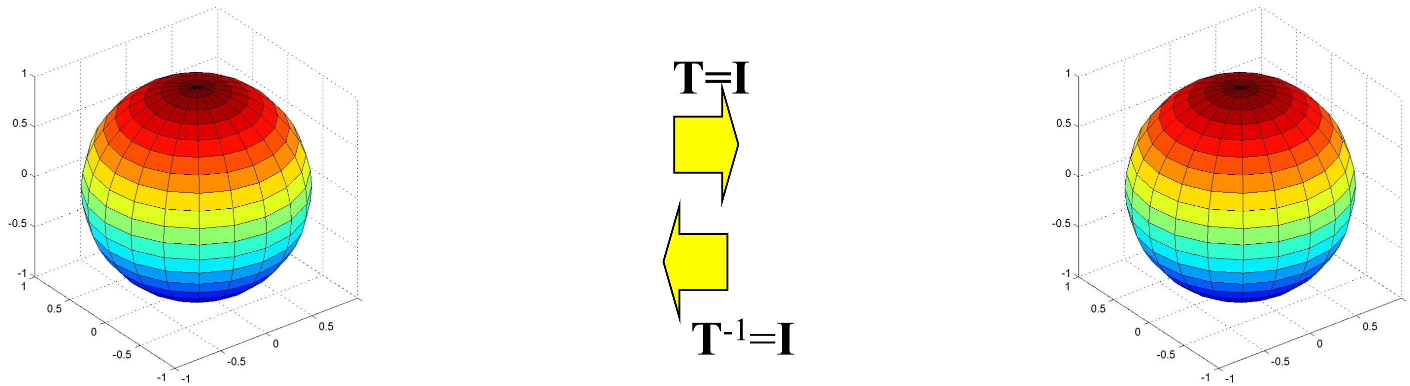
Matrix inversion (division)



- Provides a way to "undo" a linear transform
- Undoing a transform must happen as soon as it is performed
- Effect on matrix inversion: Note order of multiplication

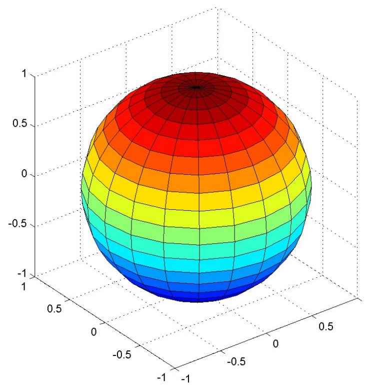
$$\mathbf{A}\cdot\mathbf{B}=\mathbf{C},\;\; \mathbf{A}=\mathbf{C}\cdot\mathbf{B}^{-1},\;\; \mathbf{B}=\mathbf{A}^{-1}\cdot\mathbf{C}$$

Matrix inversion (division)



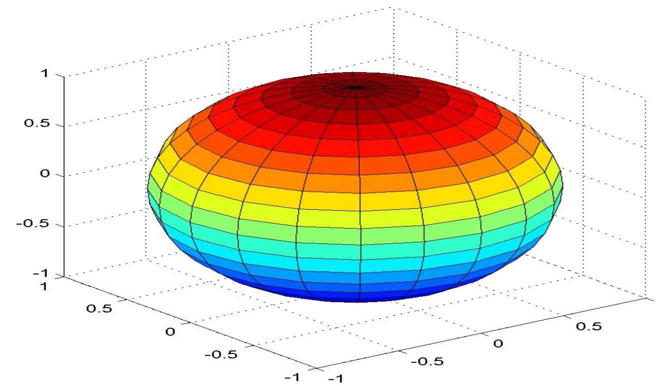
- Inverse of the unit matrix is itself

Matrix inversion (division)



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

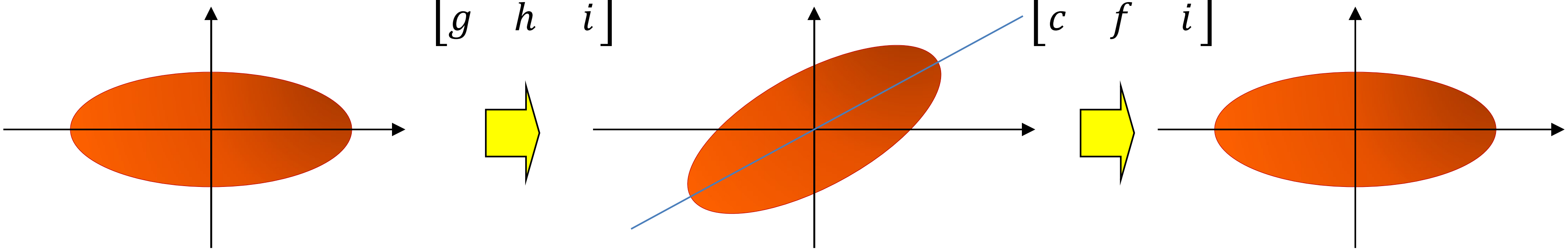
$$\mathbf{T}^{-1} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



- Inverse of the unit matrix is itself
- Inverse of a diagonal is diagonal

Matrix inversion (division)

$$\mathbf{R} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \qquad \mathbf{R}^{-1} = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

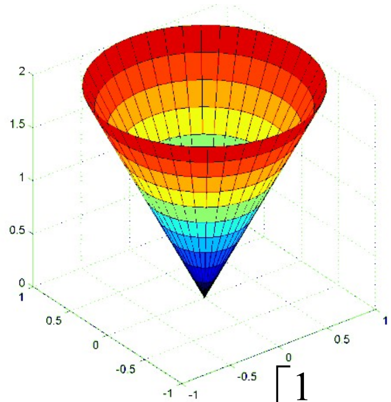


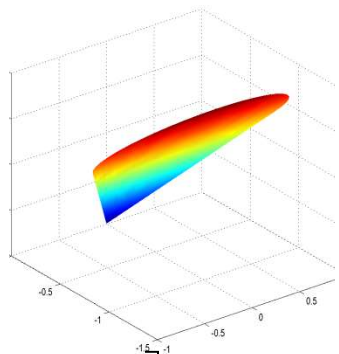
- Inverse of the unit matrix is itself
- Inverse of a diagonal is diagonal
- Inverse of a rotation is a (counter)rotation (its transpose!)
  - In 2D a forward rotation θ by is cancelled by a backward rotation of $-\theta$

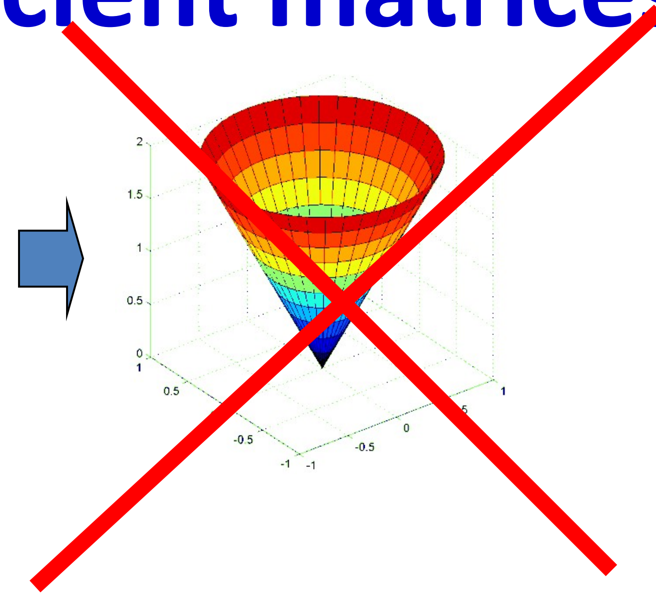
$$\mathbf{R} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \mathbf{R}^{-1} = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

  - More generally, in any number of dimensions: $\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}}$

Inverting rank-deficient matrices



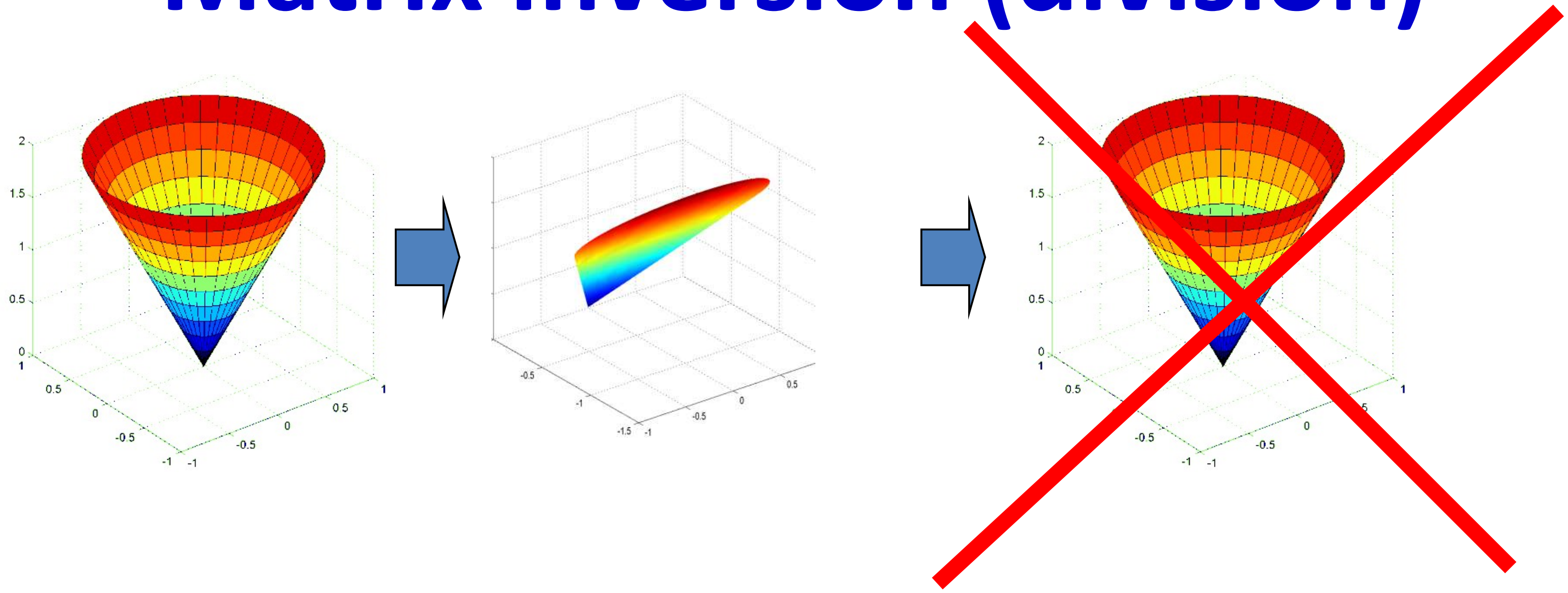




$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & .25 & -0.433 \\ 0 & -0.433 & 0.75 \end{bmatrix}$$

- 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

Matrix inversion (division)



- 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!

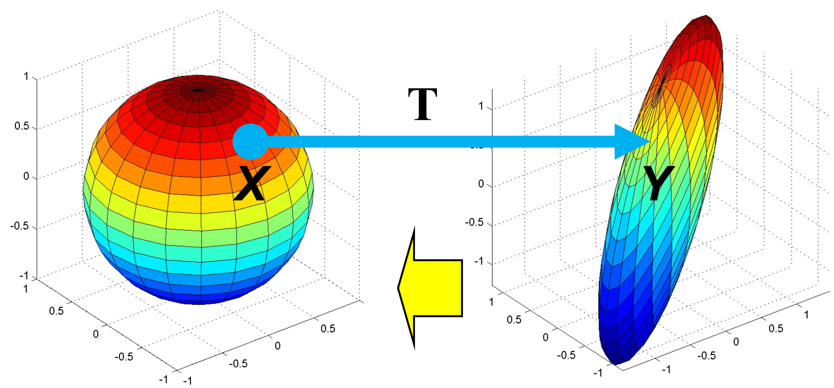
Poll 2

Poll 2

- True or false: Using the standard definition of matrix inversion, the inverse of an arbitrary matrix is always the matrix's transpose.
  - T
  - **F**

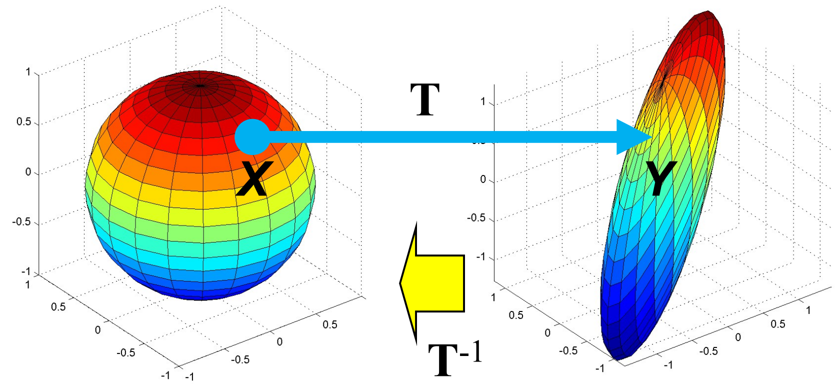
- True or false: Using the standard definition of matrix inversion, the inverse of a rank deficient matrix does NOT exist.
  - **T**
  - F

The Inverse Transform and Simultaneous Equations



- Given the Transform $\mathbf{T}$ and transformed vector ***Y***, how do we determine ***X***?

Matrix inversion (division)



- The inverse of matrix multiplication
  - Not element-wise division!!
  - E.g.

$$\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}^{-1} = \begin{bmatrix} 3/4 & -1/4 & -1/4 \\ -1/4 & 3/4 & -1/4 \\ -1/4 & -1/4 & 3/4 \end{bmatrix}$$

Inverse Transform and Simultaneous Equation

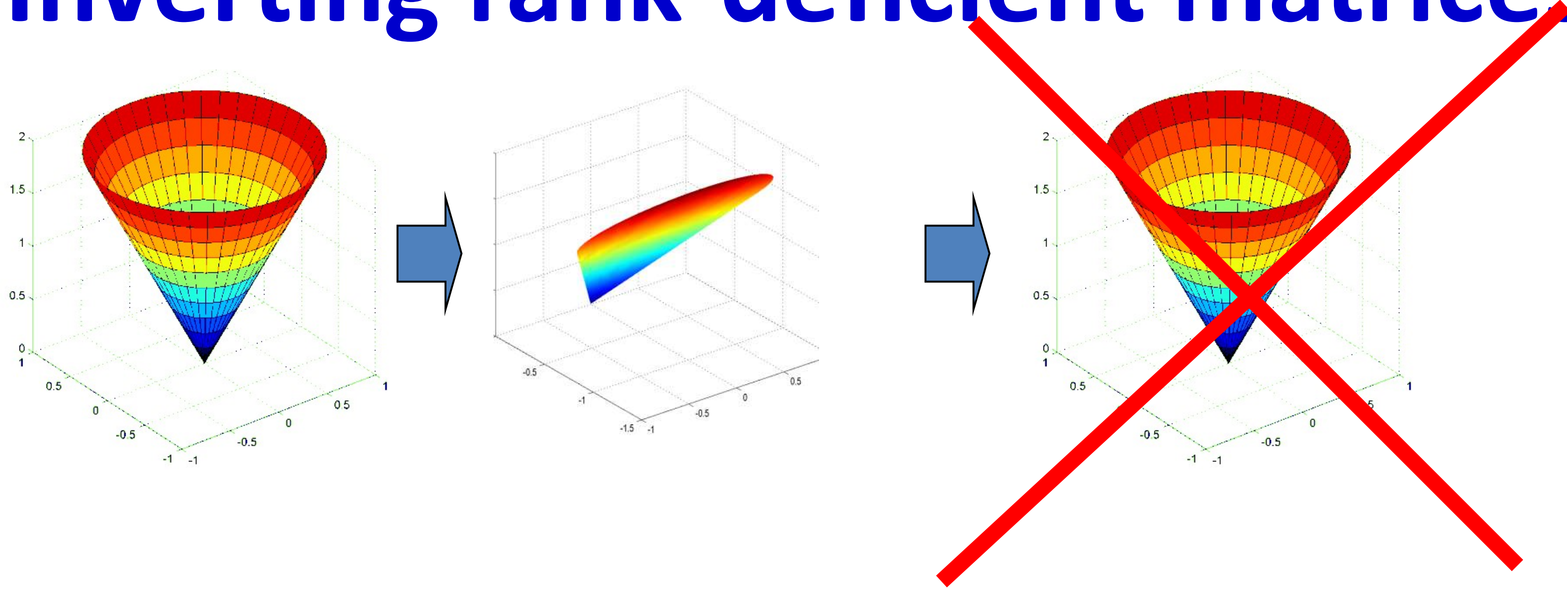
$$\mathbf{T}=\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{aligned} a &= T_{11}x + T_{12}y + T_{13}z \\ b &= T_{21}x + T_{22}y + T_{23}z \\ c &= T_{31}x + T_{32}y + T_{33}z \end{aligned}$$

Given $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ and $\mathbf{T}$ find $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$

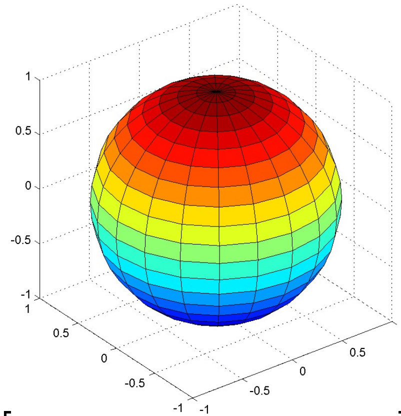
- Inverting the transform is identical to solving simultaneous equations

Inverting rank-deficient matrices



- Rank deficient matrices have no inverse
  - In this example, there is no *unique* inverse
- Here the simultaneous equations will not have a unique solution
  - Many different 3D objects could compress down to the same 2D image in our example

Non-square Matrices

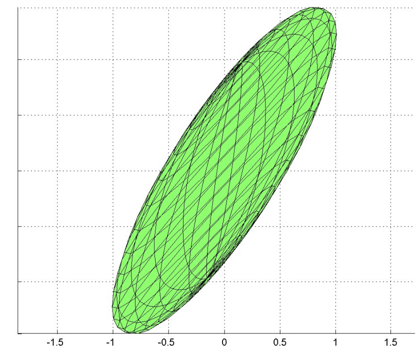


$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \\ z_1 & z_2 & . & . & z_N \end{bmatrix}$$

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
  - Fewer rows than columns → reduce axes
    - May reduce dimensionality of the data

Inverse Transform and Simultaneous Equation

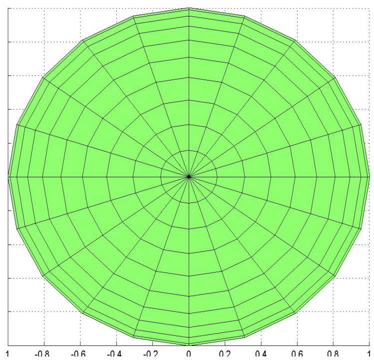
$$\mathbf{T}=\begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{T}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{array}{l} a = T_{11}x + T_{12}y + T_{13}z \\ b = T_{21}x + T_{22}y + T_{23}z \end{array}$$

Given $\begin{bmatrix} a \\ b \end{bmatrix}$ and $\mathbf{T}$ find $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$

- Inverting the transform is identical to solving simultaneous equations
- Rank-deficient transforms result in too-few ***independent*** equations
  - Cannot be inverted to obtain a *unique* solution

Non-square Matrices

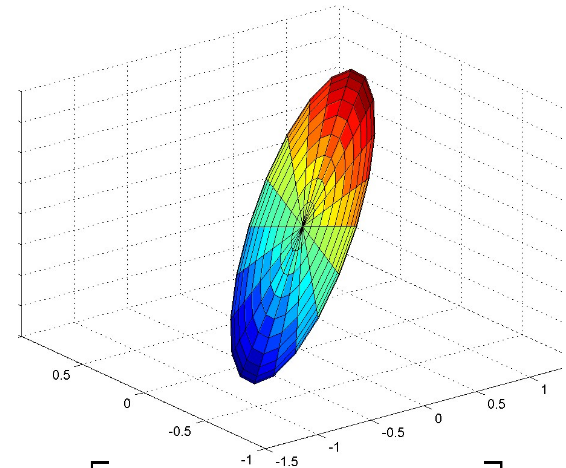


$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \end{bmatrix}$$

X = 2D data

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

P = transform

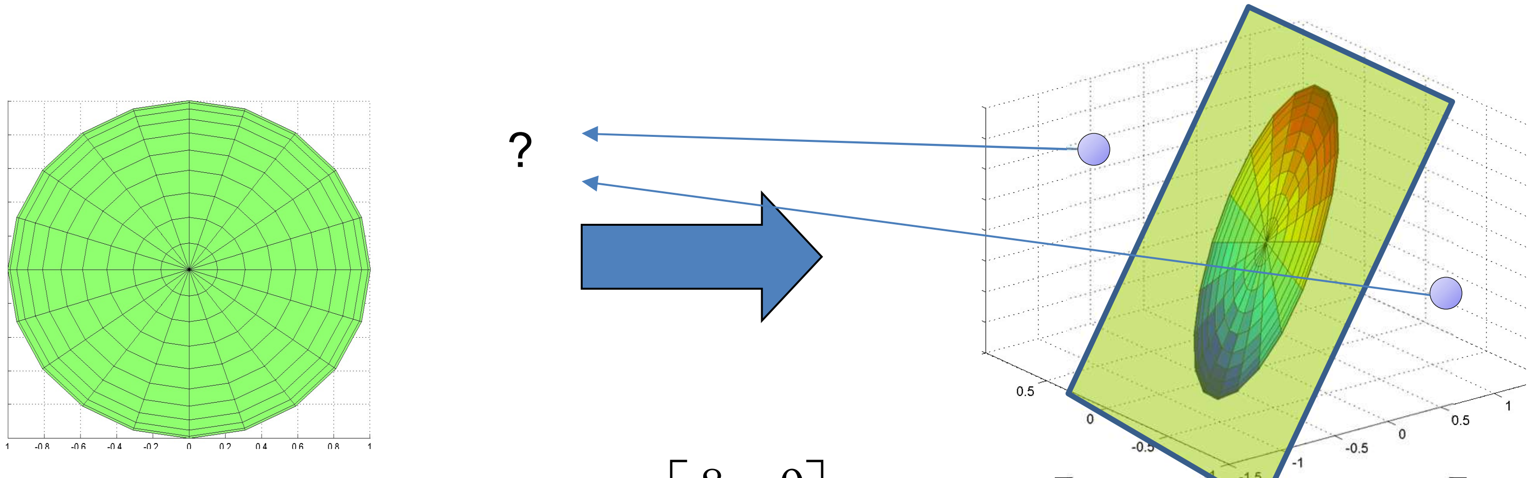


$$\begin{bmatrix} \hat{x}_1 & \hat{x}_2 & . & . & \hat{x}_N \\ \hat{y}_1 & \hat{y}_2 & . & . & \hat{y}_N \\ \hat{z}_1 & \hat{z}_2 & . & . & \hat{z}_N \end{bmatrix}$$

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



$$\begin{bmatrix} x_1 & x_2 & . & . & x_N \\ y_1 & y_2 & . & . & y_N \end{bmatrix} \qquad \begin{bmatrix} .8 & .9 \\ .1 & .9 \\ .6 & 0 \end{bmatrix} \qquad \begin{bmatrix} \hat{x}_1 & \hat{x}_2 & . & . & \hat{x}_N \\ \hat{y}_1 & \hat{y}_2 & . & . & \hat{y}_N \\ \hat{z}_1 & \hat{z}_2 & . & . & \hat{z}_N \end{bmatrix}$$

X = 2D data | P = transform | PX = 3D, rank 2

- When the transform *increases* the number of components most points in the new space will not have a corresponding preimage

Inverse Transform and Simultaneous Equation

$$\mathbf{T}=\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \\ T_{31} & T_{32} \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{T}\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{aligned} a &= T_{11}x + T_{12}y \\ b &= T_{21}x + T_{22}y \\ c &= T_{31}x + T_{32}y \end{aligned}$$

Given $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ and $\mathbf{T}$ find $\begin{bmatrix} x \\ y \end{bmatrix}$

- Inverting the transform is identical to solving simultaneous equations
- Rank-deficient transforms result in too few independent equations
  - Cannot be inverted to obtain a unique solution
- Or too *many* equations
  - Cannot be inverted to obtain an exact solution

The Pseudo Inverse (PINV)

$$V \approx T \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} \approx Pinv(T)V$$

- When you can't *really* invert T, you perform the *pseudo* inverse

Generalization to matrices

- **Unique exact solution exists**
- **T** must be square

$$\boxed{\mathbf{X} = \mathbf{TY} \Rightarrow \mathbf{Y} = \mathbf{T}^{-1}\mathbf{X}}$$

Left multiplication

$$\boxed{\mathbf{X} = \mathbf{YT} \Rightarrow \mathbf{Y} = \mathbf{XT}^{-1}}$$

Right multiplication

- **No unique exact solution exists**
  - At least one (if not both) of the forward and backward equations may be inexact
- **T** may or may not be square

$$\boxed{\mathbf{X} = \mathbf{TY} \Rightarrow \mathbf{Y} = \mathrm{Pinv}(\mathbf{T})\mathbf{X}}$$

Left multiplication

$$\boxed{\mathbf{X} = \mathbf{YT} \Rightarrow \mathbf{Y} = \mathbf{X}\mathrm{Pinv}(\mathbf{T})}$$

Right multiplication

Underdetermined Pseudo Inverse

$$\begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{aligned} a &= T_{11}x + T_{12}y + T_{13}z \\ b &= T_{21}x + T_{22}y + T_{23}z \end{aligned}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = Pinv(\mathbf{T}) \begin{bmatrix} a \\ b \end{bmatrix}$$

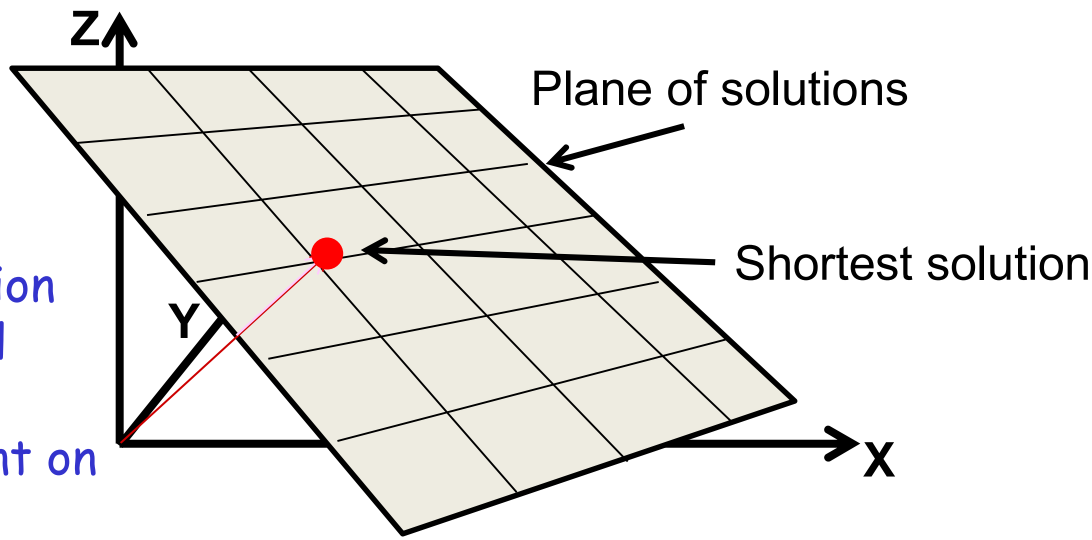


Figure only meant for illustration for the above equations, actual set of solutions is a line, not a plane. Pinv(T)A will be the point on the line closest to origin

- **Case 1:** Too many solutions
- Pinv(**T**)**A** picks the *shortest* solution

The Pseudo Inverse for the underdetermined case

$$\begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{matrix} a = T_{11}x + T_{12}y + T_{13}z \\ b = T_{21}x + T_{22}y + T_{23}z \end{matrix}$$

$$V \approx T \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} = Pinv(T)V$$

$$Pinv(\boldsymbol{T}) = \boldsymbol{T}^T(\boldsymbol{T}\boldsymbol{T}^T)^{-1}$$

$$\boldsymbol{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \boldsymbol{T}Pinv(\boldsymbol{T})\boldsymbol{V} = \boldsymbol{T}\boldsymbol{T}^T(\boldsymbol{T}\boldsymbol{T}^T)^{-1}\boldsymbol{V} = \boldsymbol{V}$$

Exact recovery

Overdetermined: Pseudo Inverse

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \\ T_{31} & T_{32} \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{T}\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{aligned} a &= T_{11}x + T_{12}y \\ b &= T_{21}x + T_{22}y \\ c &= T_{31}x + T_{32}y \end{aligned}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = Pinv(\mathbf{T})\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$||\mathbf{A} - \mathbf{TX}||^2$$

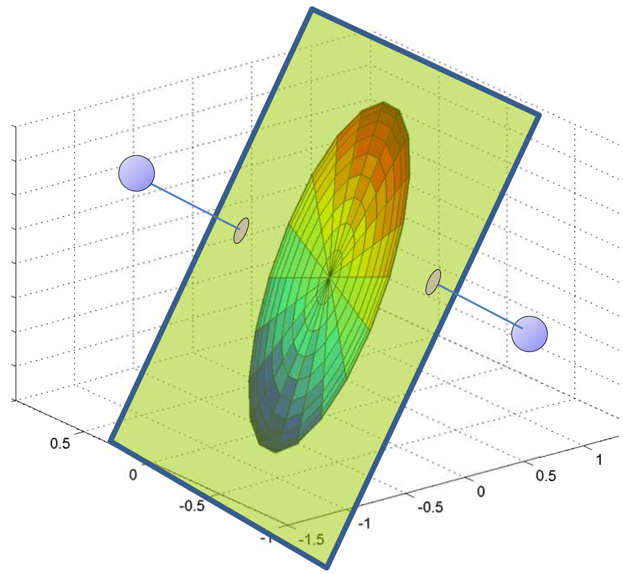


Figure only meant for illustration for the above equations, Pinv(T) will actually have 6 components. The error is a quadratic in 6 dimensions

- **Case 2:** No exact solution
- $\mathrm{Pinv}(\mathbf{T})\mathbf{A}$ picks the solution that results in the lowest error

The Pseudo Inverse for the overdetermined case

$$E = \|\boldsymbol{TX} - \boldsymbol{A}\|^2 = (\boldsymbol{TX} - \boldsymbol{A})^T(\boldsymbol{TX} - \boldsymbol{A})$$

$$E = \boldsymbol{X}^T\boldsymbol{T}^T\boldsymbol{TX} - 2\boldsymbol{X}^T\boldsymbol{T}^T\boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{A}$$

Differentiating and equating to 0 we get:

$$\boldsymbol{X} = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{A} \quad = \quad Pinv(\boldsymbol{T})\boldsymbol{A}$$

$$\boxed{Pinv(\boldsymbol{T}) = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T}$$

Shortcut: overdetermined case

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{T}\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{aligned} a &= T_{11}x + T_{12}y \\ b &= T_{21}x + T_{22}y \\ c &= T_{31}x + T_{32}y \end{aligned}$$

$$\boldsymbol{V} \approx \boldsymbol{T}\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \boldsymbol{T}^T\boldsymbol{V} \approx \boldsymbol{T}^T\boldsymbol{T}\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{V}$$

$$\boxed{Pinv(\boldsymbol{T}) = (\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T}$$

Note that in this case:

$$\boldsymbol{T}\begin{bmatrix} x \\ y \end{bmatrix} = \boldsymbol{T}Pinv(\boldsymbol{T})\boldsymbol{V} = \boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{V} \neq \boldsymbol{V}$$

Why?

Overdetermined vs Underdetermined

- Underdetermined case: Exact solution exists. We find *one* of the exact solutions. Hence..

$$\boldsymbol{T}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \boldsymbol{T}Pinv(\boldsymbol{T})\boldsymbol{V} = \boldsymbol{T}\boldsymbol{T}^T(\boldsymbol{T}\boldsymbol{T}^T)^{-1}\boldsymbol{V} = \boldsymbol{V}$$

- Overdetermined case: Solution generally does not exist. Solution is only an approximation..

$$\boldsymbol{T}\begin{bmatrix} x \\ y \end{bmatrix} = \boldsymbol{T}Pinv(\boldsymbol{T})\boldsymbol{V} = \boldsymbol{T}(\boldsymbol{T}^T\boldsymbol{T})^{-1}\boldsymbol{T}^T\boldsymbol{V} \neq \boldsymbol{V}$$

Properties of the Pseudoinverse

- For the underdetermined case:

$$\boldsymbol{T} Pinv(\boldsymbol{T}) = \mathbf{I}$$

- For the overdetermined case

$$\boldsymbol{T} Pinv(\boldsymbol{T}) = ?$$

  – We return to this question shortly

Matrix inversion (division)

- The inverse of matrix multiplication
  - Not element-wise division!!
- Provides a way to "undo" a linear transformation

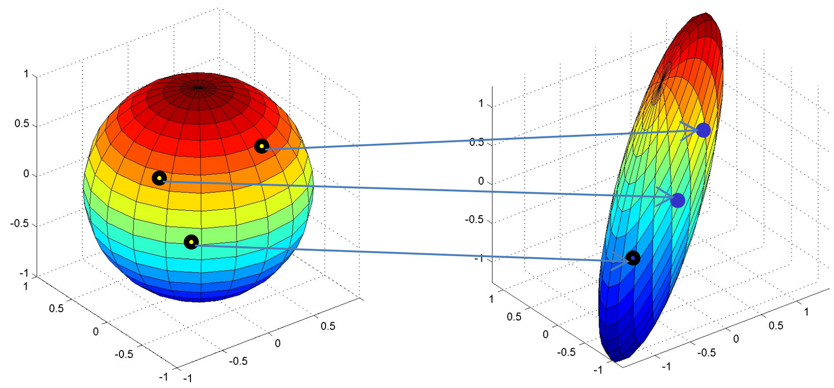
- For square matrices: Pay attention to multiplication side!

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

- If matrix is not square use a matrix pseudoinverse:

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

Finding the Transform

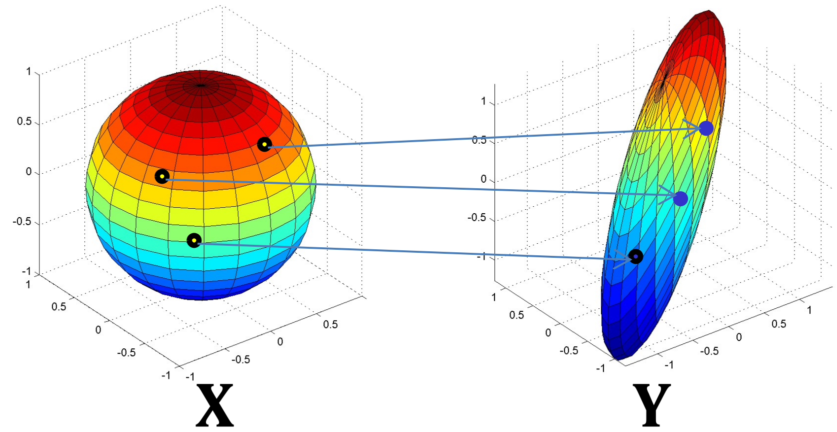


- Given examples
  - $\mathbf{T}.X_1 = Y_1$
  - $\mathbf{T}.X_2 = Y_2$
  - ..
  - $\mathbf{T}.X_N = Y_N$
- Find $\mathbf{T}$

Finding the Transform

$$\mathbf{X} = \begin{bmatrix} \uparrow & \vdots & \uparrow \\ X_1 & \ddots & X_N \\ \downarrow & \vdots & \downarrow \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} \uparrow & \vdots & \uparrow \\ Y_1 & \ddots & Y_N \\ \downarrow & \vdots & \downarrow \end{bmatrix}$$



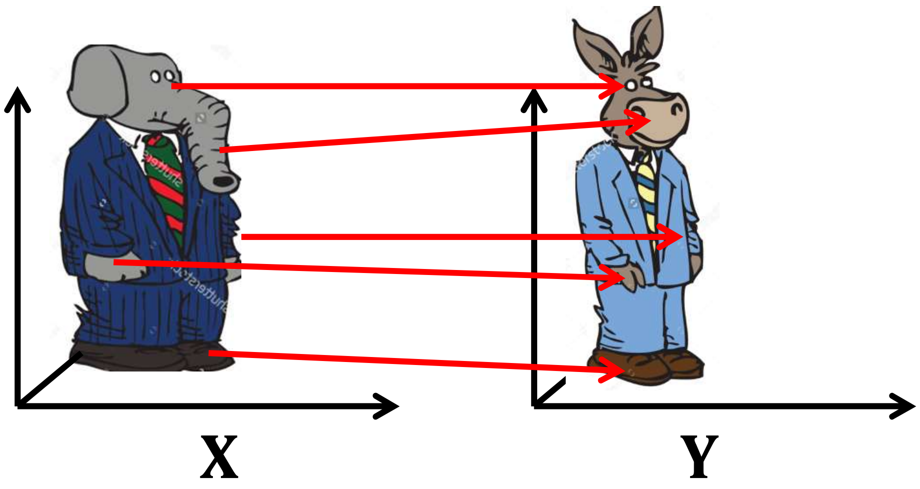
$$\mathbf{Y} = \mathbf{T}\mathbf{X} \qquad \mathbf{T} = \mathbf{Y}\mathrm{Pinv}(\mathbf{X})$$

- Pinv works here too

Finding the Transform: Inexact

$$\mathbf{X} = \begin{bmatrix} \uparrow & \vdots & \uparrow \\ X_1 & \ddots & X_N \\ \downarrow & \vdots & \downarrow \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} \uparrow & \vdots & \uparrow \\ Y_1 & \ddots & Y_N \\ \downarrow & \vdots & \downarrow \end{bmatrix}$$



$$\mathbf{Y} \approx \mathbf{TX} \Rightarrow \quad \mathbf{T} = \mathbf{Y}\mathrm{Pinv}(\mathbf{X})$$

$$\text{minimizes } \sum_i ||\mathrm{Y_i} - \mathbf{T}\mathrm{X_i}||^2$$

- Even works for inexact solutions
- We *desire* to find a linear transform **T** that maps **X** to **Y**
  - But such a linear transform doesn't really exist
- *Pinv* will give us the "best guess" for **T** that minimizes the total squared error between **Y** and **TX**

Poll 3

Poll 3

- Which of the following is **NOT** true of the pseudoinverse?
  - The pseudoinverse provides a way to approximate the inverse for rank-deficient matrices
  - The pseudoinverse can be used to “undo” a linear transformation
  - **The pseudoinverse provides a way to recover a linear transform matrix exactly**
  - There is exactly one unique solution for the pseudoinverse in the underdetermined case

Overview

- Vectors and matrices
- Basic vector/matrix operations
- Various matrix types
- Matrix properties
  - Determinant
  - Inverse
  - Rank
- Solving simultaneous equations
- **Projections**
- Eigen decomposition
- SVD

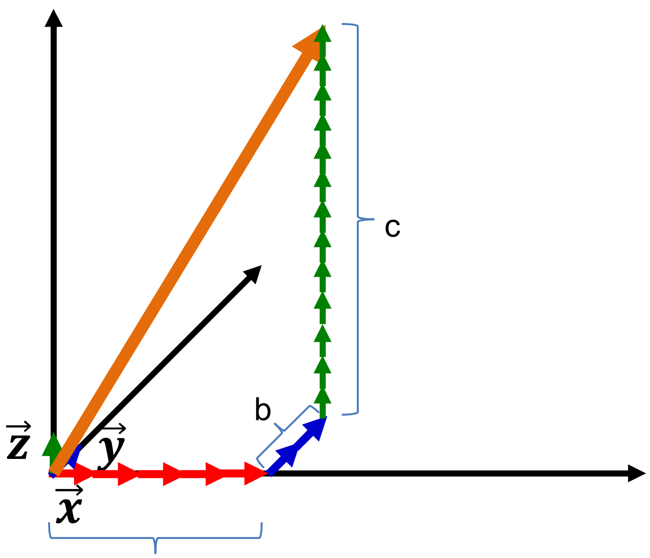
Flashback: The *true* representation of a vector

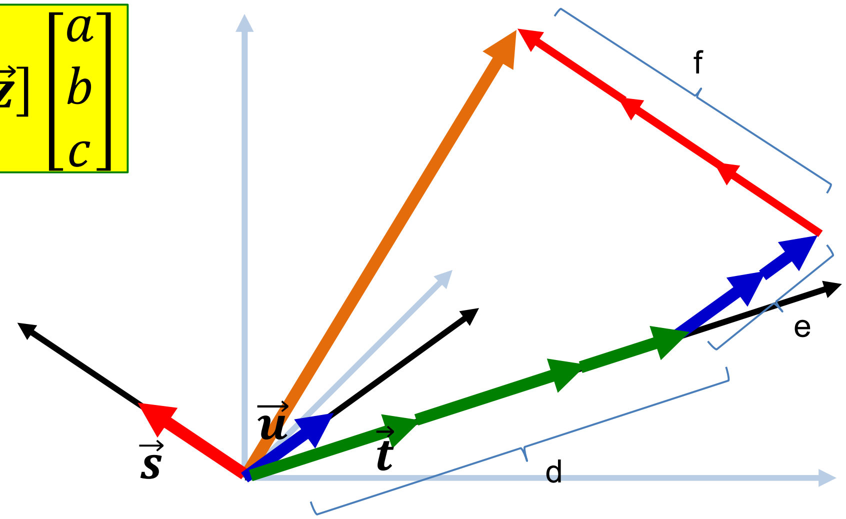
$$\boldsymbol{v} = d\vec{\boldsymbol{u}} + e\vec{\boldsymbol{v}} + f\vec{\boldsymbol{w}} \qquad \boldsymbol{v} = \begin{bmatrix} d \\ e \\ f \end{bmatrix}$$

$$\boldsymbol{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \text{using } \vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}, \vec{\boldsymbol{z}}$$

$$\boldsymbol{v} = a\vec{\boldsymbol{x}} + b\vec{\boldsymbol{y}} + c\vec{\boldsymbol{z}}$$

$$\boldsymbol{v} = [\vec{\boldsymbol{x}} \quad \vec{\boldsymbol{y}} \quad \vec{\boldsymbol{z}}] \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$





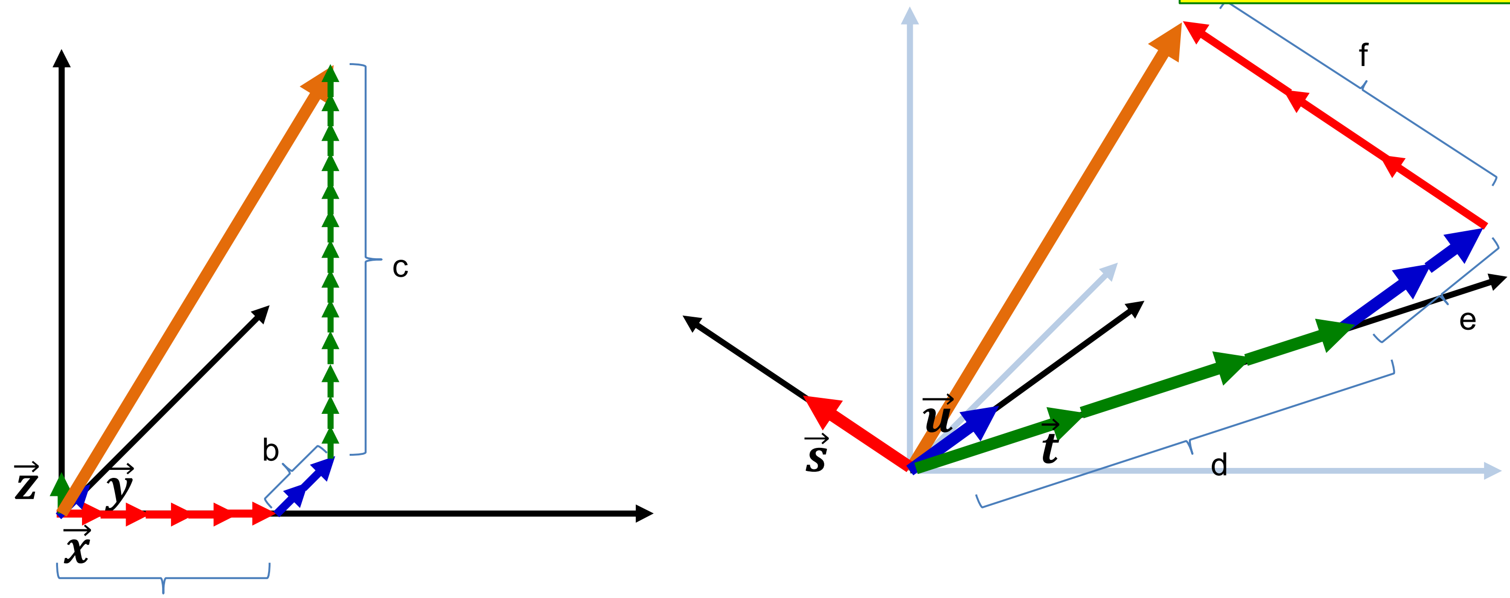
$$\boldsymbol{v} = [\vec{\boldsymbol{s}} \quad \vec{\boldsymbol{t}} \quad \vec{\boldsymbol{u}}] \begin{bmatrix} d \\ e \\ f \end{bmatrix}$$

- What the column (or row) of numbers really means
  - The “basis matrix” is implicit

Flashforward: Changing bases

$$v = [\vec{x} \quad \vec{y} \quad \vec{z}]\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$v = [\vec{s} \quad \vec{t} \quad \vec{u}]\begin{bmatrix} d \\ e \\ f \end{bmatrix}$$



- Given representation $[a, b, c]$ and bases $\vec{x} \quad \vec{y} \quad \vec{z}$, how do we derive the representation $[d\ e\ f]$ in terms of a different set of bases $\vec{s} \quad \vec{t} \quad \vec{u}$?

Matrix as a Basis transform

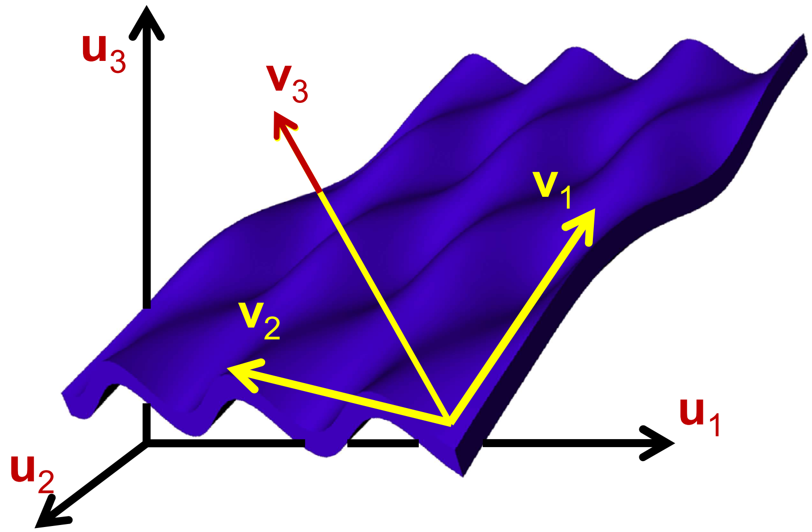
$$\mathbf{X} = a\mathbf{v}_1 + b\mathbf{v}_2 + c\mathbf{v}_3, \Leftarrow \mathbf{X} = x\mathbf{u}_1 + y\mathbf{u}_2 + z\mathbf{u}_3$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- A matrix transforms a representation in terms of a standard basis $\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_3$ to a representation in terms of a different bases $\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_3$

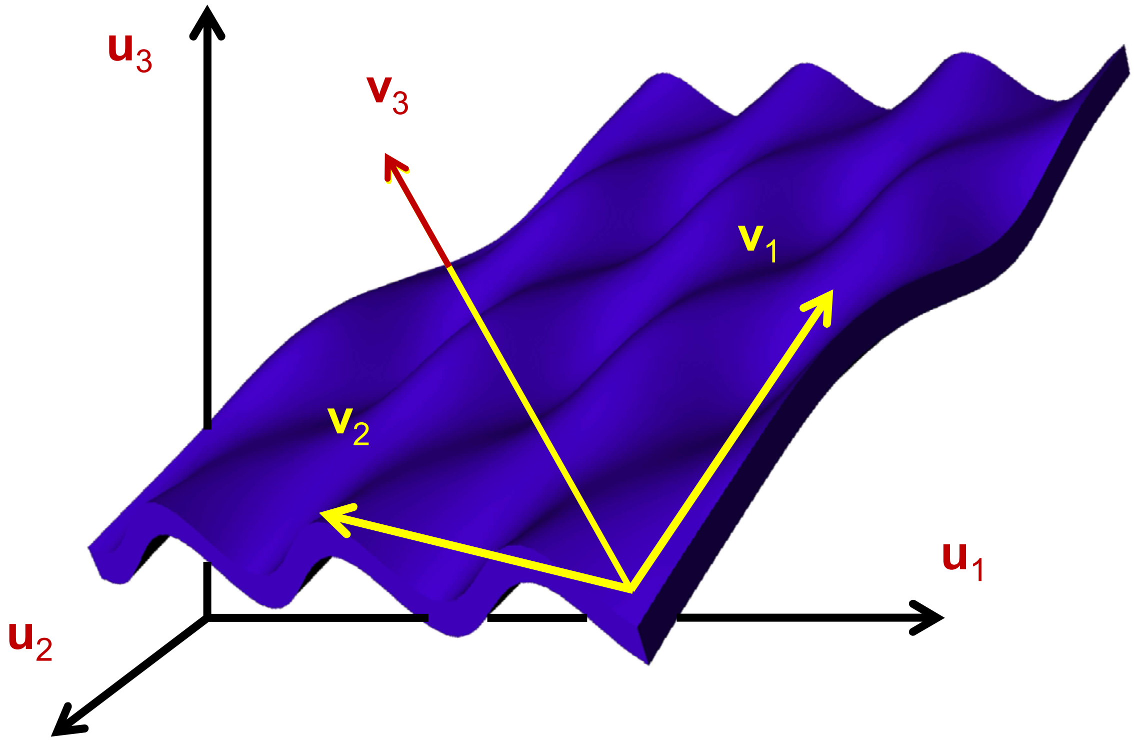
- Finding best bases: Find matrix that transforms standard representation to these bases

Basis based representation



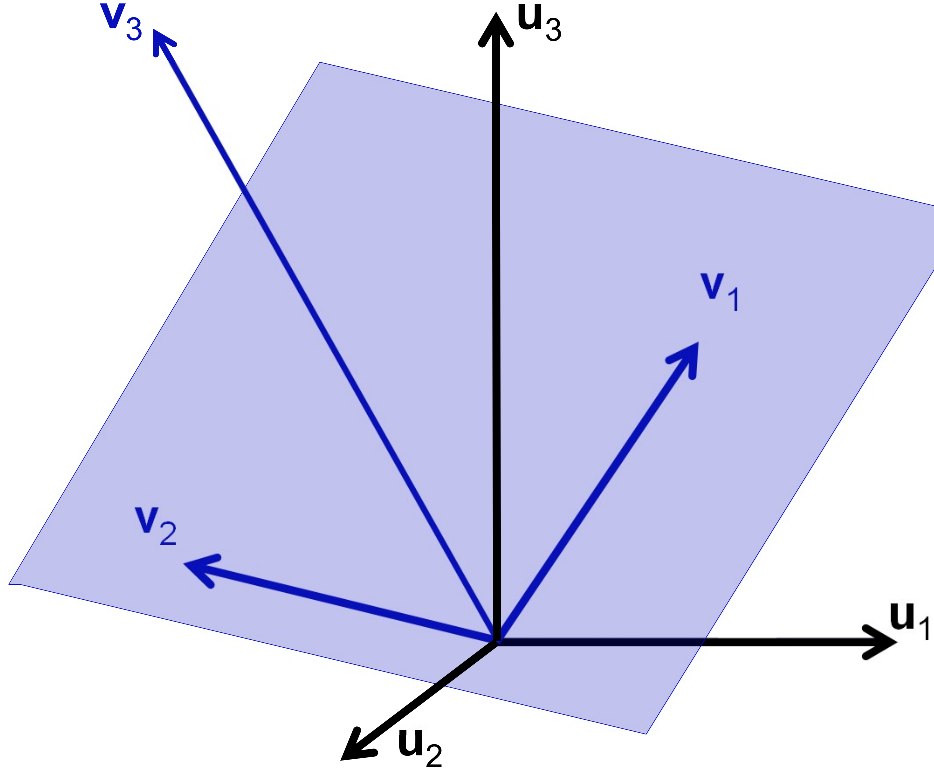
- A "good" basis captures *data* structure
- Here $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}_3$ all take large values for data in the set
- But in the ($\mathbf{v}_1$ $\mathbf{v}_2$ $\mathbf{v}_3$) set, coordinate values along $\mathbf{v}_3$ are always small for data on the blue sheet
  - $\mathbf{v}_3$ likely represents a "noise subspace" for these data

Basis based representation



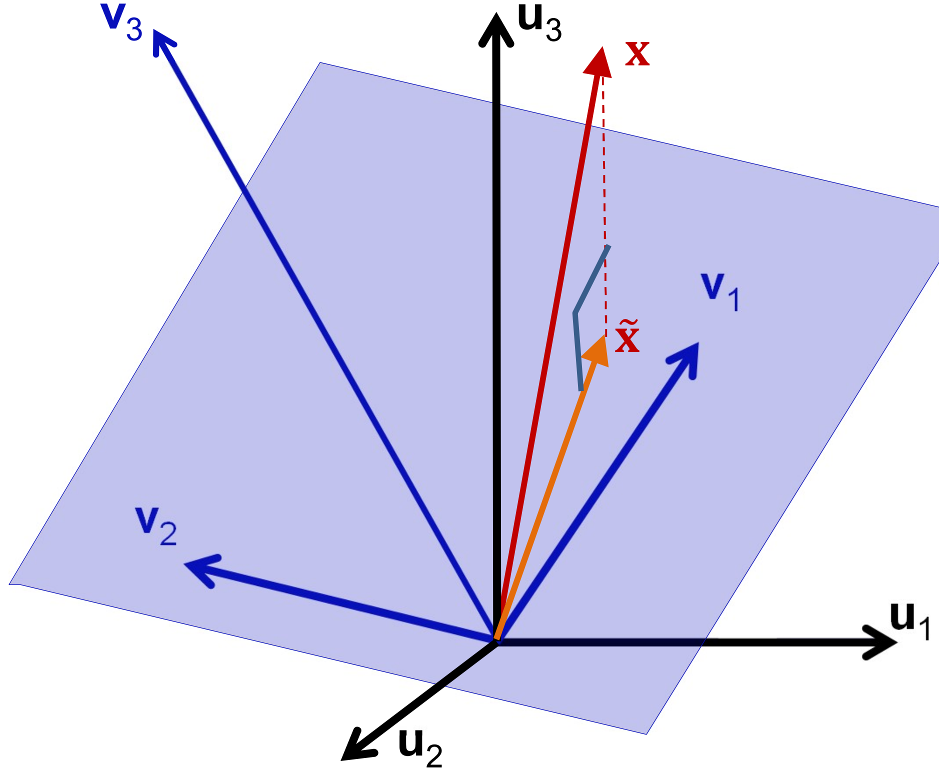
- *The most important challenge* in ML: Find the best set of bases for a given data set

Basis based representation



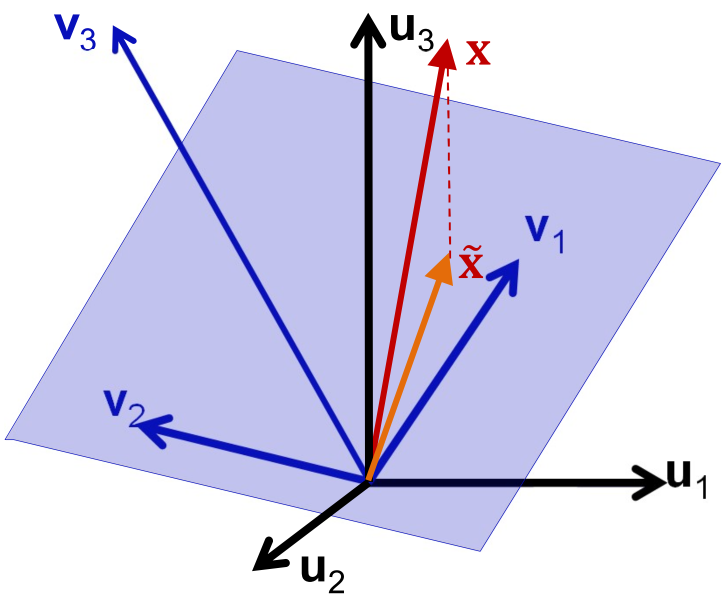
- **Modified problem:** Given the new bases v_1, v_2, v_3
  - Find best representation of every data point on v_1-v_2 plane
    - Put it on the main sheet and disregard the v3 component

Basis based representation



- **Modified problem:**
  - For any vector $\mathbf{x}$
  - Find the closest approximation $\tilde{\mathbf{x}} = a\mathbf{v}_1 + b\mathbf{v}_2$
    - Which lies entirely in the $\mathbf{v}_1$-$\mathbf{v}_2$ plane

Basis based representation



$$\mathbf{V} = [\mathbf{v}_1 \mathbf{v}_2] \qquad \mathbf{a} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\mathbf{x} \approx \mathbf{V}\mathbf{a}$$

$$\Downarrow$$

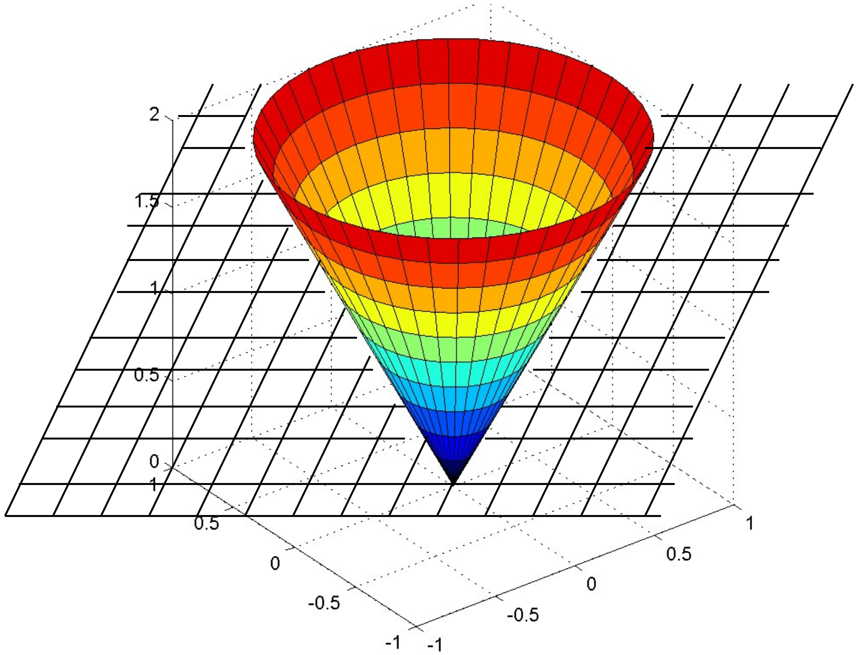
$$\mathbf{a} = \mathbf{V}^{+}\mathbf{x}$$

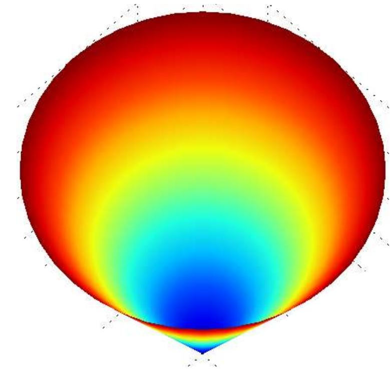
$$\Downarrow$$

$$\tilde{\mathbf{x}} = \mathbf{V}\mathbf{V}^{+}\mathbf{x}$$

- $\mathbf{P} = \mathbf{V}\mathbf{V}^{+}$ is the "projection" matrix that "projects" any vector $\mathbf{x}$ down to its "shadow" $\tilde{\mathbf{x}}$ on the $\mathbf{v}_1$-$\mathbf{v}_2$ plane
  - Expanding: $\mathbf{P} = \mathbf{V}\left(\mathbf{V}^{\mathrm{T}}\mathbf{V}\right)^{-1}\mathbf{V}^{\mathrm{T}}$

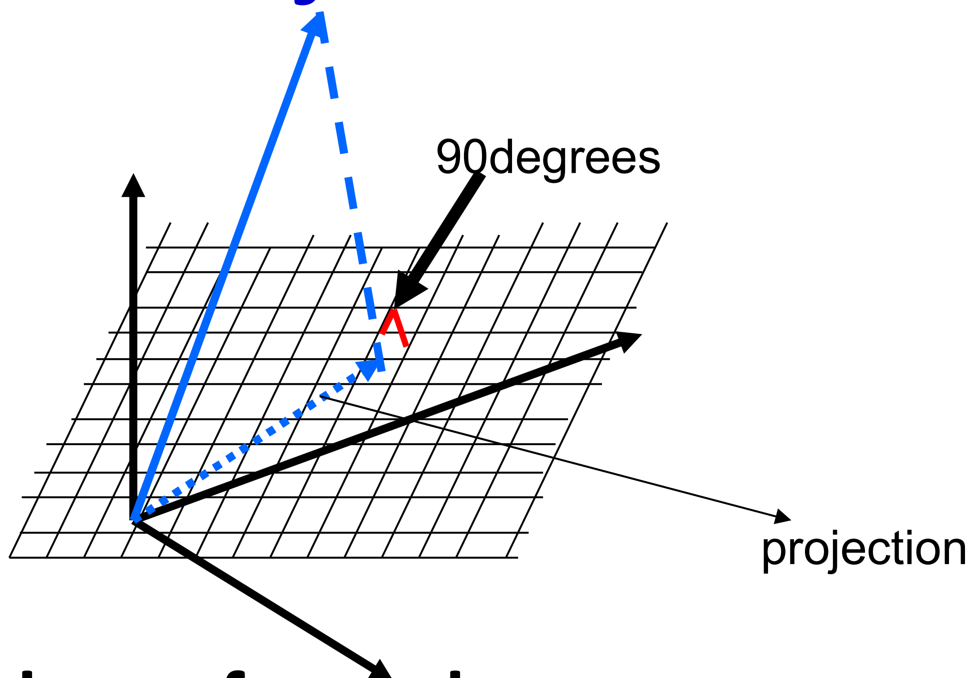
Projections onto a plane





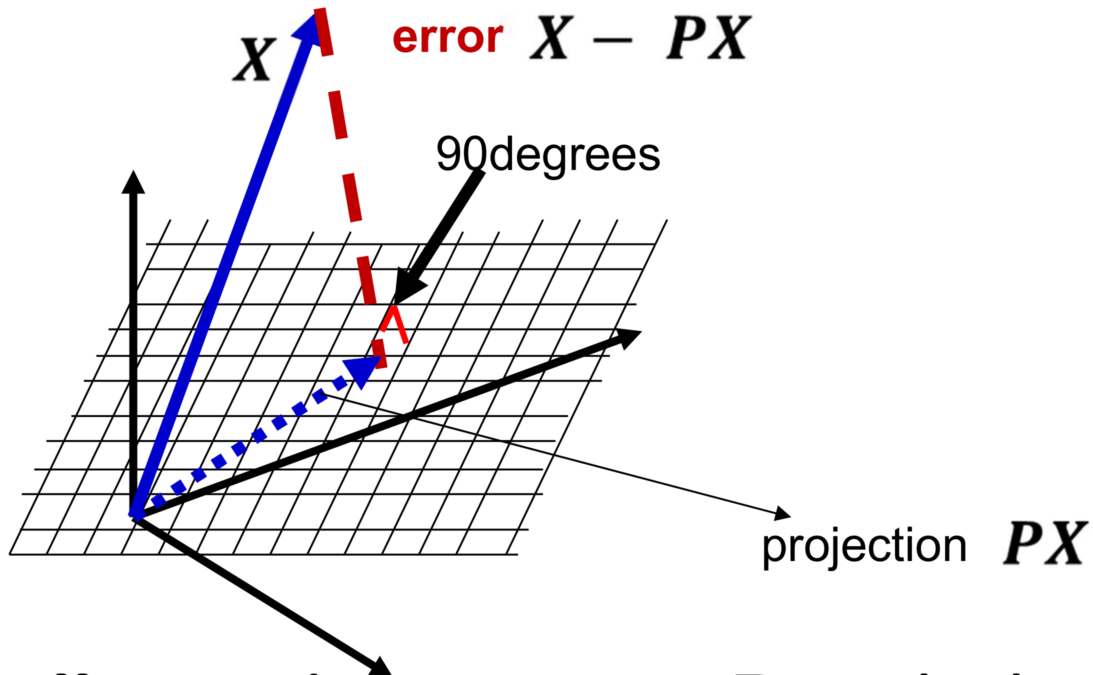
- What would we see if the cone to the left were transparent if we looked at it from above the plane shown by the grid?
  – Normal to the plane
  – Answer: the figure to the right
- How do we get this? Projection

Projections



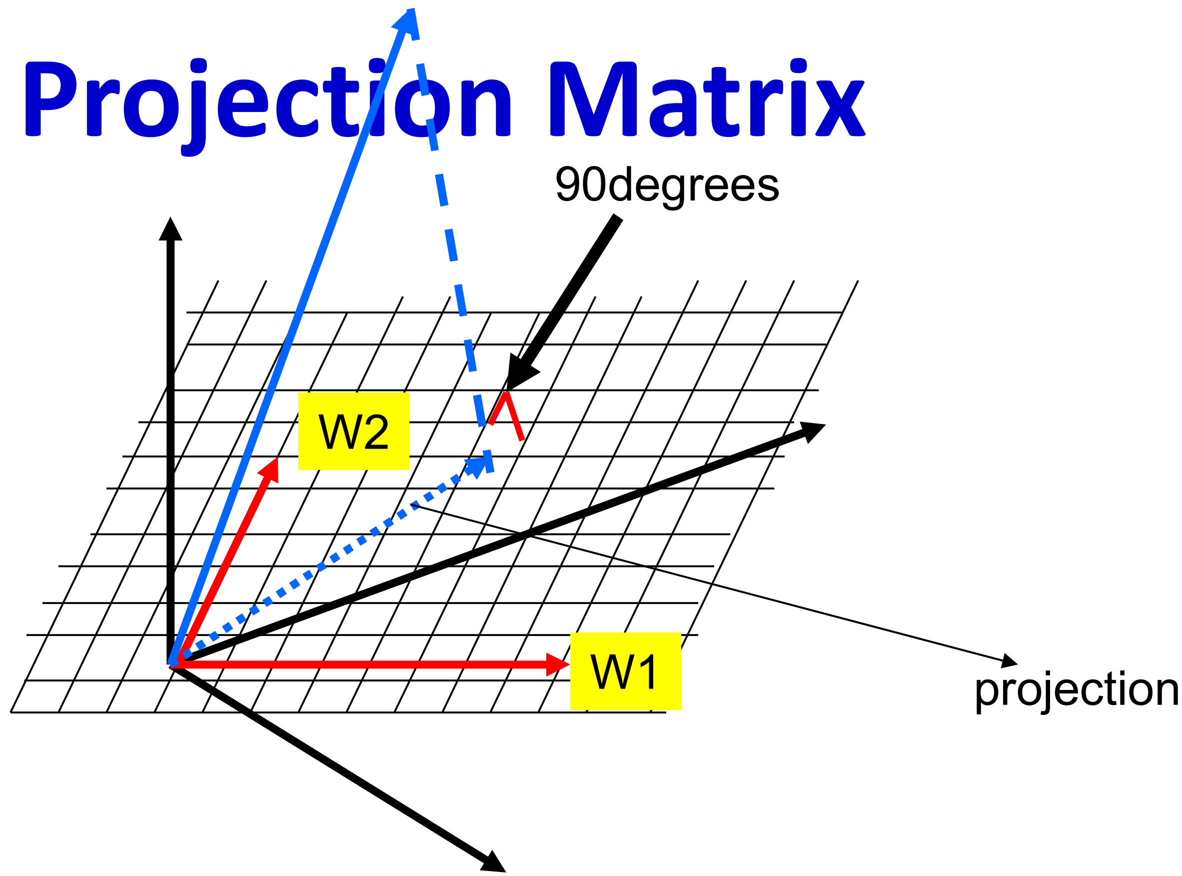
- **Actual problem: for each vector**
  - What is the corresponding vector on the plane that is “closest approximation” to it?
  - What is the *transform* that converts the vector to its approximation on the plane?

Projections



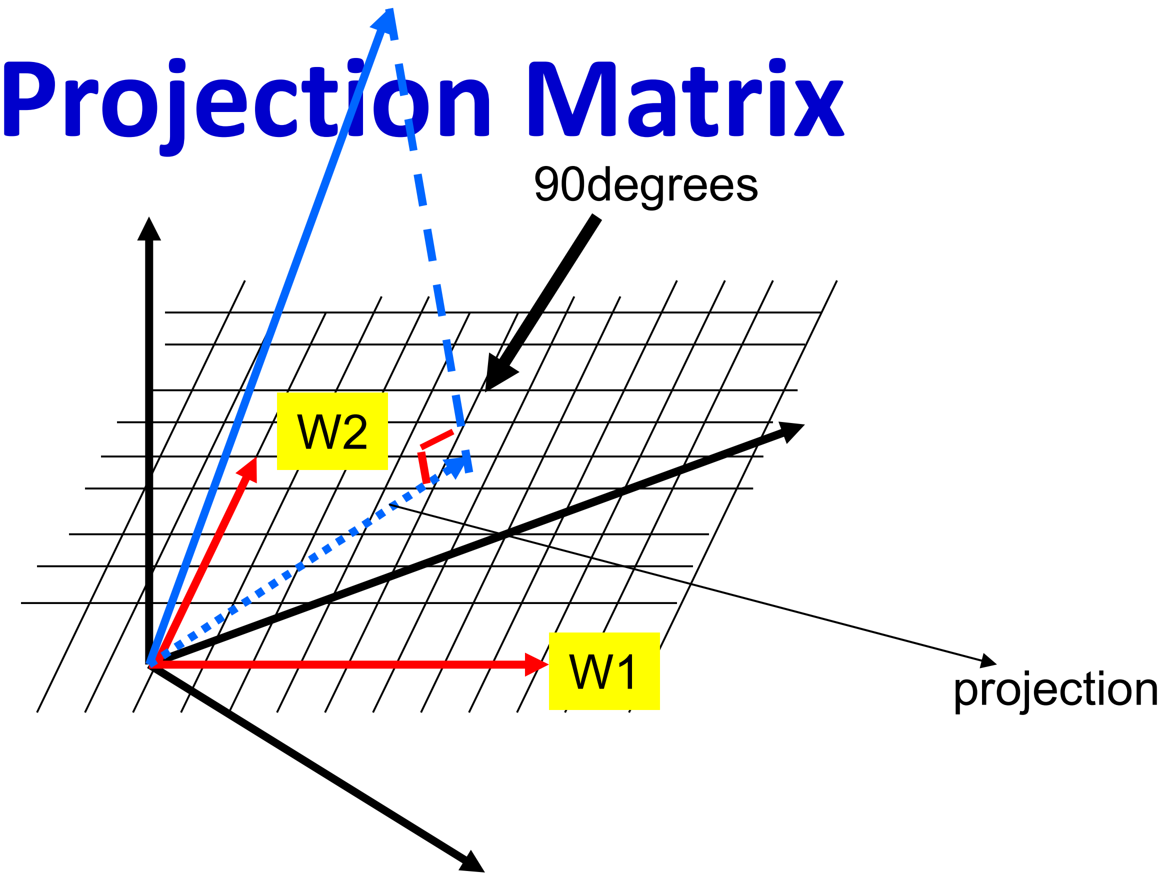
- **Arithmetically:** Find the matrix $\boldsymbol{P}$ such that
  - For **every** vector $\boldsymbol{X}$, $\boldsymbol{PX}$ lies on the plane
    - The plane is the column space of $\boldsymbol{P}$
  - $||X - \boldsymbol{P}X||^2$ is the smallest possible

Projection Matrix



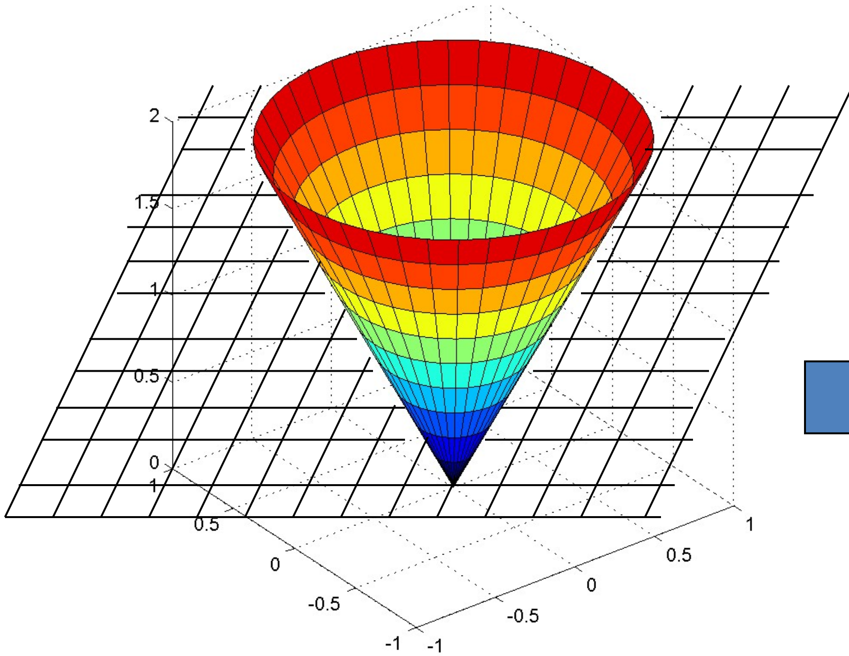
- Consider any set of *independent* vectors (bases) $\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots$ on the plane
  - Arranged as a matrix $[\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots]$
  - The plane is the *column space* of the matrix
- Find the projection matrix $\boldsymbol{P}$ that projects on to the plane formed from $[\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots]$

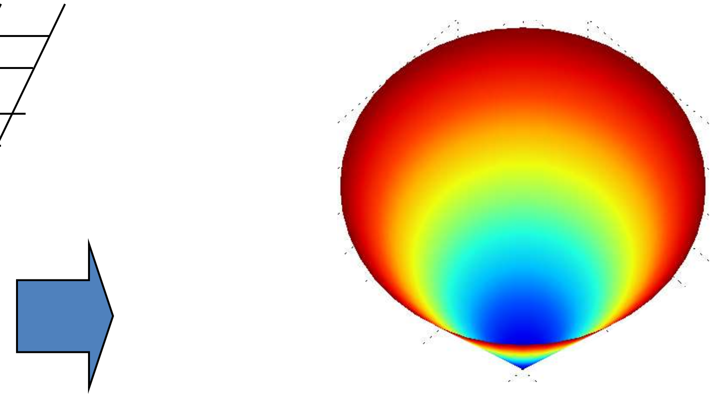
Projection Matrix



- Given a set of vectors $\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots$ which form a matrix $\boldsymbol{W} = [\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots]$
- The projection matrix to transform a vector $\boldsymbol{X}$ to its projection on the plane is
  - $\boldsymbol{P} = \boldsymbol{Pinv}(\boldsymbol{W})\boldsymbol{W}^T$
  - $\ldots = \boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{W}^T$

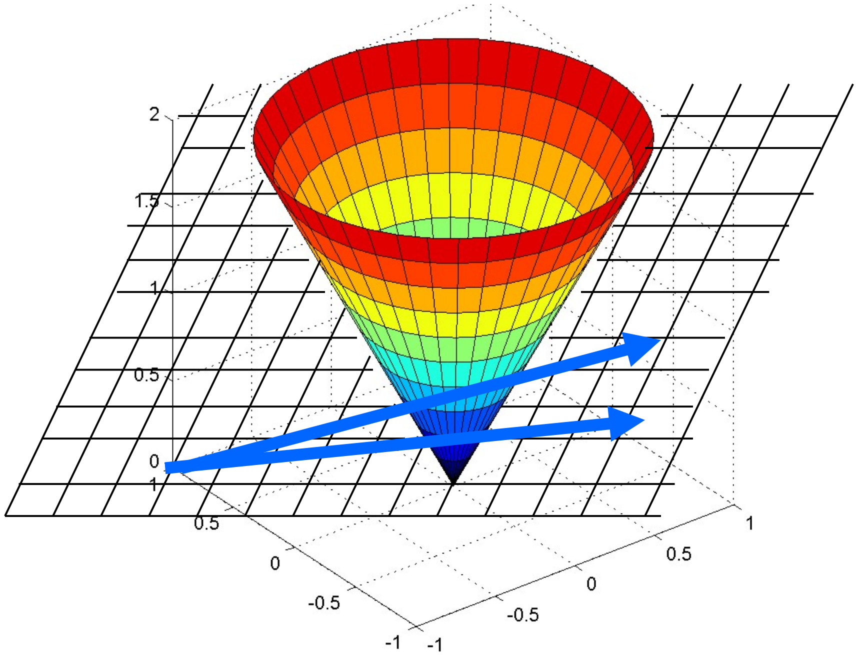
Projections

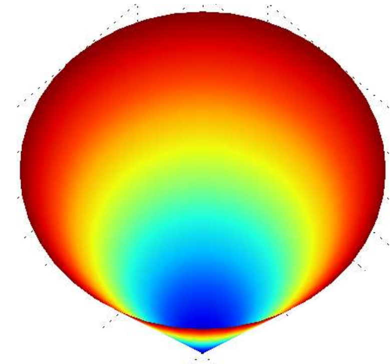




- HOW?

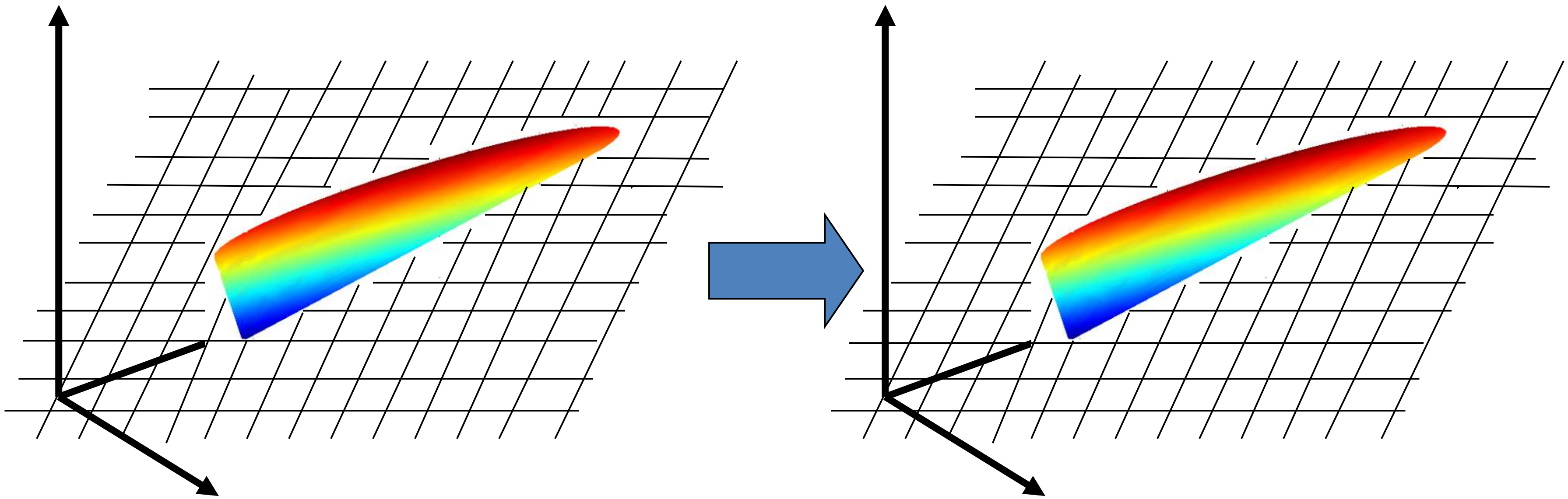
Projections





- Draw any two vectors W_1 and W_1 W_2 that lie on the plane
  - ***ANY two*** **so long as they have different angles**
- Compose a matrix $\mathbf{W} = [W_1\ W_2..\,]$
- Compose the projection matrix $\mathbf{P} = \mathbf{W}\ (\mathbf{W}^T\mathbf{W})^{-1}\ \mathbf{W}^T$
- Multiply every point on the cone by $\mathbf{P}$ to get its projection

Projection matrix properties



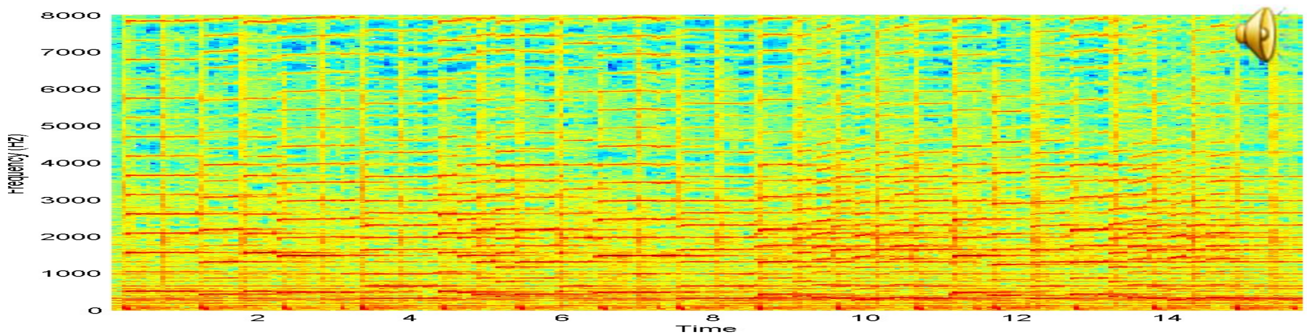
- The projection of any vector that is already on the plane is the vector itself
  - $\mathbf{PX} = \mathbf{X}$ if $\mathbf{X}$ is on the plane
  - If the object is already on the plane, there is no further projection to be performed
- The projection of a projection is the projection
  - $\mathbf{P}(\mathbf{PX}) = \mathbf{PX}$
- Projection matrices are *idempotent*
  - $\mathbf{P}^2 = \mathbf{P}$

Projections: A more physical meaning

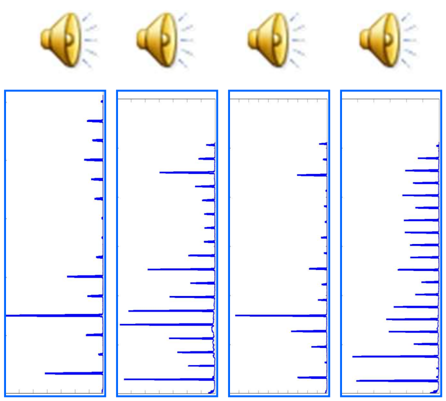
- Let $\mathbf{W_1}$, $\mathbf{W_2}$ .. $\mathbf{W_k}$ be "bases"
- We want to explain our data in terms of these "bases"
  - We often cannot do so
  - But we can explain a significant portion of it

- The portion of the data that can be expressed in terms of our vectors $\mathbf{W_1}$, $\mathbf{W_2}$, ... $\mathbf{W_k}$, is the projection of the data on the $\mathbf{W_1}$ … $\mathbf{W_k}$ (hyper) plane
  - In our previous example, the "data" were all the points on a cone, and the bases were vectors on the plane

Projection : an example with sounds



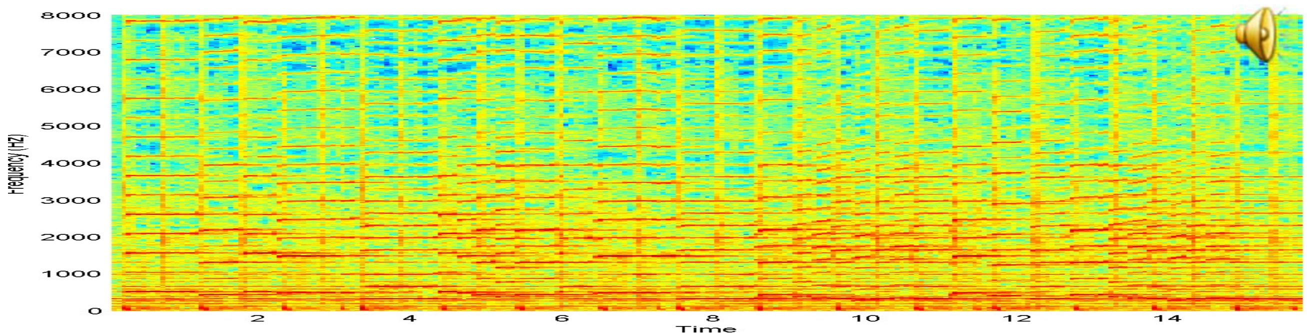
- The spectrogram (matrix) of a piece of music



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

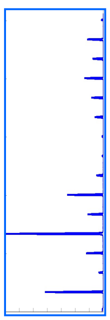
Projection: one note

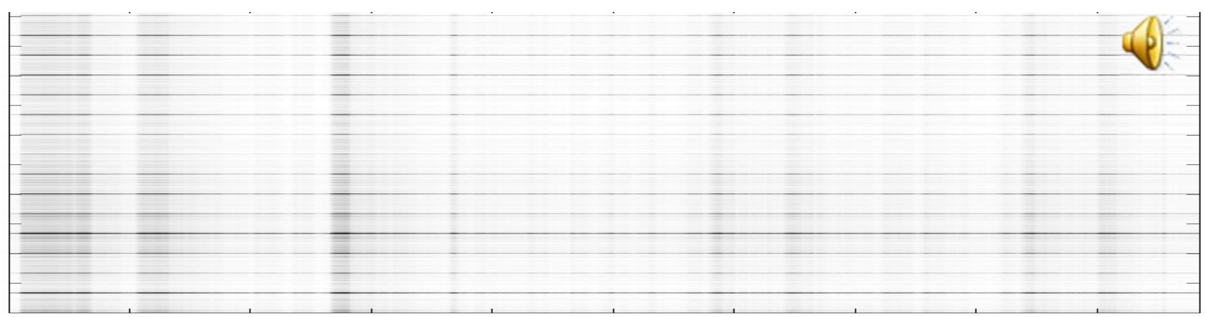
M =



- The spectrogram (matrix) of a piece of music

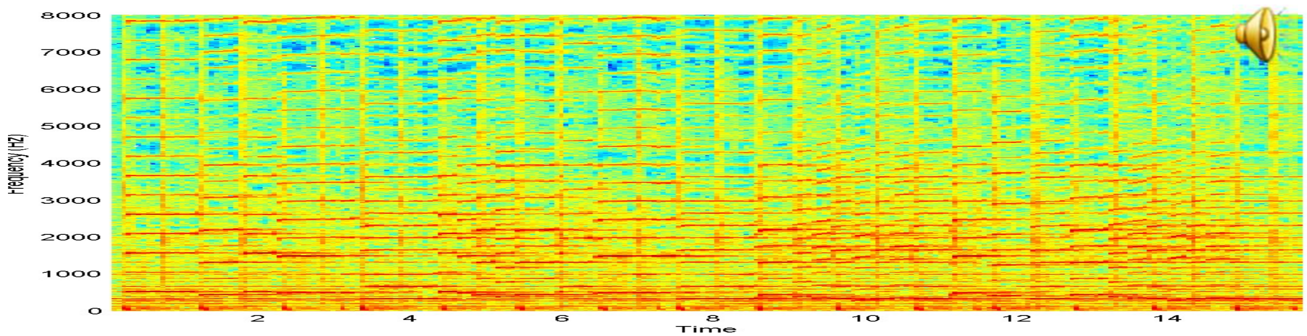
W =



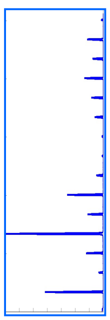


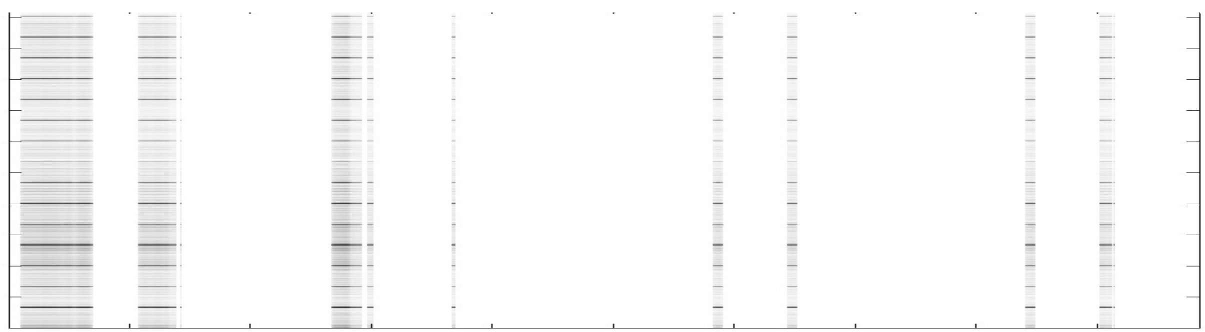
- M = spectrogram; W = note
- $P = W(W^T W)^{-1} W^T$
- Projected Spectrogram = PM

Projection: one note – cleaned up

M = 

- The spectrogram (matrix) of a piece of music

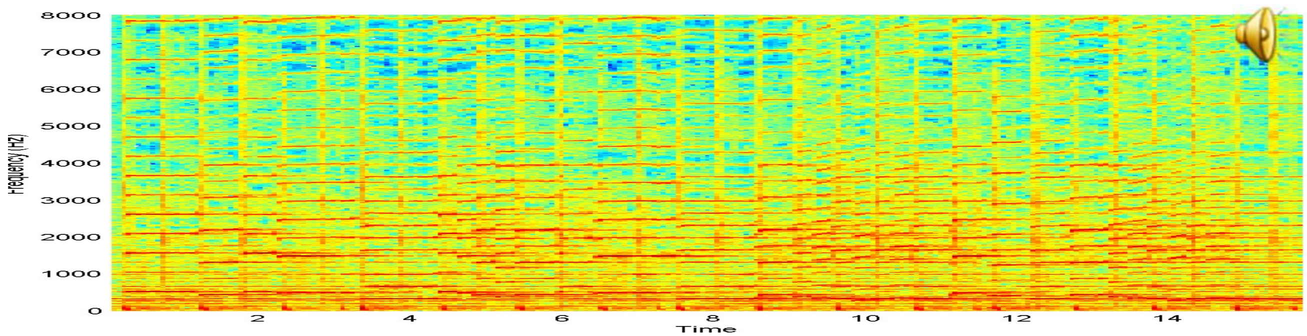
W = 



- Floored all matrix values below a threshold to zero

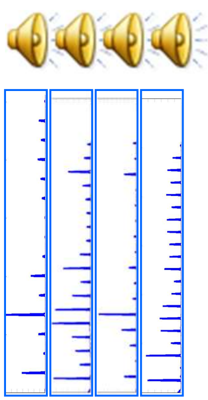
Projection: multiple notes

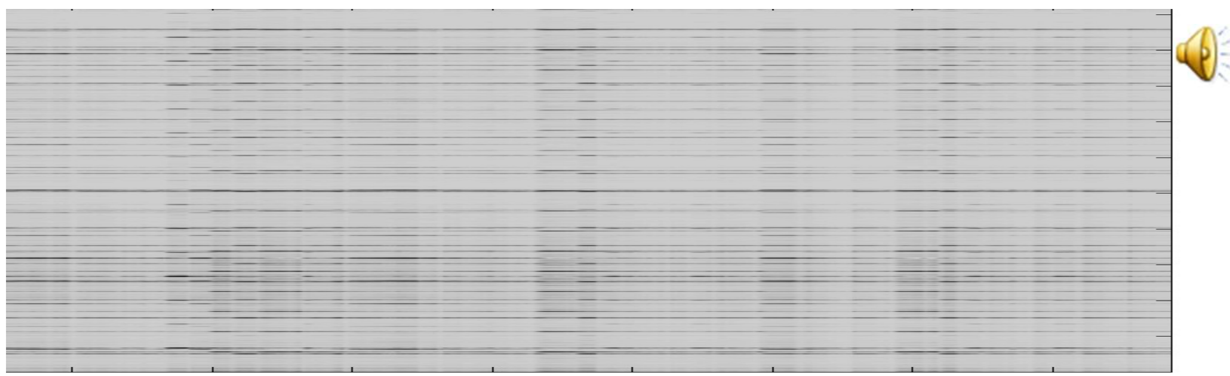
M =



- The spectrogram (matrix) of a piece of music

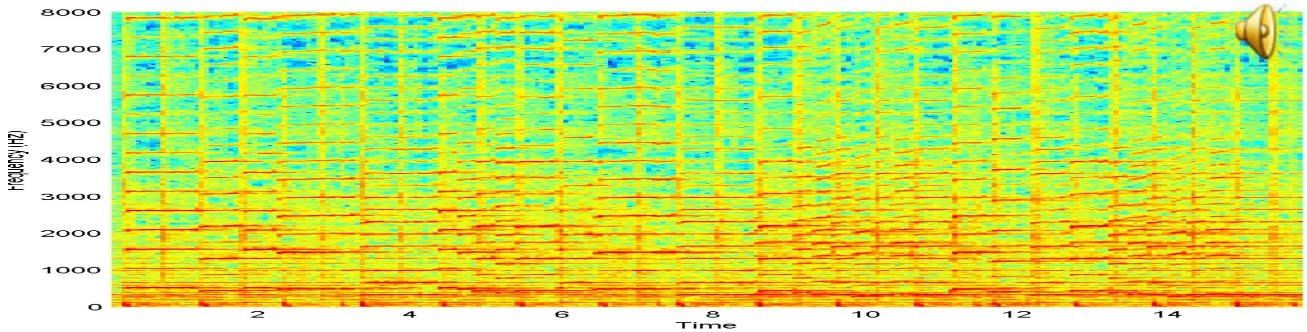
W =



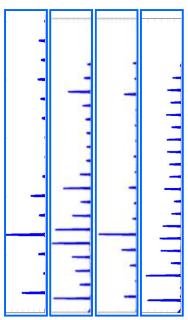


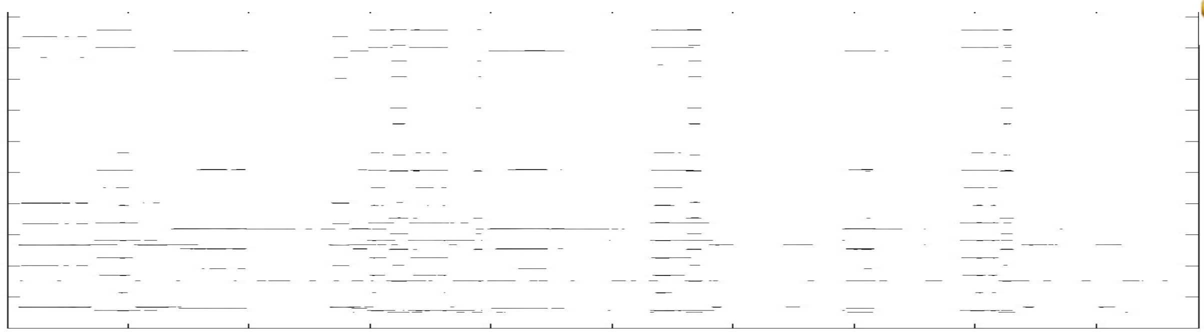
- $P = W (W^T W)^{-1} W^T$
- Projected Spectrogram = P * M

Projection: multiple notes, cleaned up

M = 

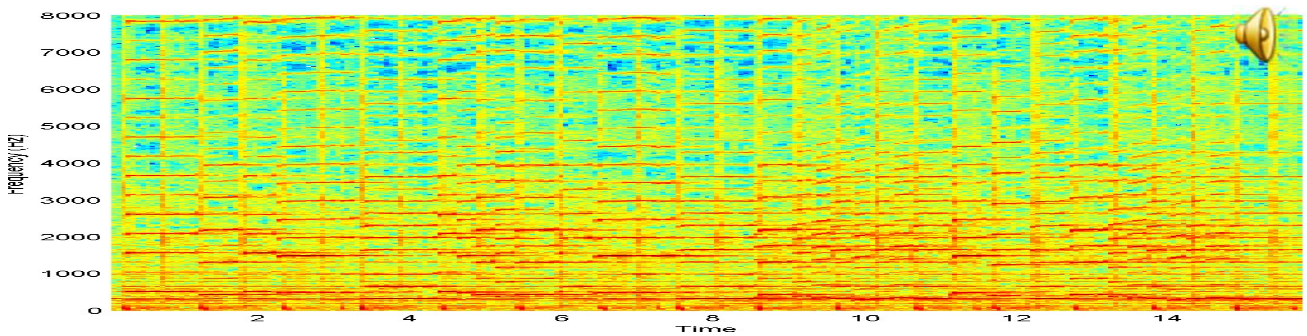
- The spectrogram (matrix) of a piece of music

W = 



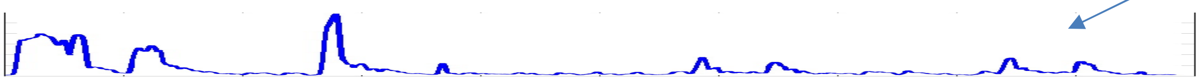
- $\boldsymbol{P} = \boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{W}^T$
- Projected Spectrogram = $\boldsymbol{PM}$

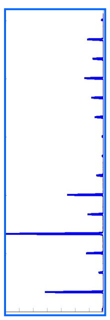
Projection: one note

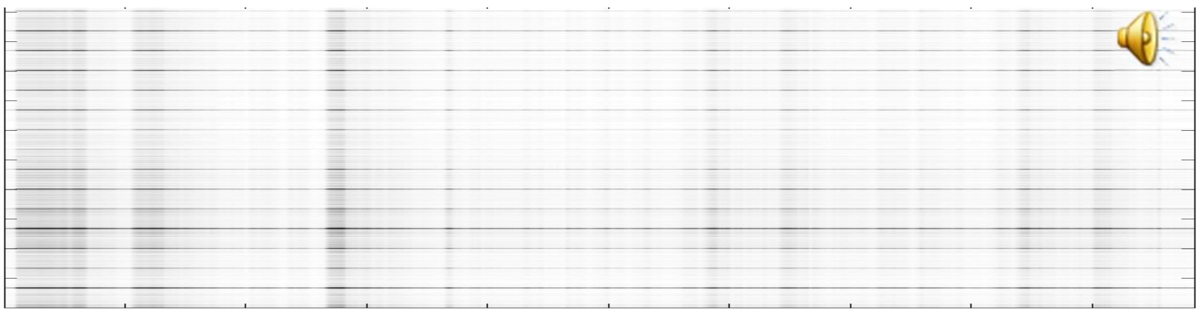
M = 

- The spectrogram (matrix) of a piece of music

$T = W^{+}M$



W = 



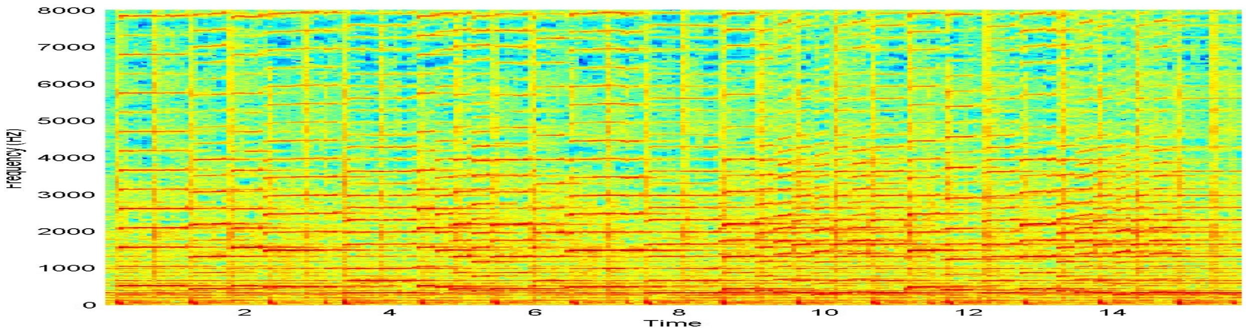
- The "transcription" of the note is

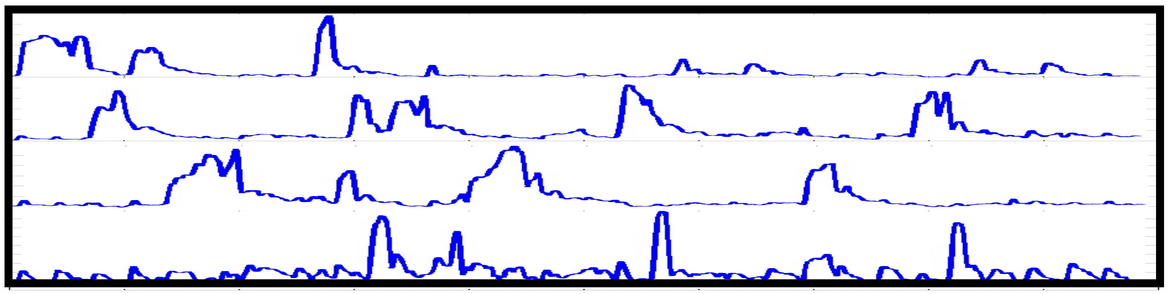
$$T = W^{+}M = (W^{T}W)^{-1}W^{T}M$$

- Projected Spectrogram = $WT = PM$

Explanation with multiple notes

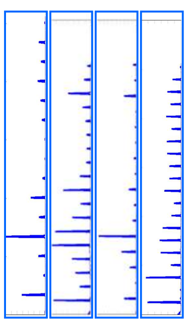
M =

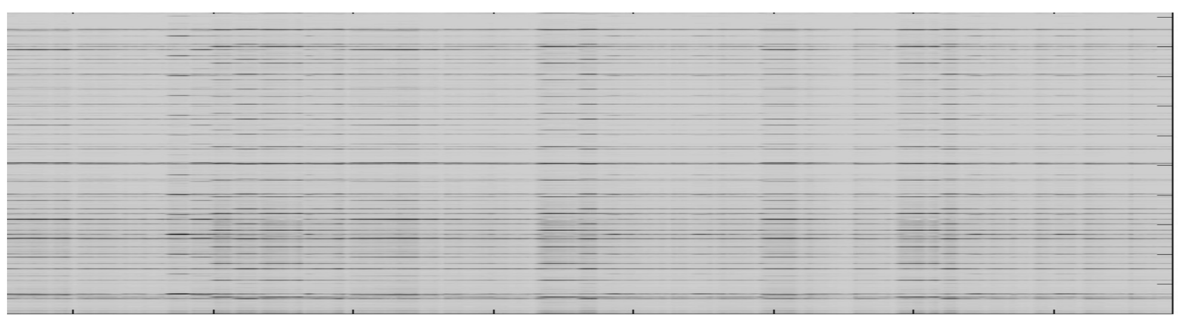




$T = W^{+}M$

W =





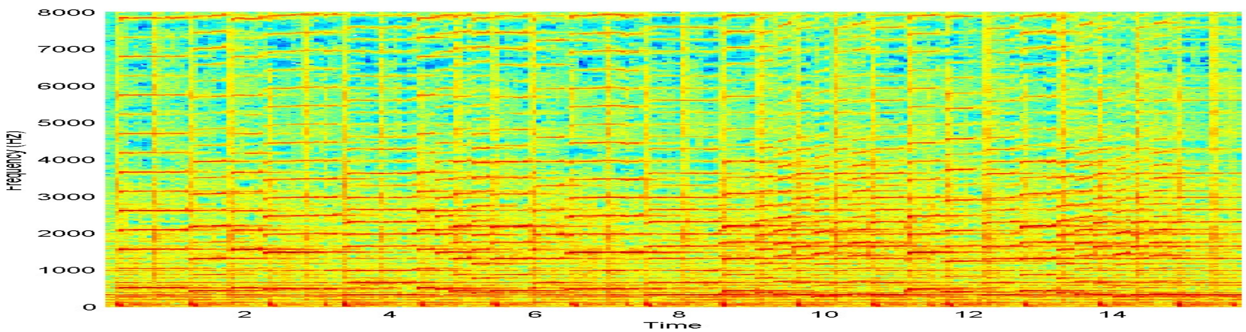
- The "transcription" of the set of notes is

$$T = W^{+}M = (W^{T}W)^{-1}W^{T}M$$

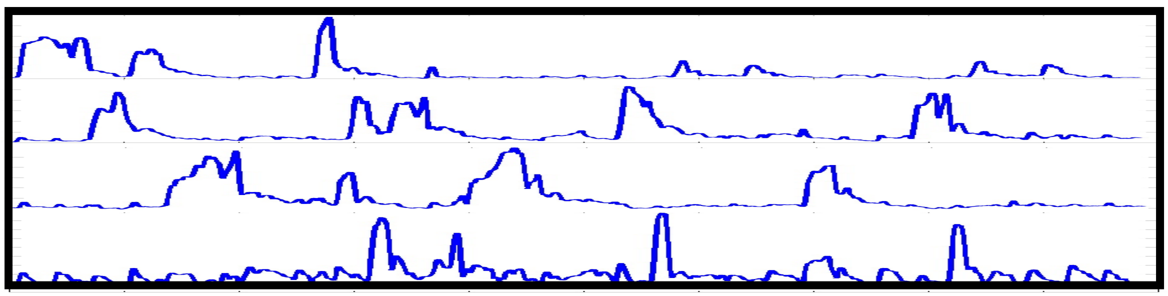
- Projected Spectrogram = $WT = PM$

How about the other way?

M =



T =



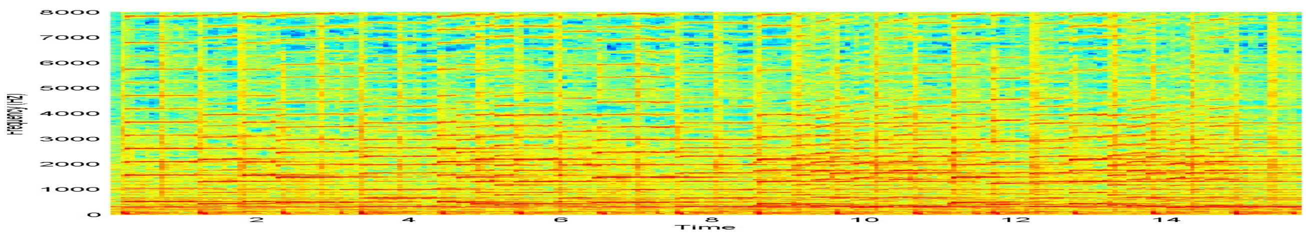
W = ?

U = ?

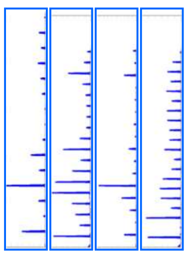
- $WT \approx M$ $\quad W = M\,Pinv(T)$ $\quad U = WT$

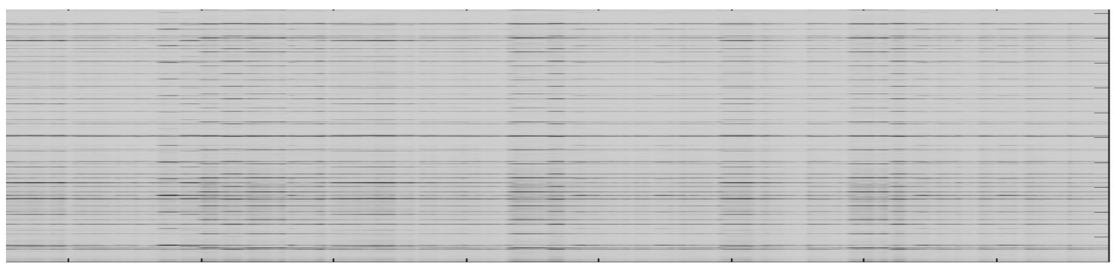
Projections are often examples of rank-deficient transforms

M =



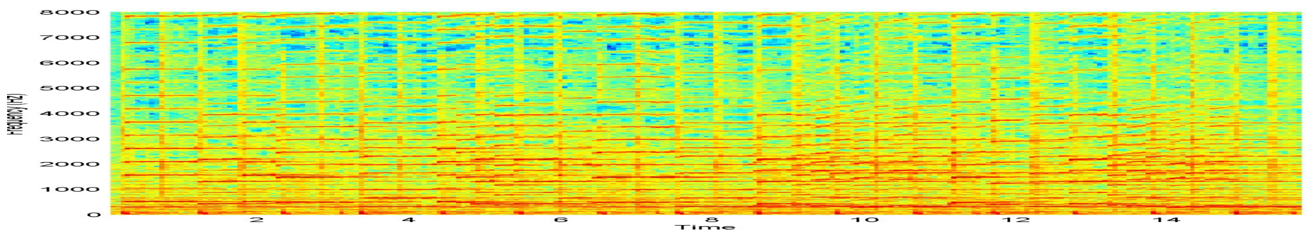
W =

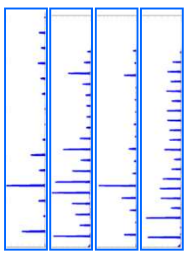


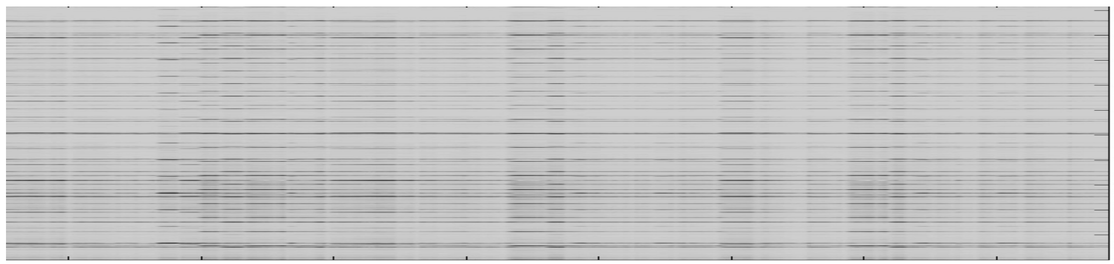


- $P = W(W^T W)^{-1} W^T$; Projected Spectrogram : $M_{proj} = PM$
- The original spectrogram can never be recovered
  - P is rank deficient
- P explains all vectors in the new spectrogram as a mixture of only the 4 vectors in W
  - There are only a maximum of 4 ***linearly independent*** bases
  - Rank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

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!

Pseudo-inverse (PINV)

- $Pinv()$ applies to non-square matrices and non-invertible square matrices
- $Pinv\ (Pinv(\mathbf{A}))\ =\ \mathbf{A}$
- $\mathbf{A}Pinv(\mathbf{A})$= projection matrix!
  - Projection onto the columns of $\mathbf{A}$

- If $\mathbf{A}$ is a $K \times N$ matrix and $K > N$, $\mathbf{A}$ projects N-dimensional vectors into a higher-dimensional K-dimensional space
  - $Pinv(\mathbf{A})$ is a $N \times K$ matrix
  - $Pinv(\mathbf{A})\mathbf{A} = \mathbf{I}$ in this case
- Otherwise $\mathbf{A}Pinv(\mathbf{A}) = \mathbf{I}$

Poll 4

Poll 4

- The projection of a data onto a set of bases recovers the portion of the data that can be “explained” by those bases (T or F)
  - **T**
  - F

- For an NxM matrix T such that N > M (a tall matrix), T*pinv(T) is a projection matrix that projects vectors on the columns of T
  - **T**
  - F

Overview

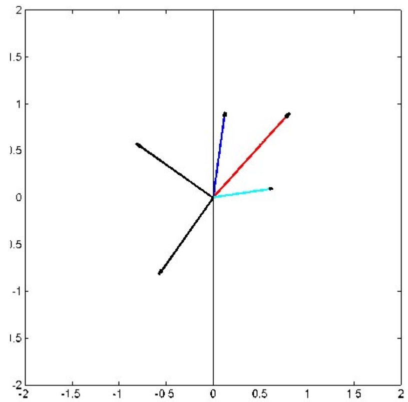
- Vectors and matrices
- Basic vector/matrix operations
- Various matrix types
- Matrix properties
  - Determinant
  - Inverse
  - Rank
- Solving simultaneous equations
- Projections
- **Eigen decomposition**
- **SVD**

Eigenanalysis

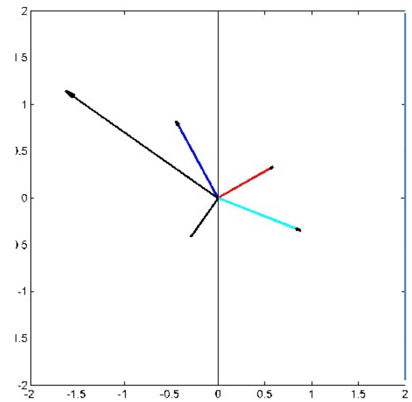
- If something can go through a process mostly unscathed in character it is an *eigen*-something
  - Sound example:
- A vector that can undergo a matrix multiplication and keep pointing the same way is an *eigenvector*
  - Its length can change though
- How much its length changes is expressed by its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

EigenVectors and EigenValues

Black vectors are eigen vectors



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

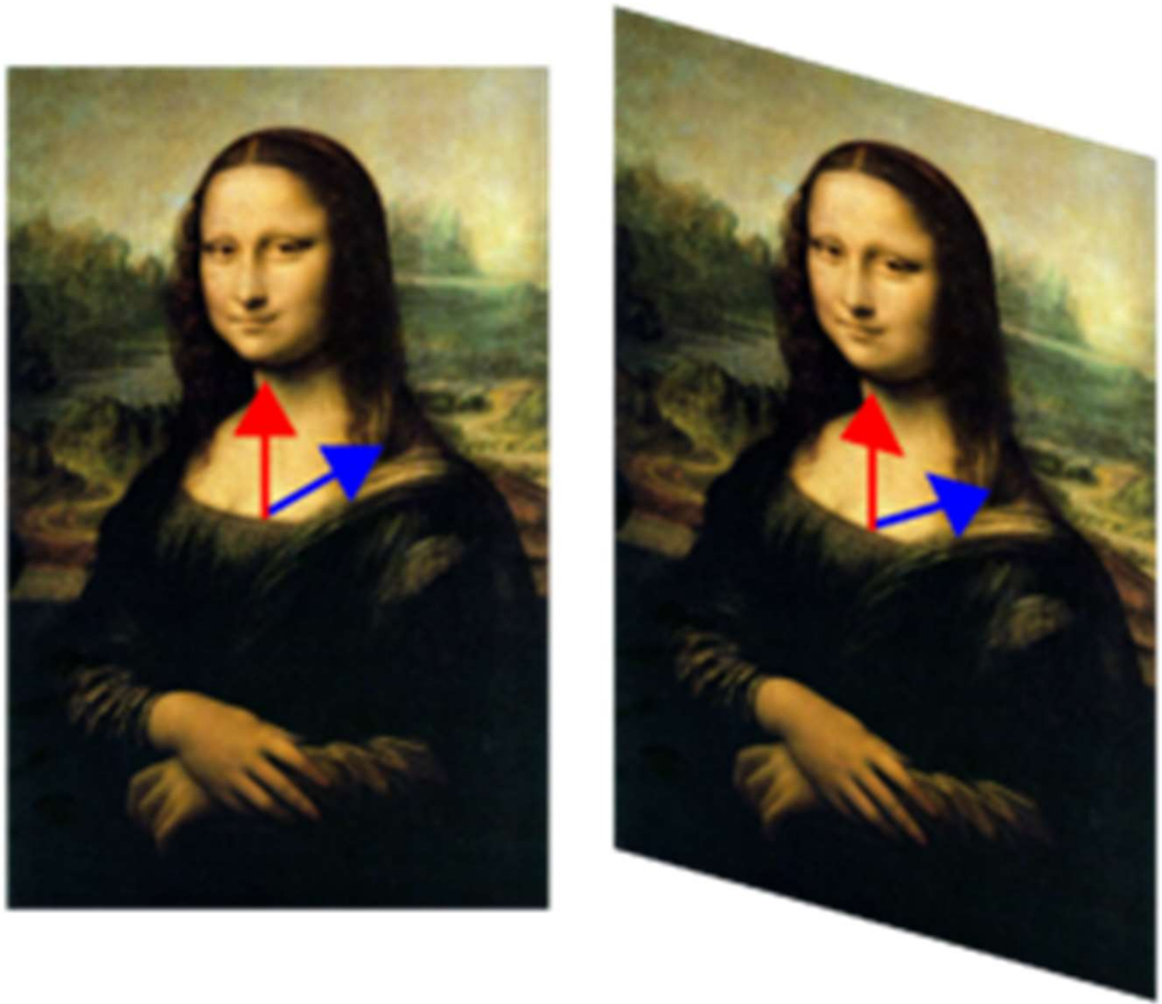


- Vectors that do not change angle upon transformation
  - They may change length

$$MV = \lambda V$$

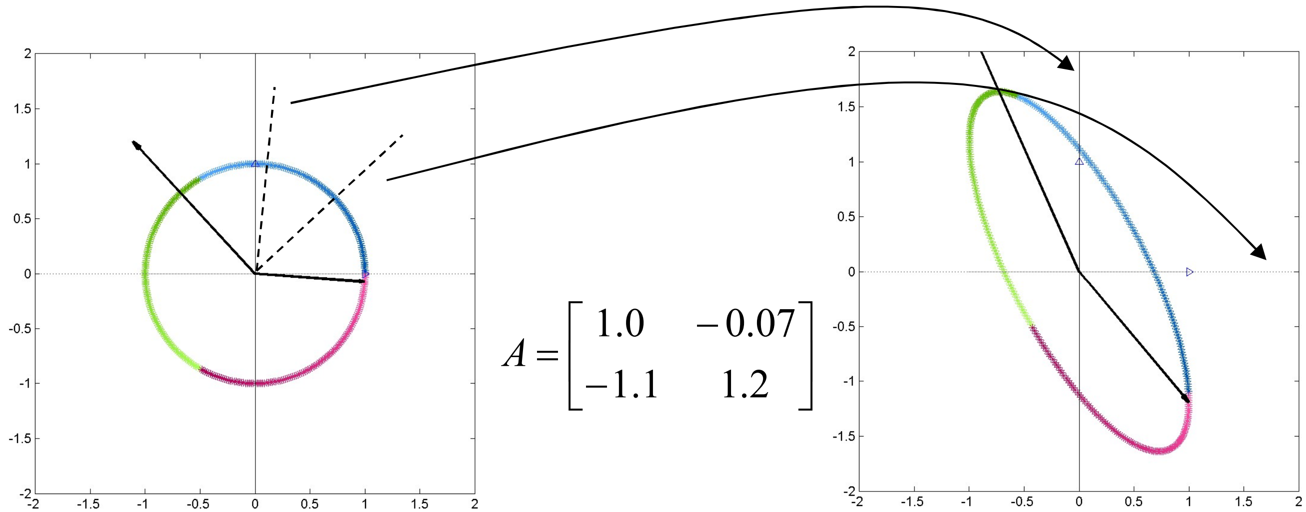
  - V = eigen vector
  - λ = eigen value

Eigen vector example



Shear transforms are “defective” matrices

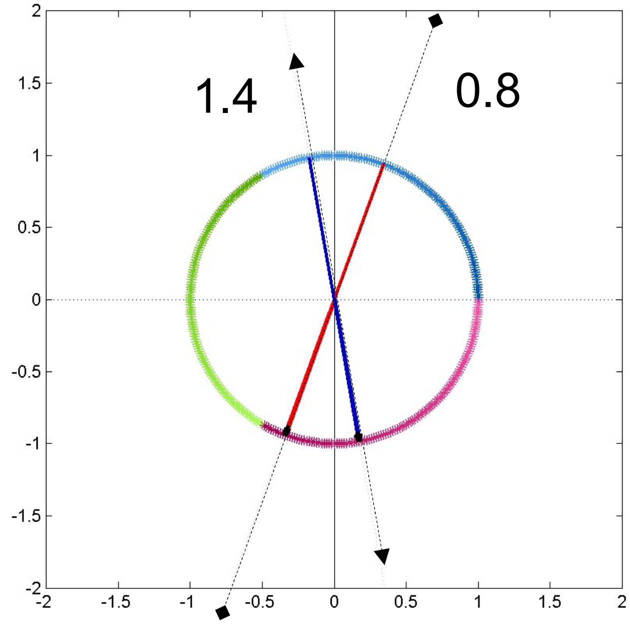
Matrix multiplication revisited



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

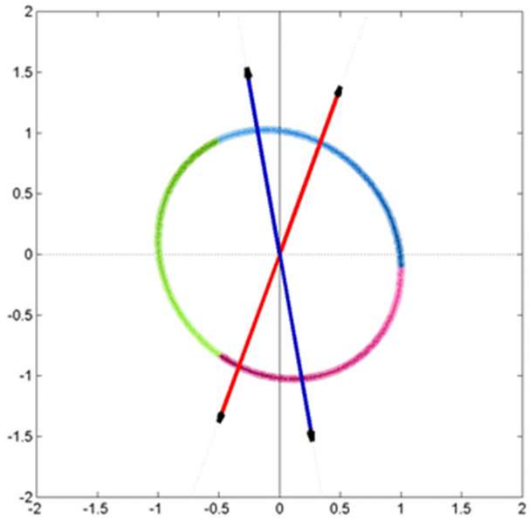
- Matrix transformation "transforms" the space
  - Warps the paper so that the normals to the two vectors now lie along the axes

A stretching operation



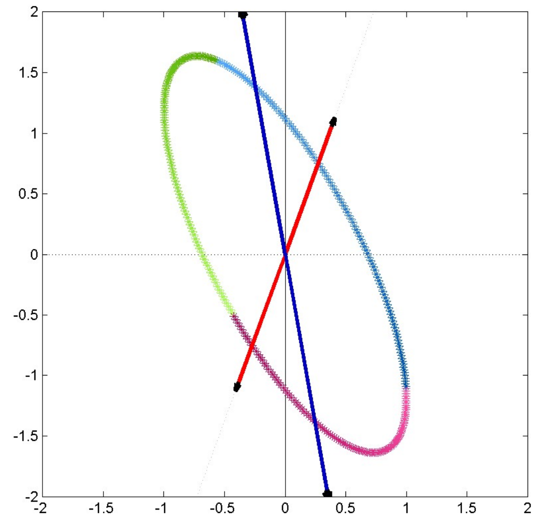
- Draw two lines
- Stretch / shrink the paper along these lines by factors λ_1 and λ_2
  - The factors could be negative – implies flipping the paper
- The result is a transformation of the space

A stretching operation



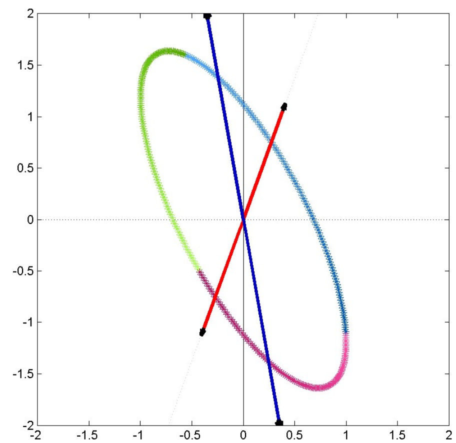
- Draw two lines
- Stretch / shrink the paper along these lines by factors λ_1 and λ_2
  - The factors could be negative – implies flipping the paper
- The result is a transformation of the space

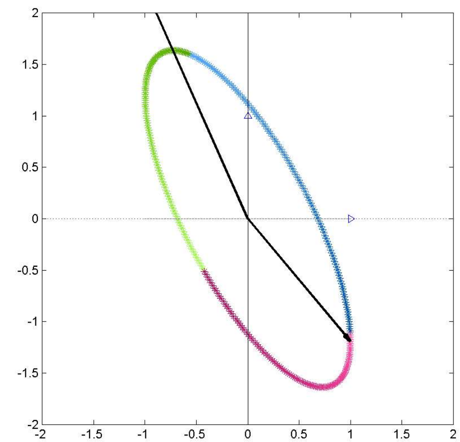
A stretching operation



- 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





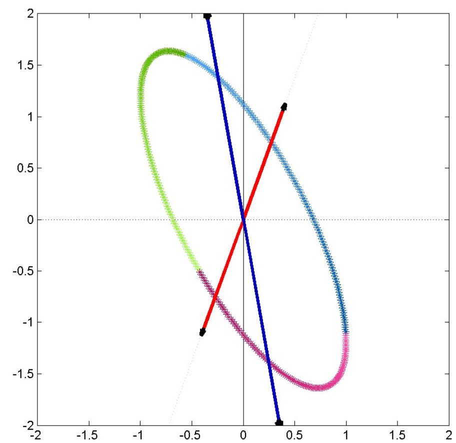
- The result of the stretching is exactly the same as transformation by a matrix
- The axes of stretching/shrinking are the eigenvectors
  - The degree of stretching/shrinking are the corresponding eigenvalues
- The EigenVectors and EigenValues convey all the information about the matrix

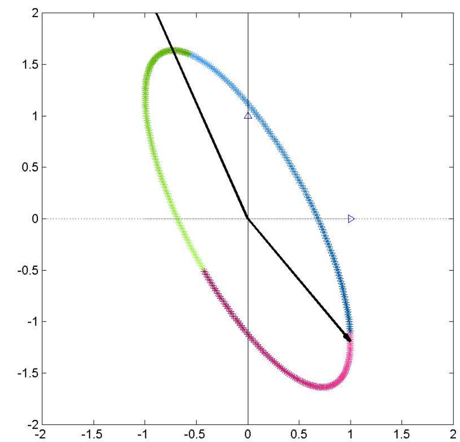
Physical interpretation of eigen vector

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

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

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





- 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
- The determinant of the matrix is the product of the eigenvalues

$$|M| = |V||\Lambda||V^{-1}| = C.\prod_i \lambda_i \,.\, C^{-1} = \prod_i \lambda_i$$

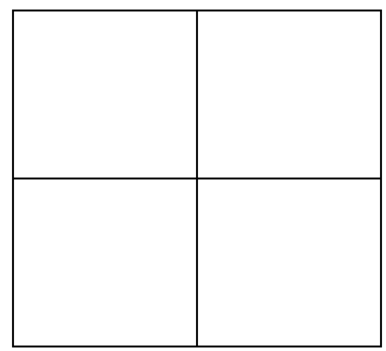
Eigen Analysis

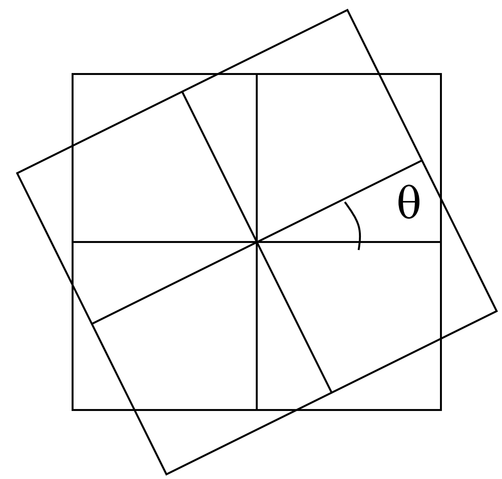
- Not all square matrices have nice eigen values and vectors
  - E.g. consider a rotation matrix

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

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

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

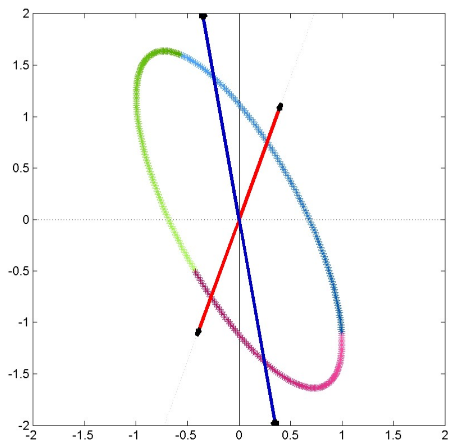




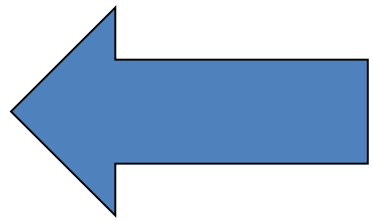
  - This rotates every vector in the plane
    - No vector that remains unchanged

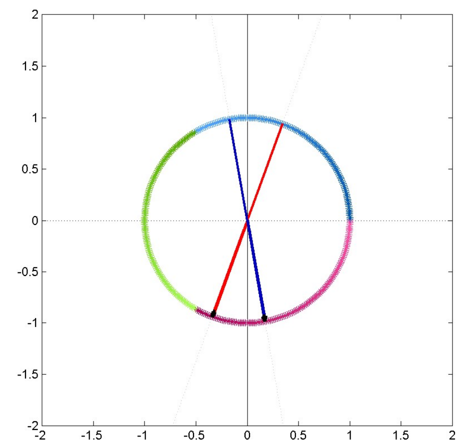
- In these cases the Eigen vectors and values are complex
  - Actually complex conjugate pairs

Singular Value Decomposition



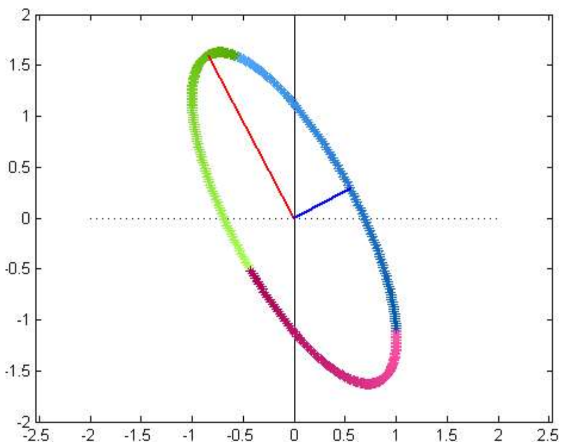
$$A = \begin{bmatrix} 1.0 & -0.07 \\ -1.1 & 1.2 \end{bmatrix}$$



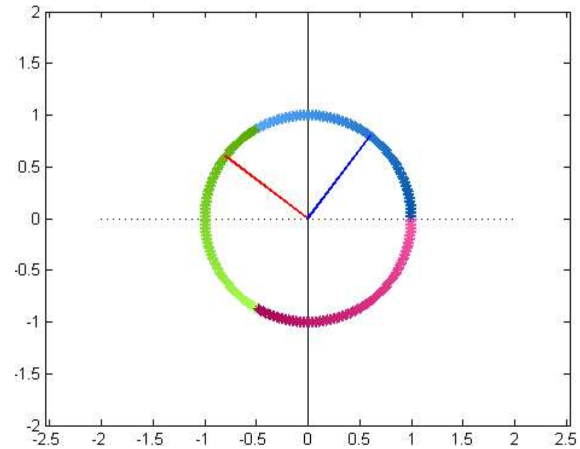


- Matrix transformations convert circles to ellipses
- Eigen vectors are vectors that do not change direction in the process
- There is another key feature of the ellipse to the left that carries information about the transform
  - Can you identify it?

Singular Value Decomposition

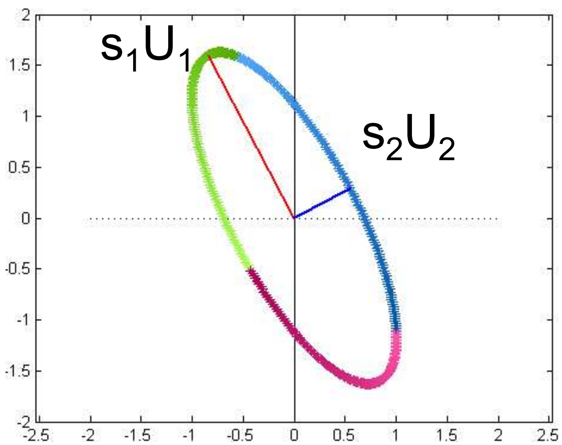


$$A = \begin{bmatrix} 1.0 & -0.07 \\ -1.1 & 1.2 \end{bmatrix}$$



- 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!

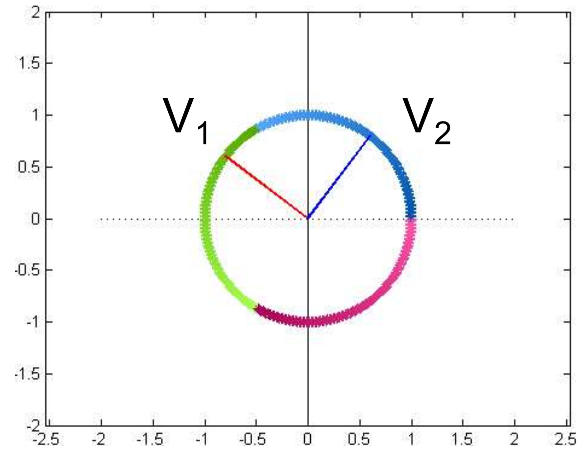
Singular Value Decomposition



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

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

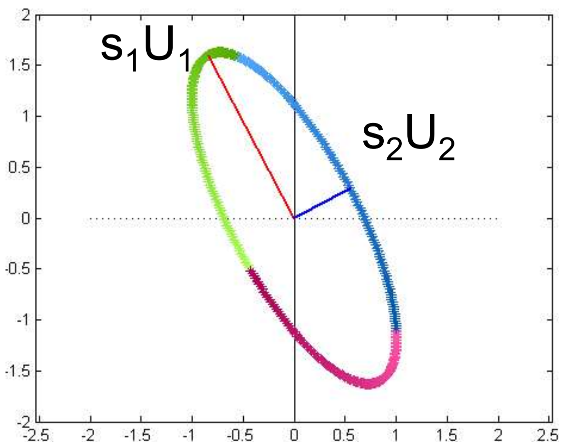
matlab:
[U,S,V] = svd(A)



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

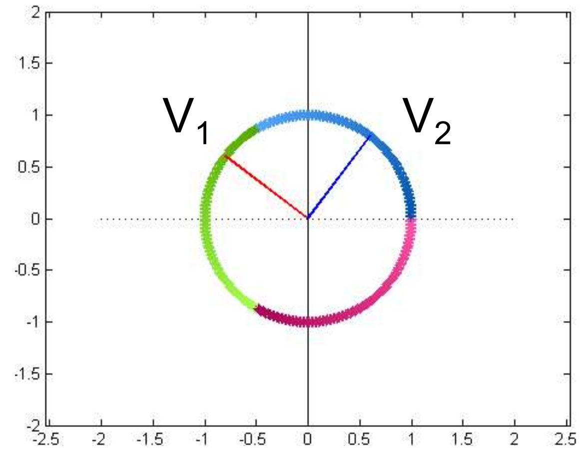
- The *right singular vectors* in V are transformed to the *left singular vectors* in U
  - And scaled by the *singular values* that are the diagonal entries of S

Singular Value Decomposition



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

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



- A matrix $\boldsymbol{A}$ converts *right* singular vectors $\boldsymbol{V}$ to *left* singular vectors $\boldsymbol{U}$
- $\boldsymbol{A}^{\mathrm{T}}$ converts $\boldsymbol{U}$ to $\boldsymbol{V}$

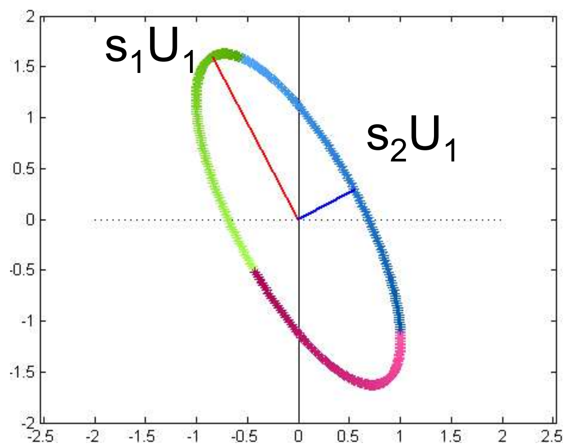
Singular Value Decomposition

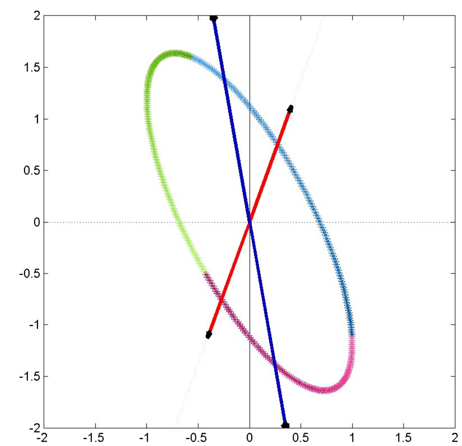
- The left and right singular vectors are not the same
  – If A is not a square matrix, the left and right singular vectors will be of different dimensions

- 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

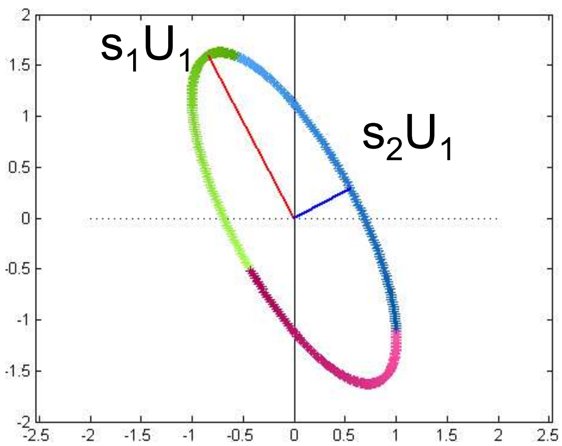


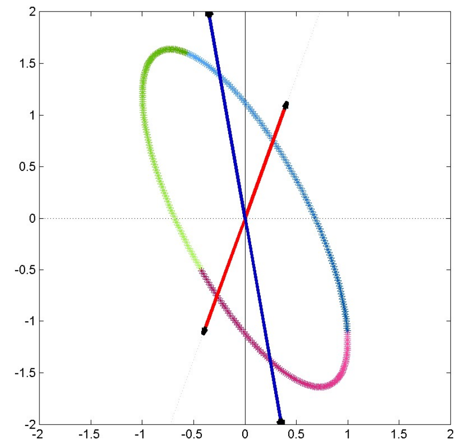


- Square matrices: product of singular values = determinant of the matrix
  - This is also the product of the *eigen* values
  - I.e. there are two different sets of axes whose products give you the area of an ellipse

- 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

SVD vs. Eigen Analysis



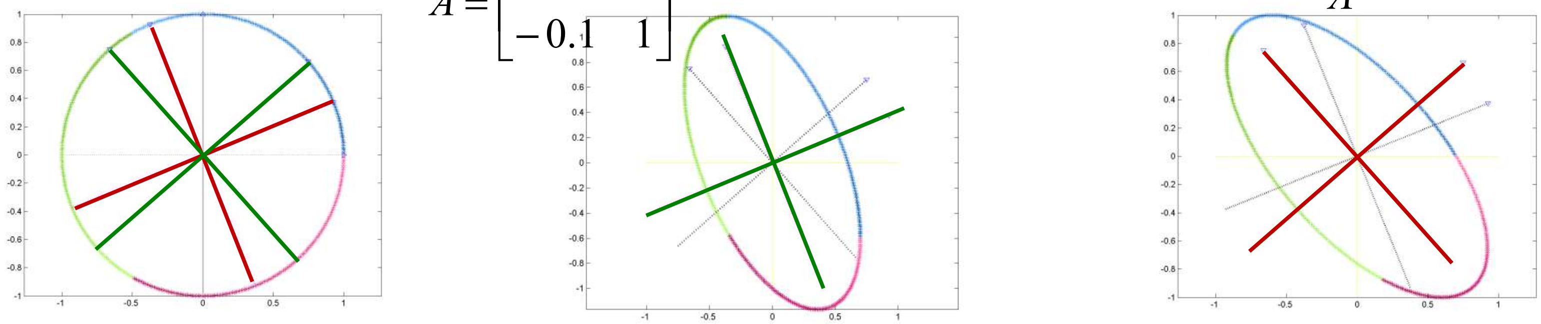


- Eigen analysis of a matrix **A**:
  - Find vectors such that their absolute directions are not changed by the transform
- SVD of a matrix **A**:
  - Find orthogonal set of vectors such that the *angle* between them is not changed by the transform
- For one class of matrices, these two operations are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

A matrix vs. its transpose

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

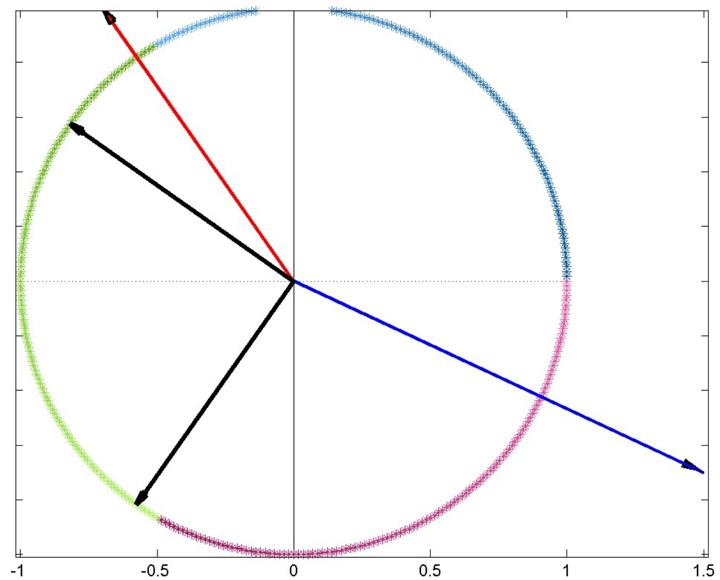
A^T



- 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

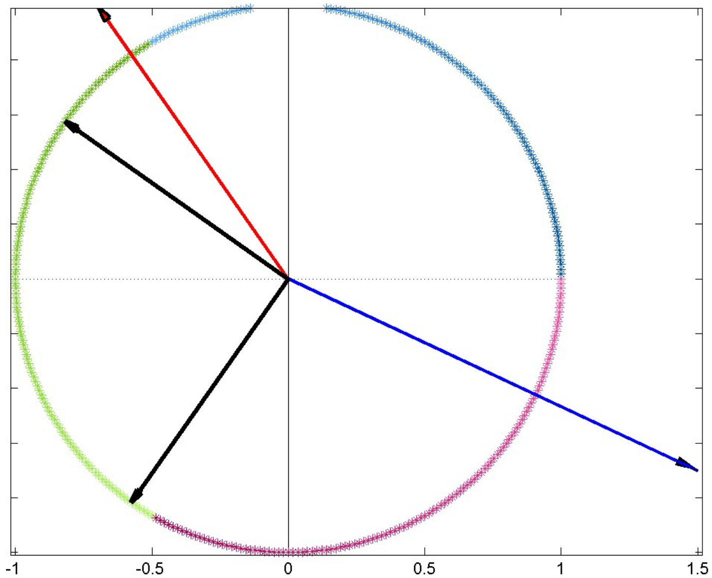
$$\begin{bmatrix} 1.5 & -0.7 \\ -0.7 & 1 \end{bmatrix}$$



- 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

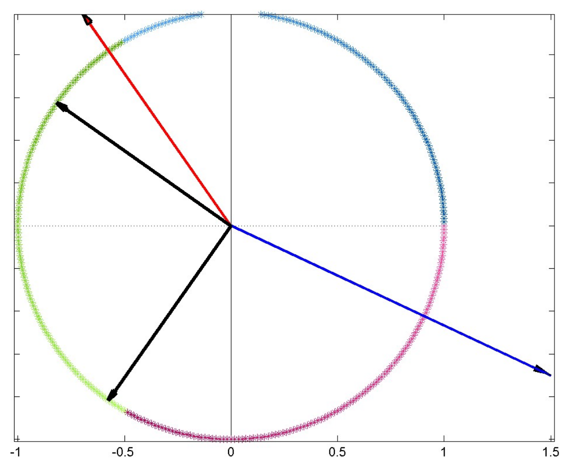
Symmetric Matrices

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

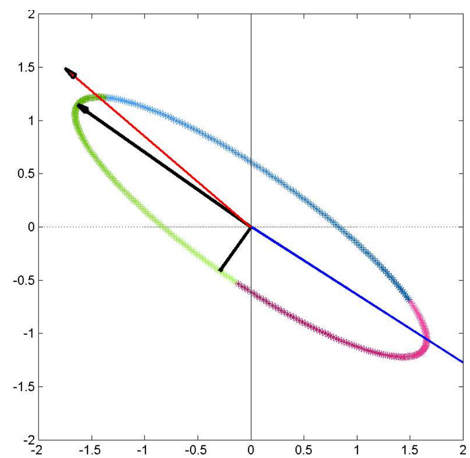


- Matrices that do not change on transposition
  - Row and column vectors are identical
- Symmetric matrix: Eigen vectors and Eigen values are always real
- Eigen vectors are always orthogonal
  - At 90 degrees to one another

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
  - $V_i^T V_i = 1$
  - $V_i^T V_j = 0, i \mathrel{!}= j$
- 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
- $M = V \Lambda V^T$

Definiteness..

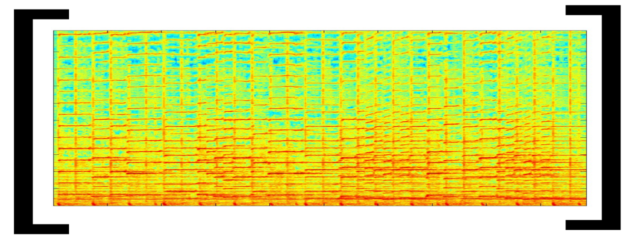
- SVD: Singular values are always positive!
- Eigen Analysis: Eigen values can be real or imaginary
  - Real, positive Eigen values represent stretching of the space along the Eigen vector
  - Real, *negative* Eigen values represent stretching and *reflection* (across origin) of Eigen vector
  - Complex 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** along orthogonal axes
    - Transformation has no permutation or 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
  - $x^T A x$ 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 these and other gram matrices later

SVD on data-container matrices



$$\mathbf{X} = [X_1 \ X_2 \quad \cdots X_N]$$

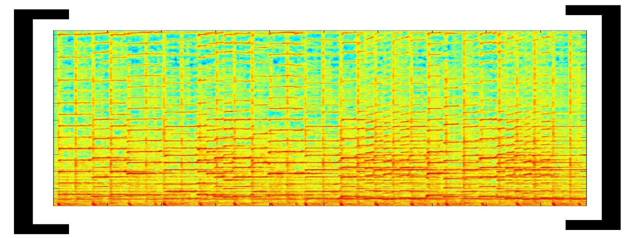
$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$$

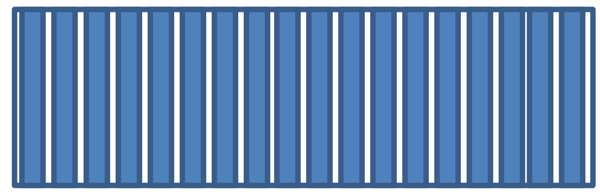
- We can also perform SVD on matrices that are *data containers*
- **S** is a d x N rectangular matrix
  - N vectors of dimension d
- **U** is an orthogonal matrix of d vectors of size d
  - All vectors are length 1
- **V** is an orthogonal matrix of N vectors of size N
- **S** is a d x N diagonal matrix with non-zero entries only on diagonal

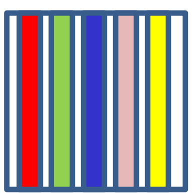
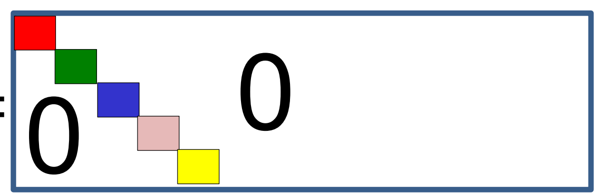
SVD on data-container matrices

$$\mathbf{X} = [X_1 \; X_2 \quad \cdots X_N]$$

$$\boxed{\mathbf{X} = \mathbf{USV}^{\mathrm{T}}}$$



$\mathbf{X} =$ 

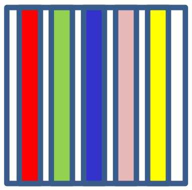
$\mathbf{U} =$  $\mathbf{S} =$ 

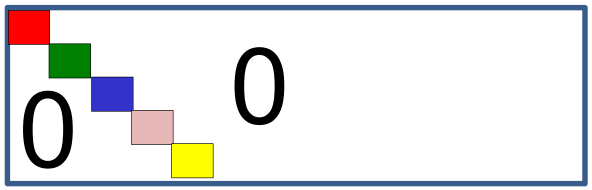
$\mathbf{V}^{\mathrm{T}} =$ 

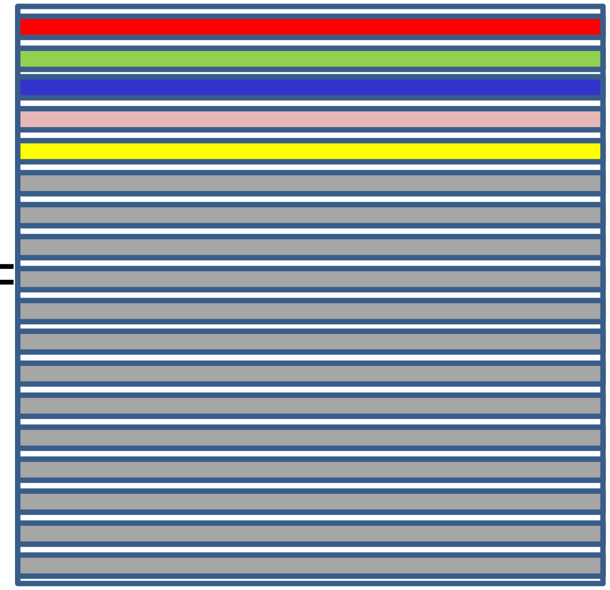
$|U_{\mathrm{i}}| = 1.0$ for every vector in $\mathbf{U}$

$|V_{\mathrm{i}}| = 1.0$ for every vector in $\mathbf{V}$

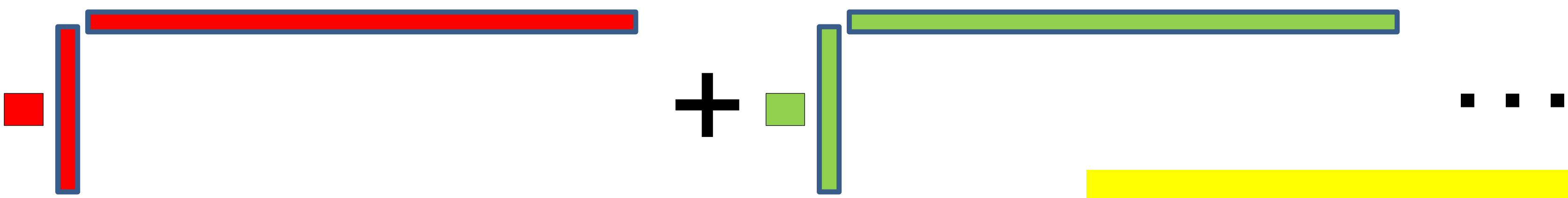
SVD on data-container matrices

$\mathbf{U} =$ 

$\mathbf{S} =$ 

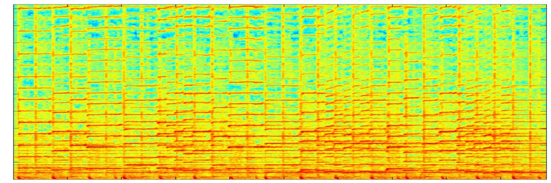
$\mathbf{V}^{\mathrm{T}} =$ 

$\mathbf{X} = \mathbf{USV}^{\mathrm{T}} =$



$$\mathbf{X} = \sum_i s_i U_i V_i^{\,T}$$

Expanding the SVD



$$\mathbf{X} = [X_1 \; X_2 \quad \cdots X_N]$$

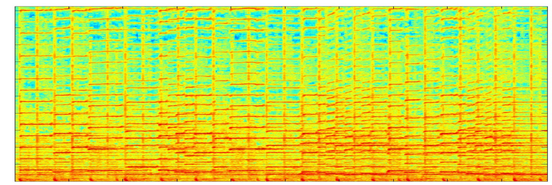
$$\mathbf{X} = \mathbf{USV}^{\mathrm{T}}$$

$$\mathbf{X} = s_1 U_1 V_1^T + s_2 U_2 V_2^T + s_3 U_3 V_3^T + s_4 U_4 V_4^T + \ldots$$

- Each left singular vector and the corresponding right singular vector contribute on “basic” component to the data

- The “magnitude” of its contribution is the corresponding singular value

Expanding the SVD



$$\mathbf{X} = [X_1 \ X_2 \ \ \cdots X_N]$$

$$\mathbf{X} = \mathbf{USV}^{\mathrm{T}}$$

$$\mathbf{X} = s_1 U_1 V_1^T + s_2 U_2 V_2^T + s_3 U_3 V_3^T + s_4 U_4 V_4^T + \ldots$$

magnitude basis modulation

- Each left singular vector and the corresponding right singular vector contribute on "basic" component to the data

- The "magnitude" of its contribution is the corresponding singular value

Expanding the SVD

$$\mathbf{X} = s_1 U_1 V_1^T + s_2 U_2 V_2^T + s_3 U_3 V_3^T + s_4 U_4 V_4^T + \ldots$$

- Each left singular vector and the corresponding right singular vector contribute on "basic" component to the data
- The "magnitude" of its contribution is the corresponding singular value
- Low singular-value components contribute little, if anything
  - Carry little information
  - Are often just "noise" in the data

Expanding the SVD

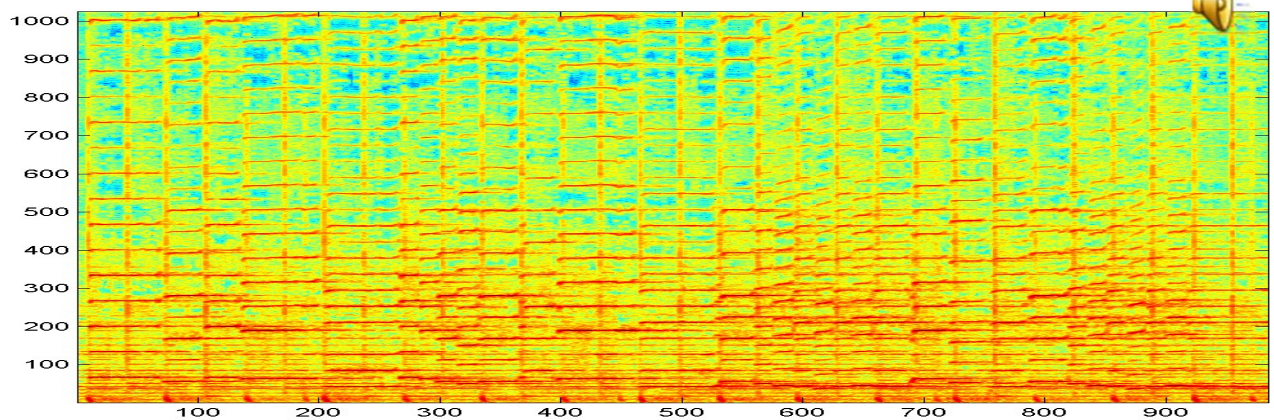
$$\mathbf{X} = s_1 U_1 V_1^T + s_2 U_2 V_2^T + s_3 U_3 V_3^T + s_4 U_4 V_4^T + \ldots$$

$$\mathbf{X} \approx s_1 U_1 V_1^T + s_2 U_2 V_2^T$$

- Low singular-value components contribute little, if anything
  - Carry little information
  - Are often just "noise" in the data

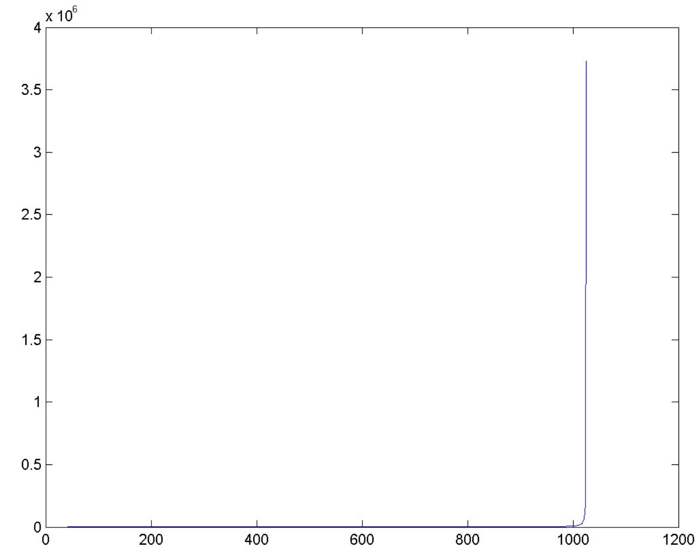
- Data can be recomposed using only the "major" components with minimal change of value
  - Minimum squared error between original data and recomposed data
  - Sometimes eliminating the low-singular-value components will, in fact "clean" the data

An audio example



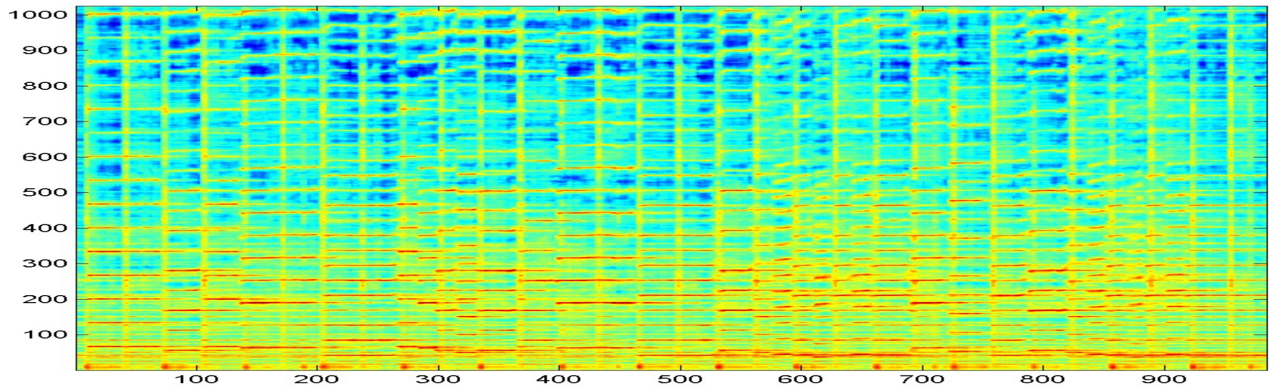
- The spectrogram has 974 vectors of dimension 1025
  - A 1024x974 matrix!
- Decompose: $\mathbf{M} = \mathbf{USV}^{\mathrm{T}} = \sum_i s_i U_i V_i^{\mathrm{T}}$
- **U** is 1024 x 1024
- **V** is 974 x 974
- There are 974 non-zero singular values S_i

Singular Values



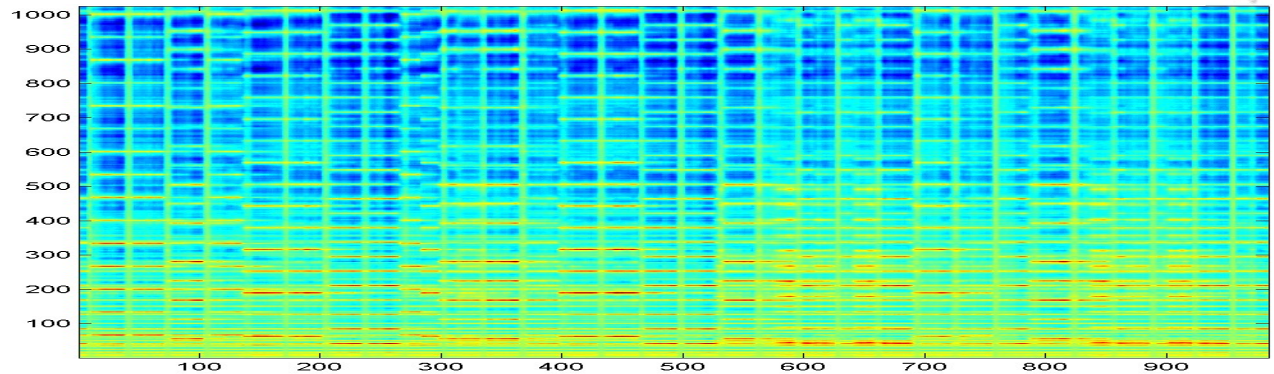
- Singular values for spectrogram **M**
  - Most Singluar values are close to zero
  - The corresponding components are "unimportant"

An audio example



- The same spectrogram constructed from only the 25 highest singular-value components
  - Looks similar
    - With 100 components, it would be indistinguishable from the original
  - Sounds pretty close
  - Background "cleaned up"

With only 5 components



- The same spectrogram constructed from only the 5 highest-valued components
  - Corresponding to the 5 largest singular values
  - Highly recognizable
  - Suggests that there are actually only 5 significant unique note combinations in the music

- Next up: A brief trip through optimization..