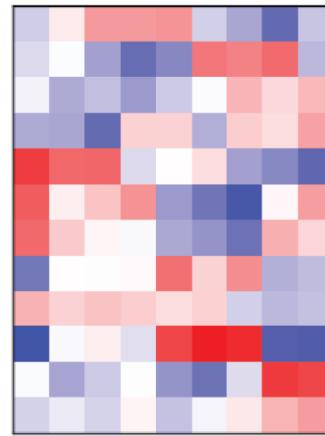
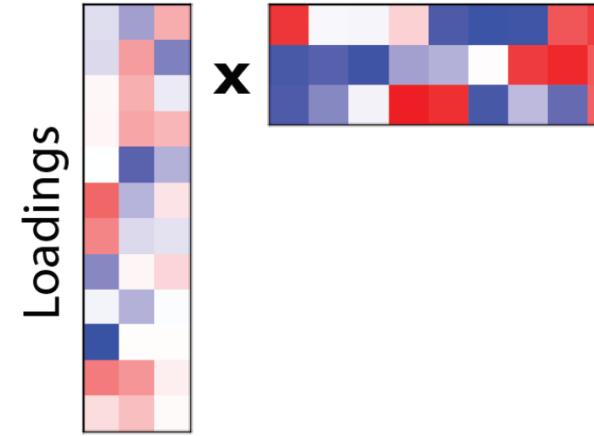


Original Data

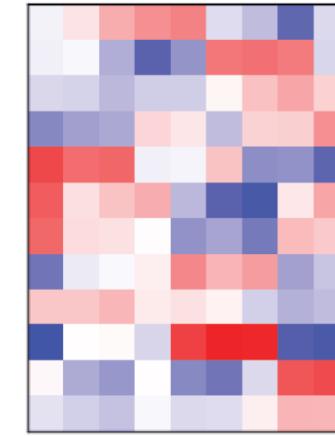


\approx

Components

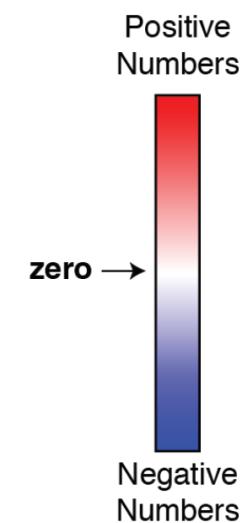
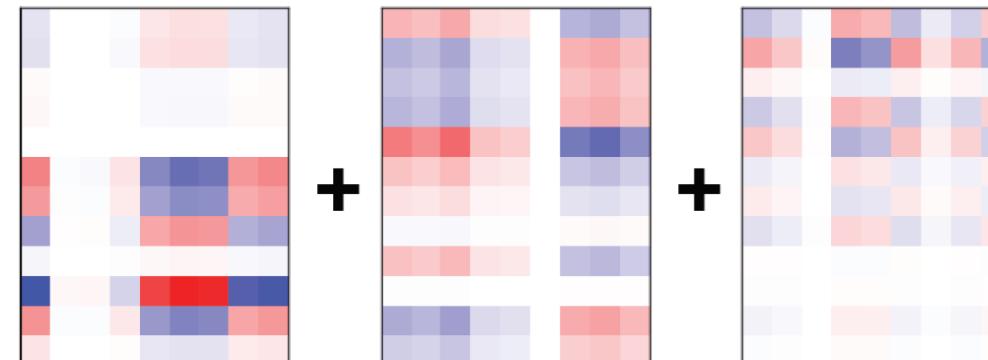


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



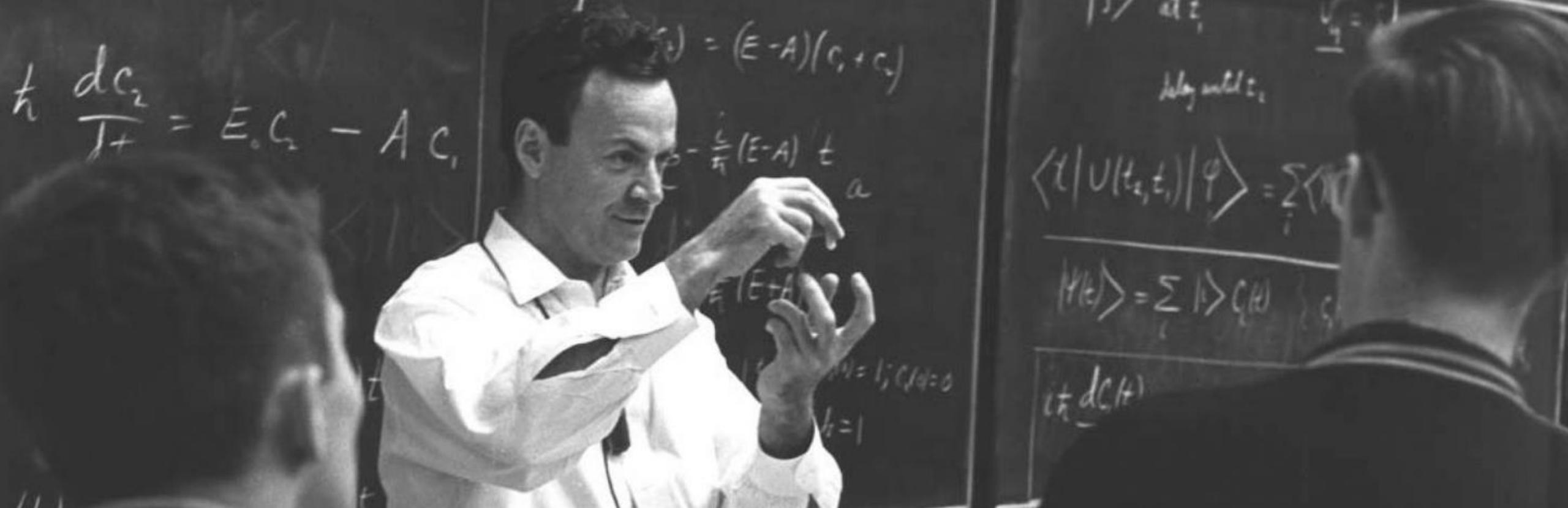
@neurosyntaxacademy



@Neuro_Syntax



neurosyntax



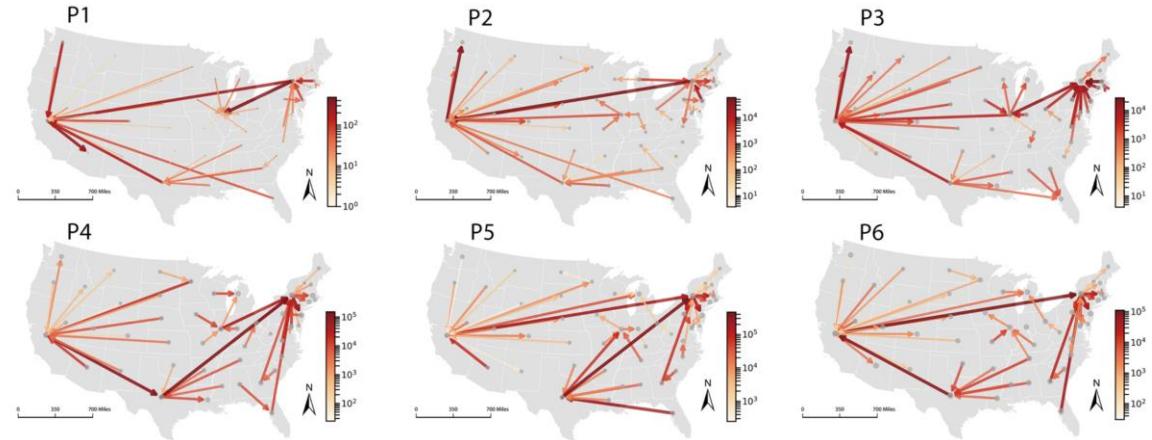
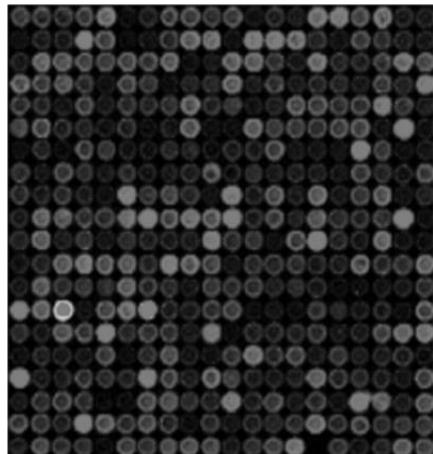
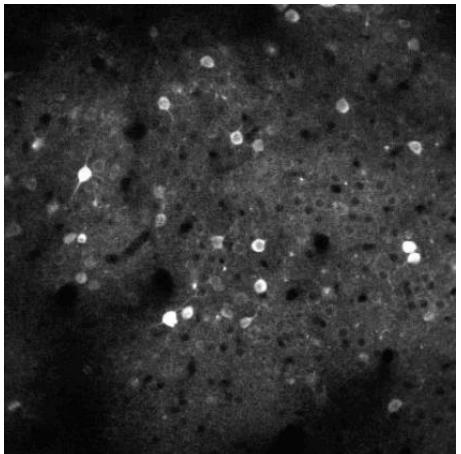
I learned very early the difference between knowing the name of something and knowing something.

Richard Feynman

Session 1

Lecturer: Saman Abbaspoor

Discover patterns in nature and explain them

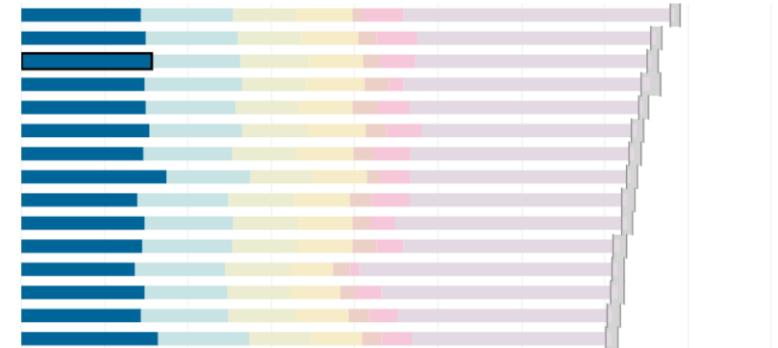


Official World Happiness Report 2021 Ranking

Figure 2.1 Ranking of Happiness based on a three-year-average 2018-2020.

Country

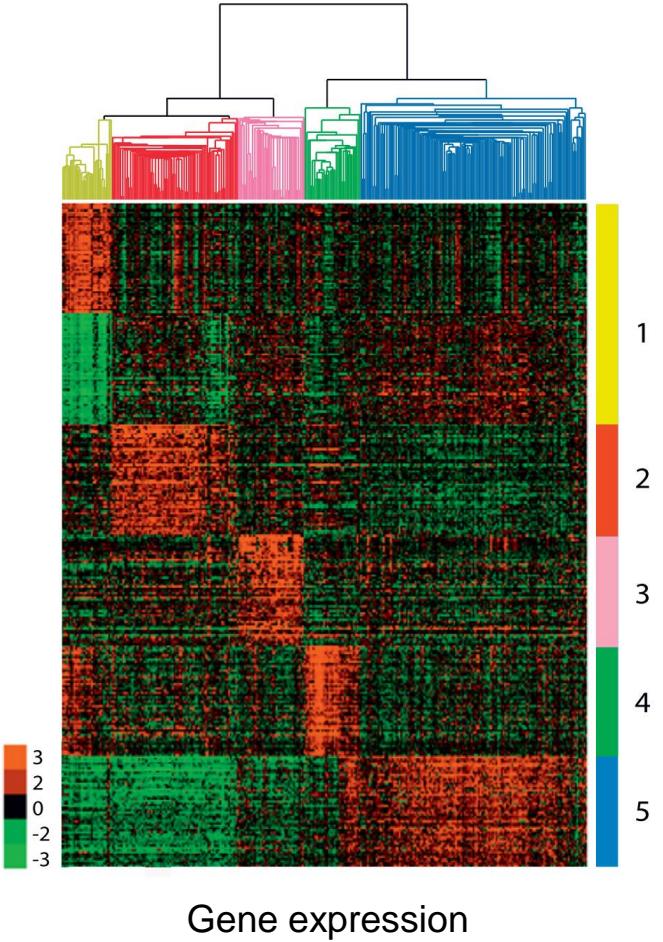
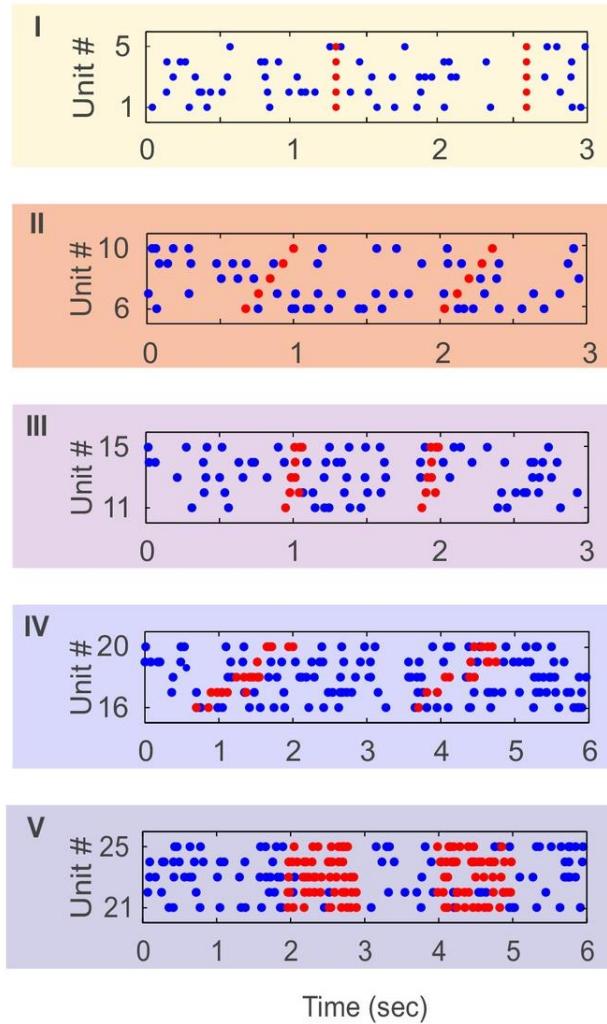
1. Finland (7.842)
2. Denmark (7.620)
3. Switzerland (7.571)
4. Iceland (7.554)
5. Netherlands (7.464)
6. Norway (7.392)
7. Sweden (7.363)
8. Luxembourg (7.324)*
9. New Zealand (7.277)
10. Austria (7.268)
11. Australia (7.183)
12. Israel (7.157)
13. Germany (7.155)
14. Canada (7.103)
15. Ireland (7.085)



Patterns

Regularity and Predictability:
patterns involve a predictable and repeatable arrangement of elements.

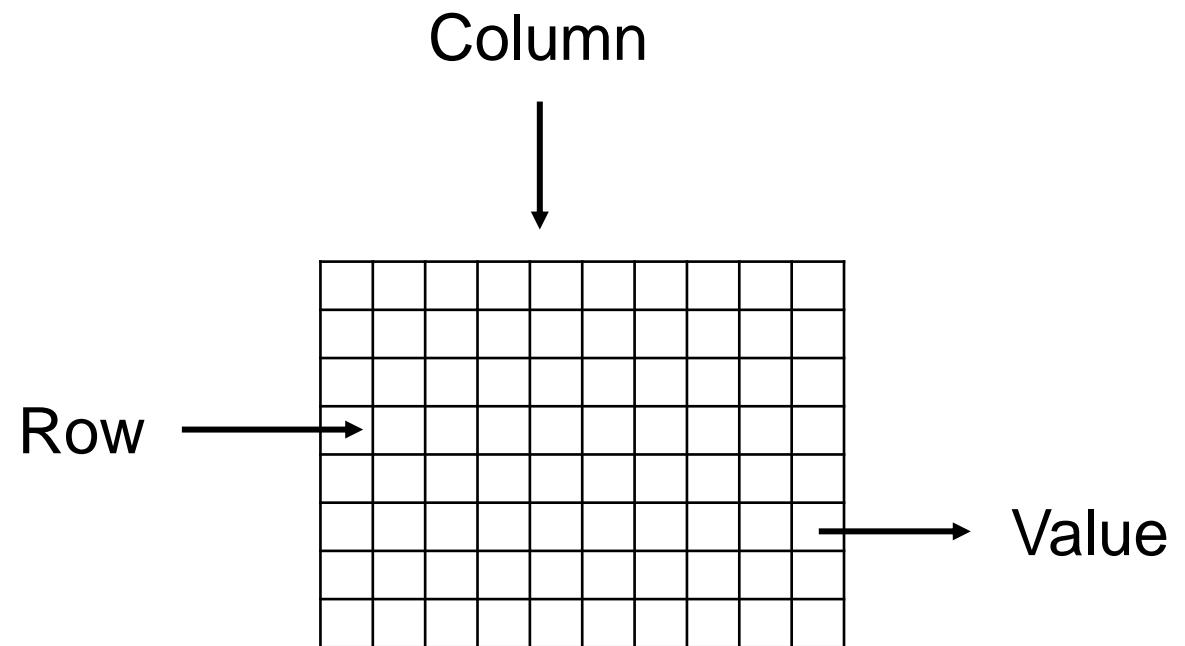
In terms of correlation, a pattern can be defined as a statistical relationship between two or more variables where changes in one variable are systematically associated with changes in another variable.



Observation and data structure

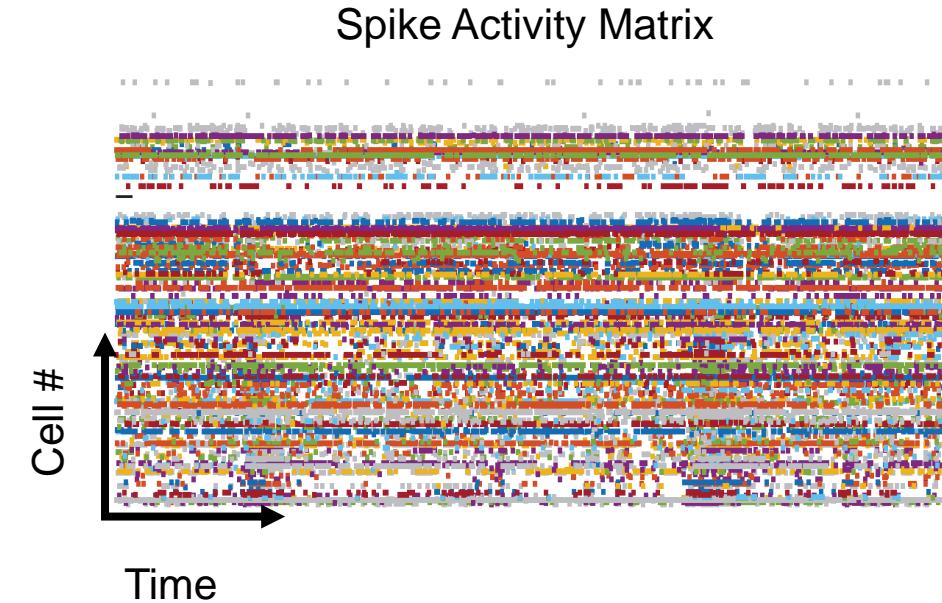
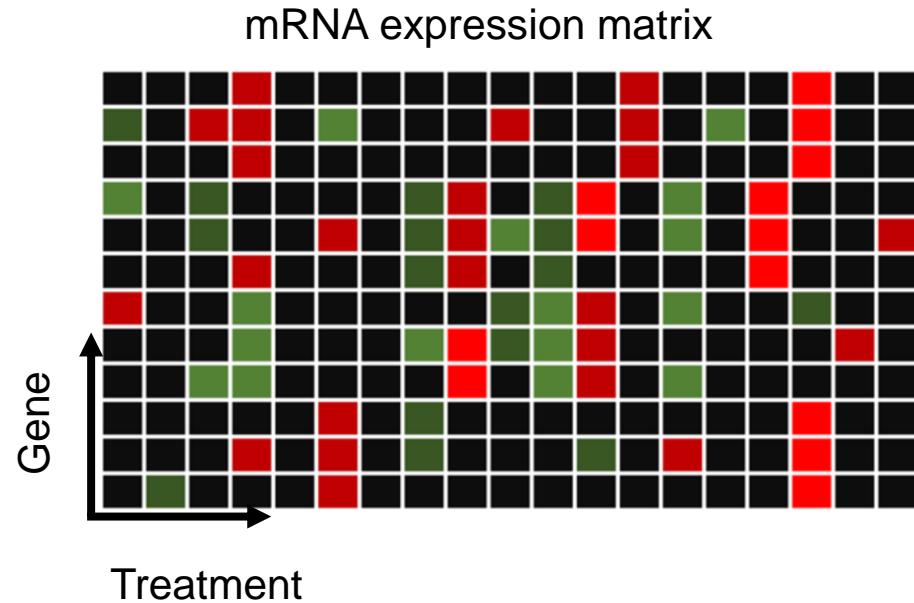
Matrix as a data structure

Rows, Columns, and Values
in a matrix are meaningful
under the context of the data
structure

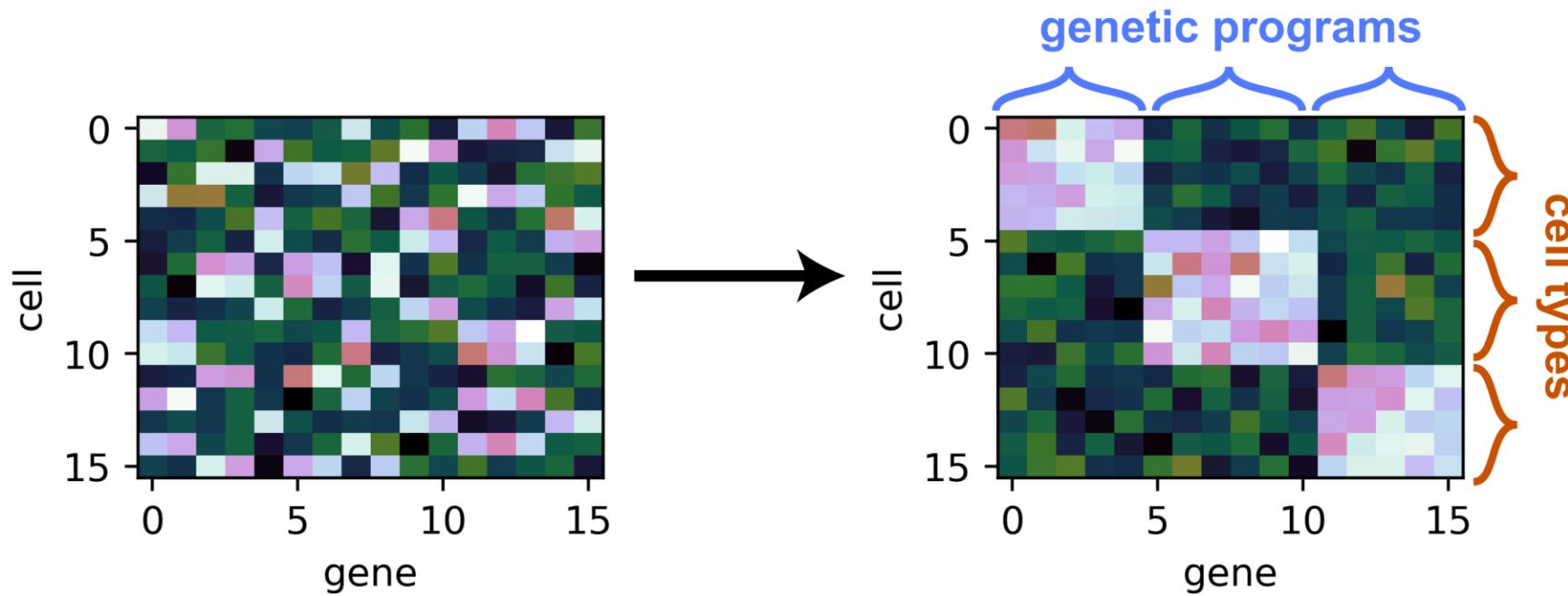


Observation and data structure

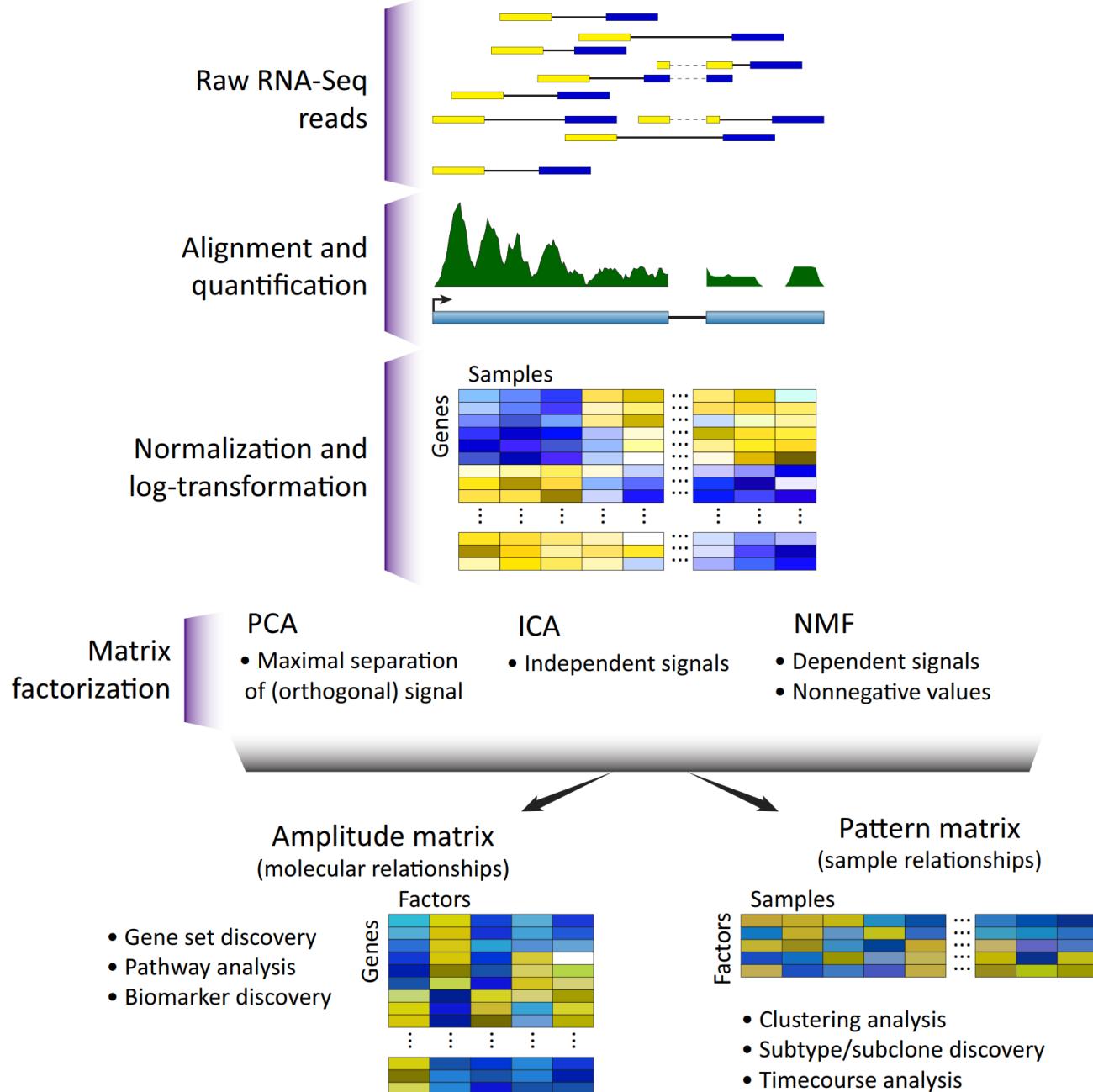
Matrix as a data structure



Discover patterns in high-throughput datasets



Enter the Matrix: Factorization Uncovers Patterns



Applications of matrix decomposition

Neuroscience data

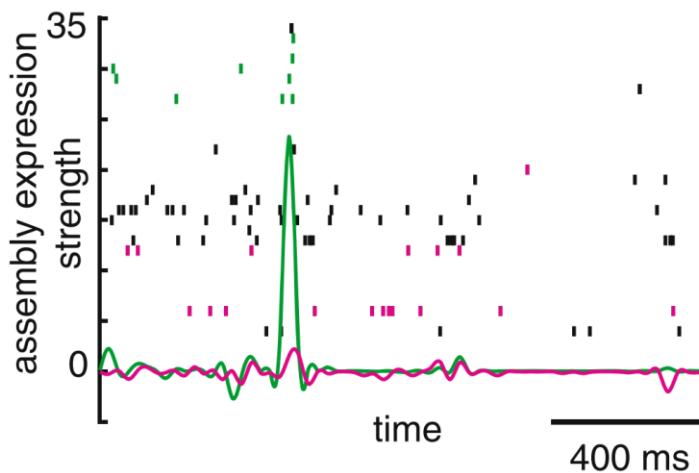
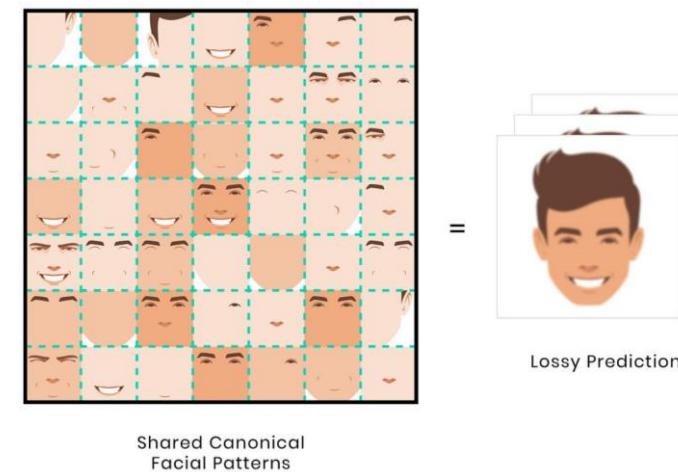
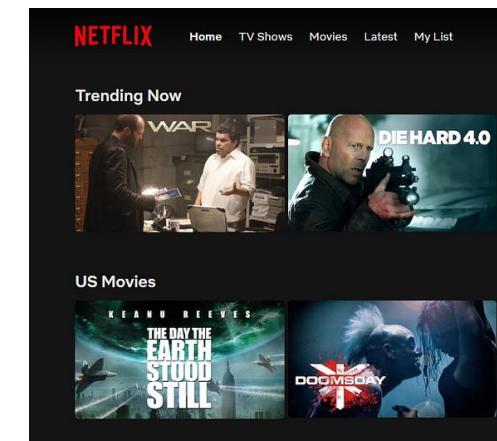


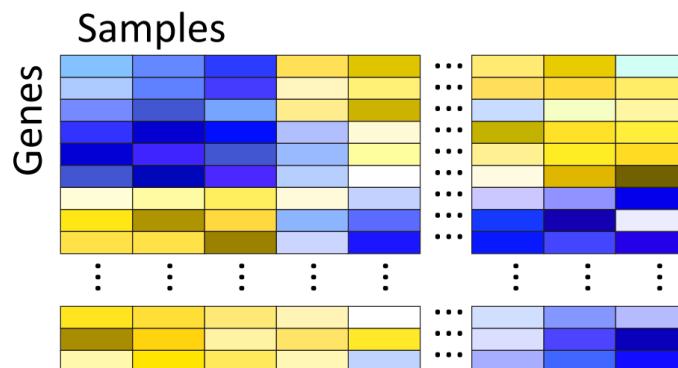
Image recognition



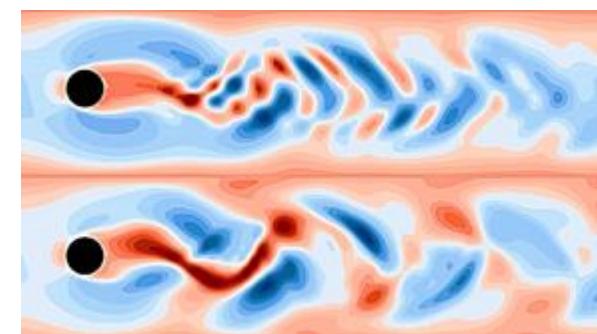
Search engines



Omics data



Fluid Dynamics

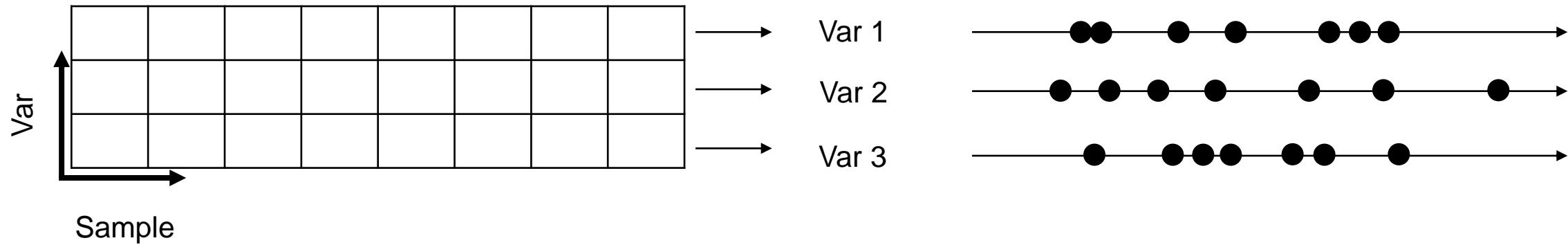


Google



High-dimensional data spaces

Projecting high throughput data into high-dimensional spaces

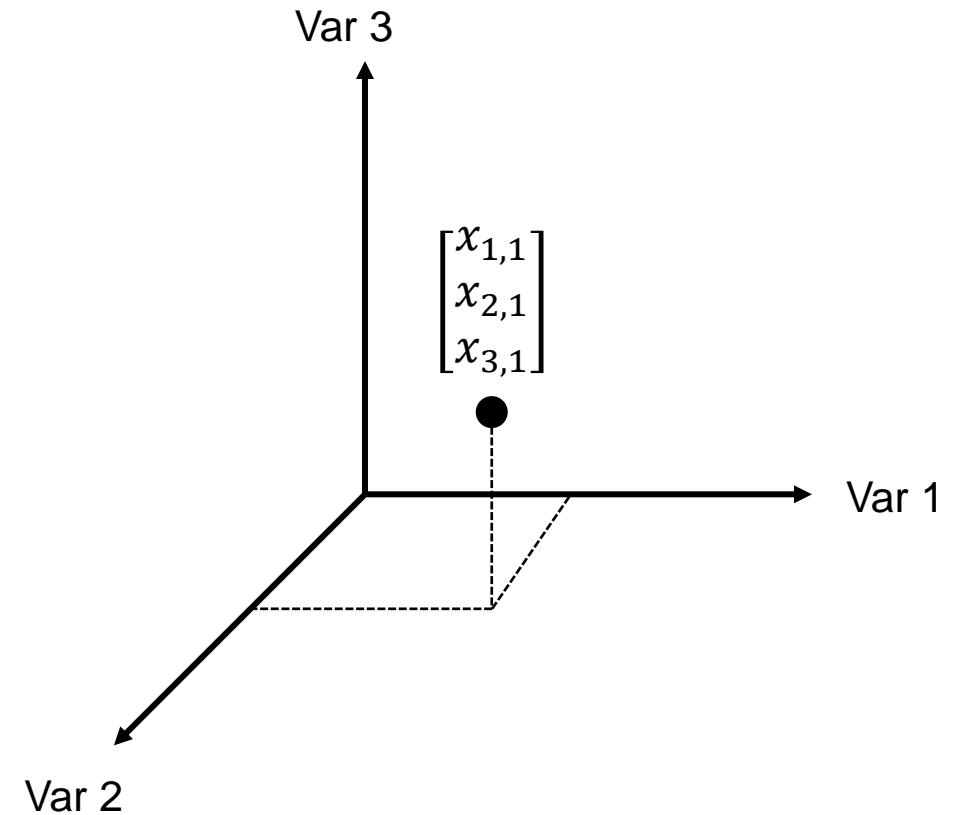


Projecting high throughput data into high-dimensional spaces

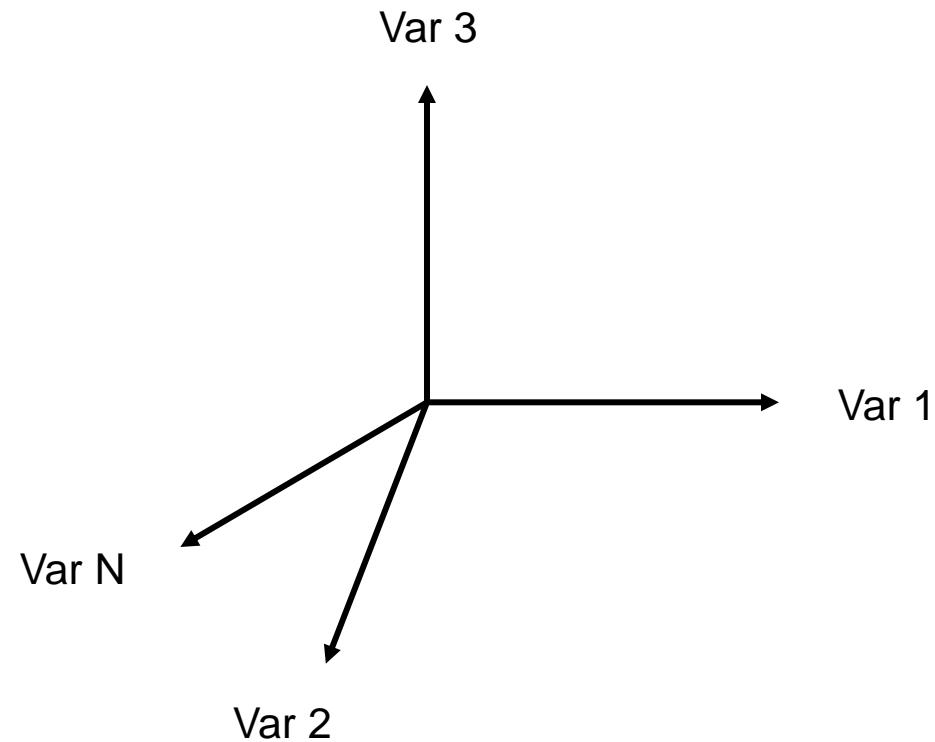
Var

Sample

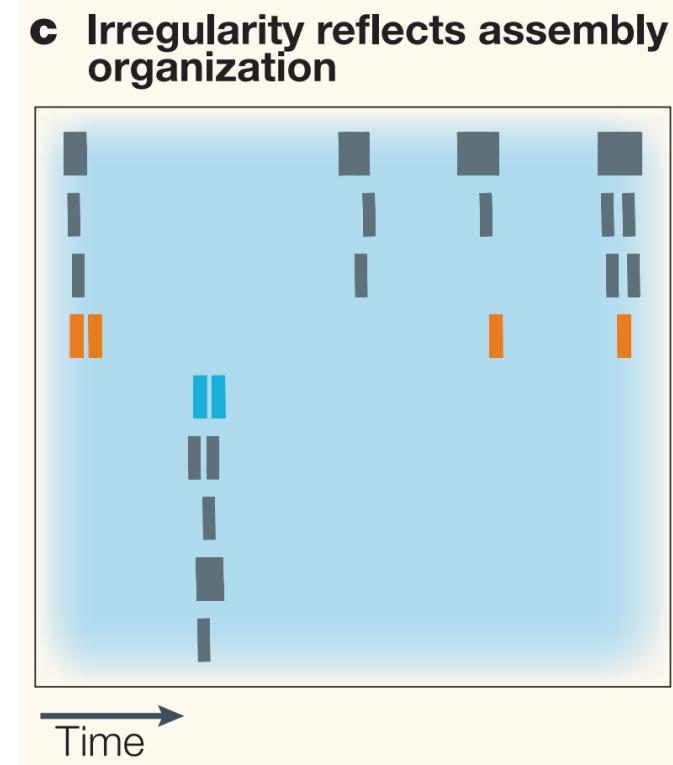
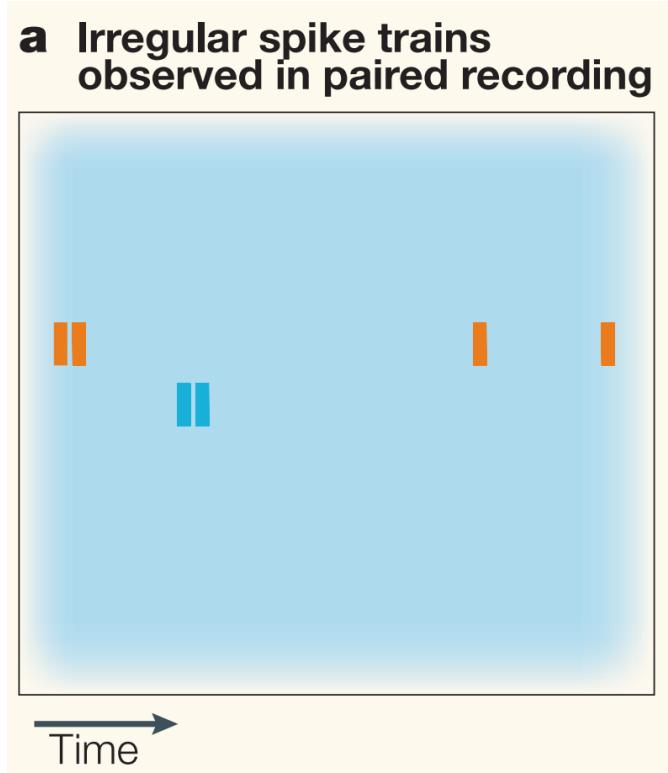
$x_{1,1}$						
$x_{2,1}$						
$x_{3,1}$						



Projecting high throughput data into high-dimensional spaces

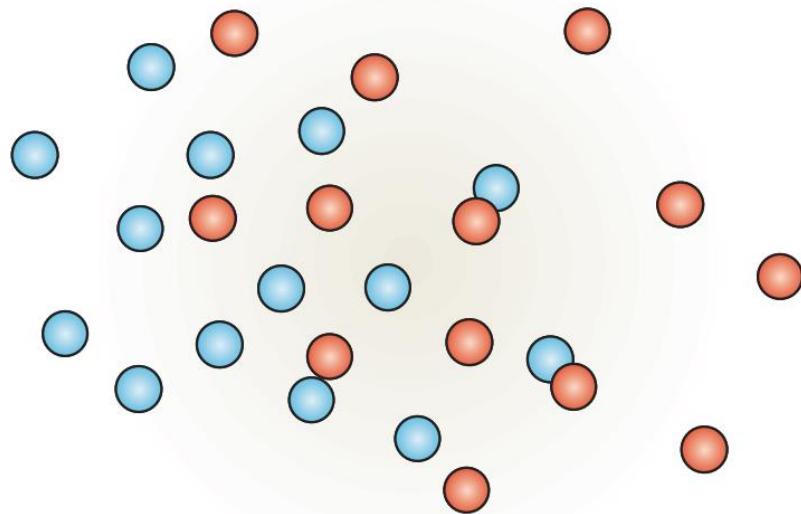


Higher dimensional data spaces might uncover patterns

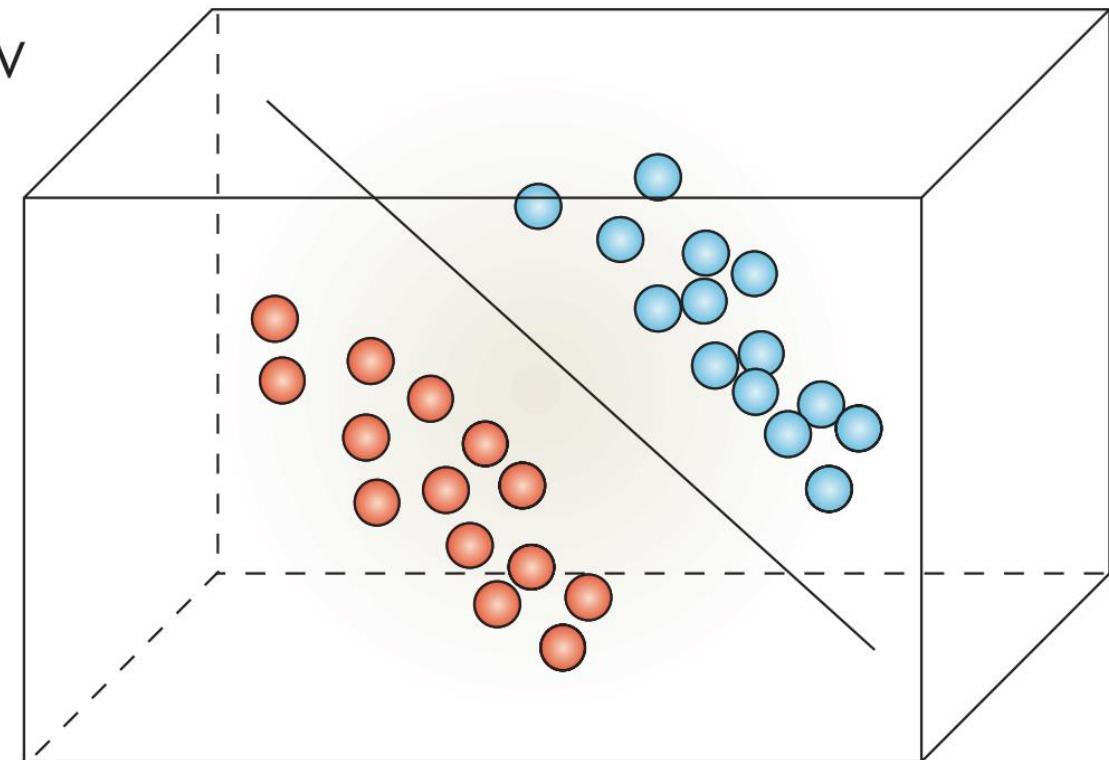


Cluster separability in data space

iii



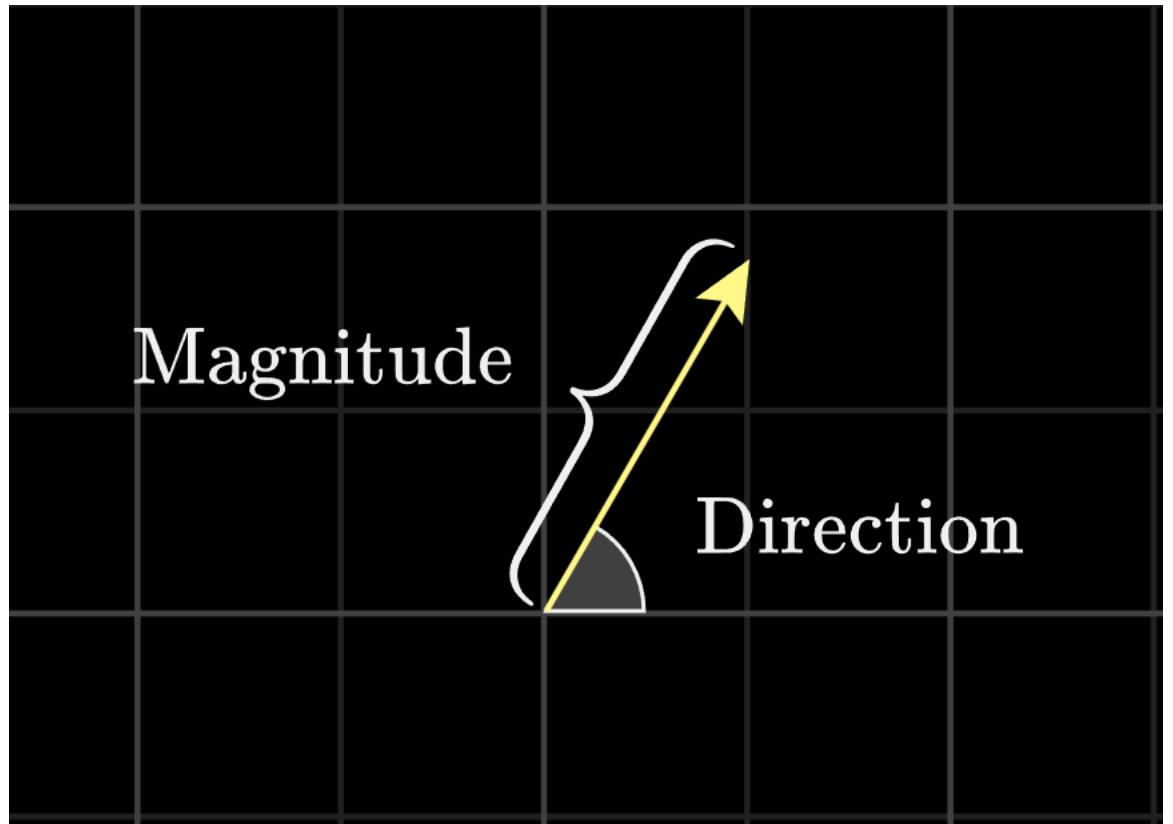
iv



Vector

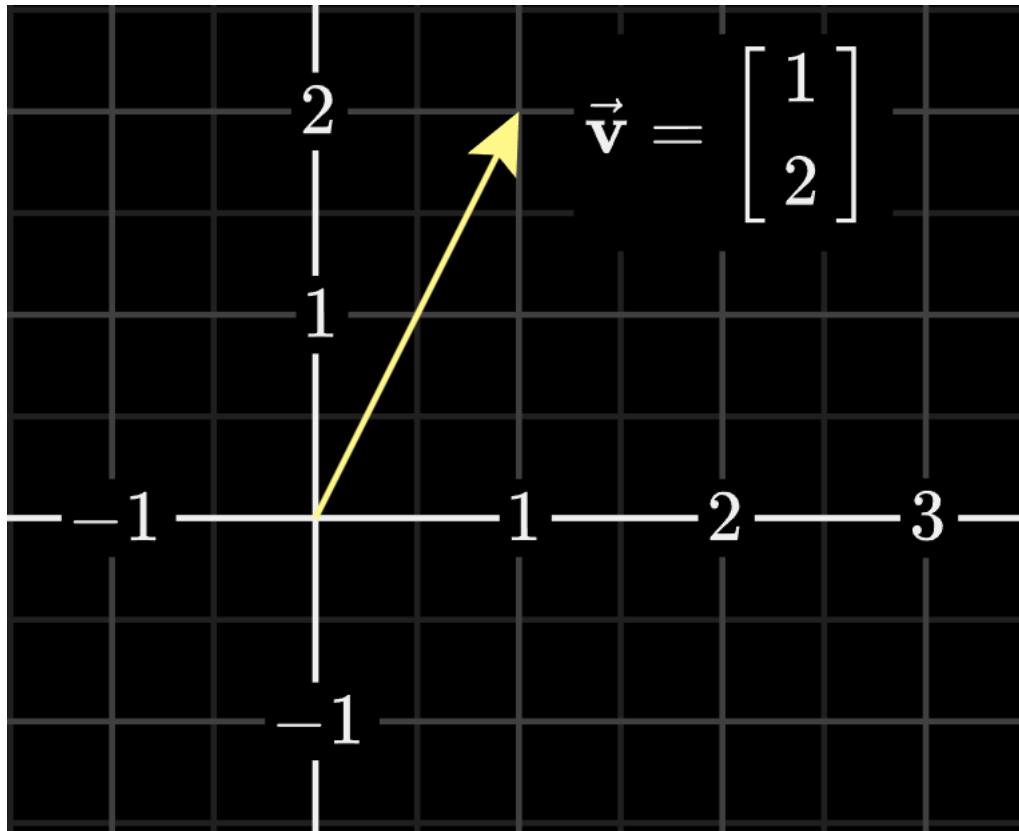
Vectors

A quantity that has magnitude as well as direction is called a vector. The point A from where the vector starts is called its initial point, and the point B where it ends is called its terminal point. The distance between initial and terminal points of a vector is called the magnitude (or length) of the vector.



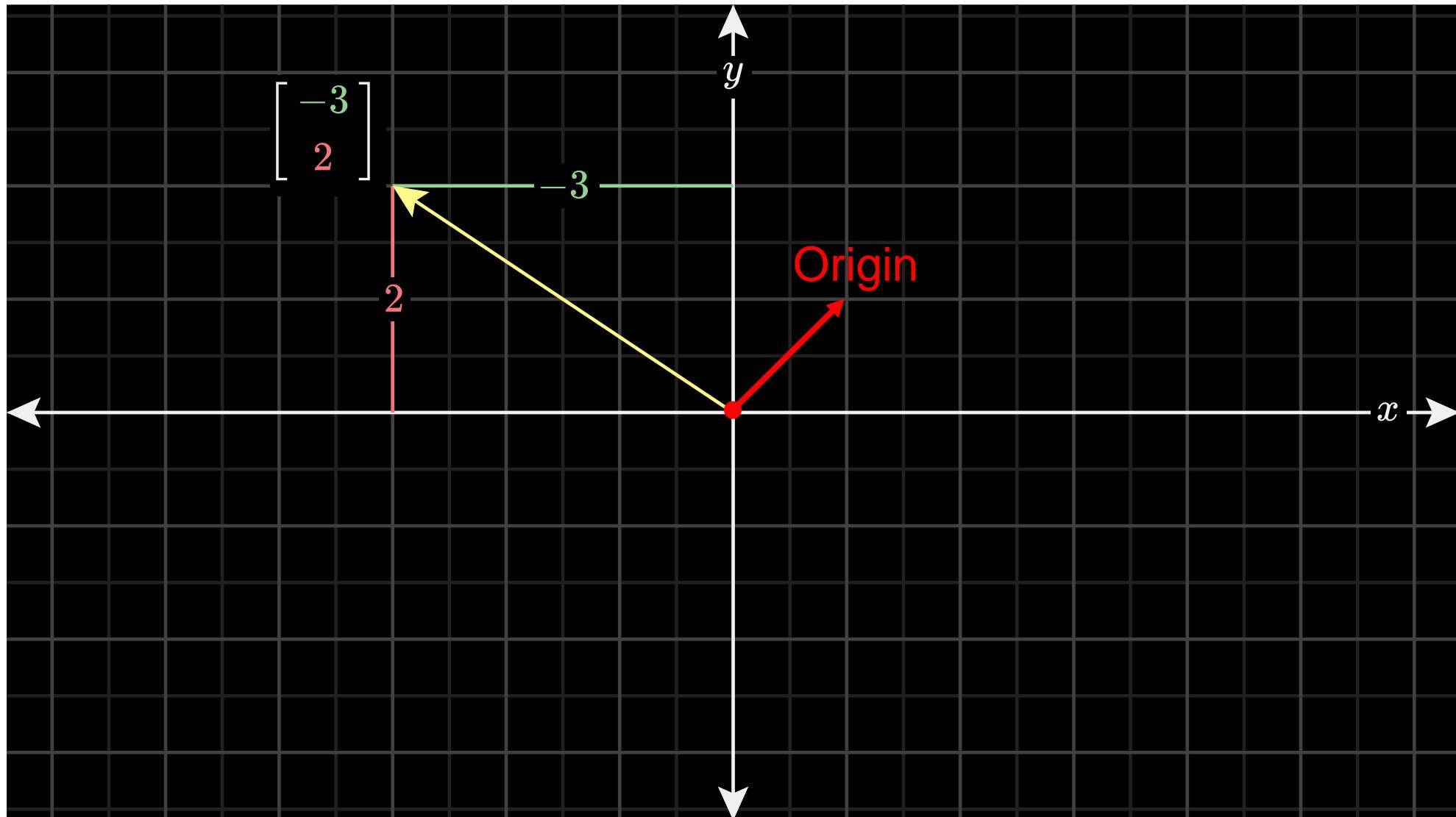
Vectors

A quantity that has magnitude as well as direction is called a vector. The point A from where the vector starts is called its initial point, and the point B where it ends is called its terminal point. The distance between initial and terminal points of a vector is called the magnitude (or length) of the vector.

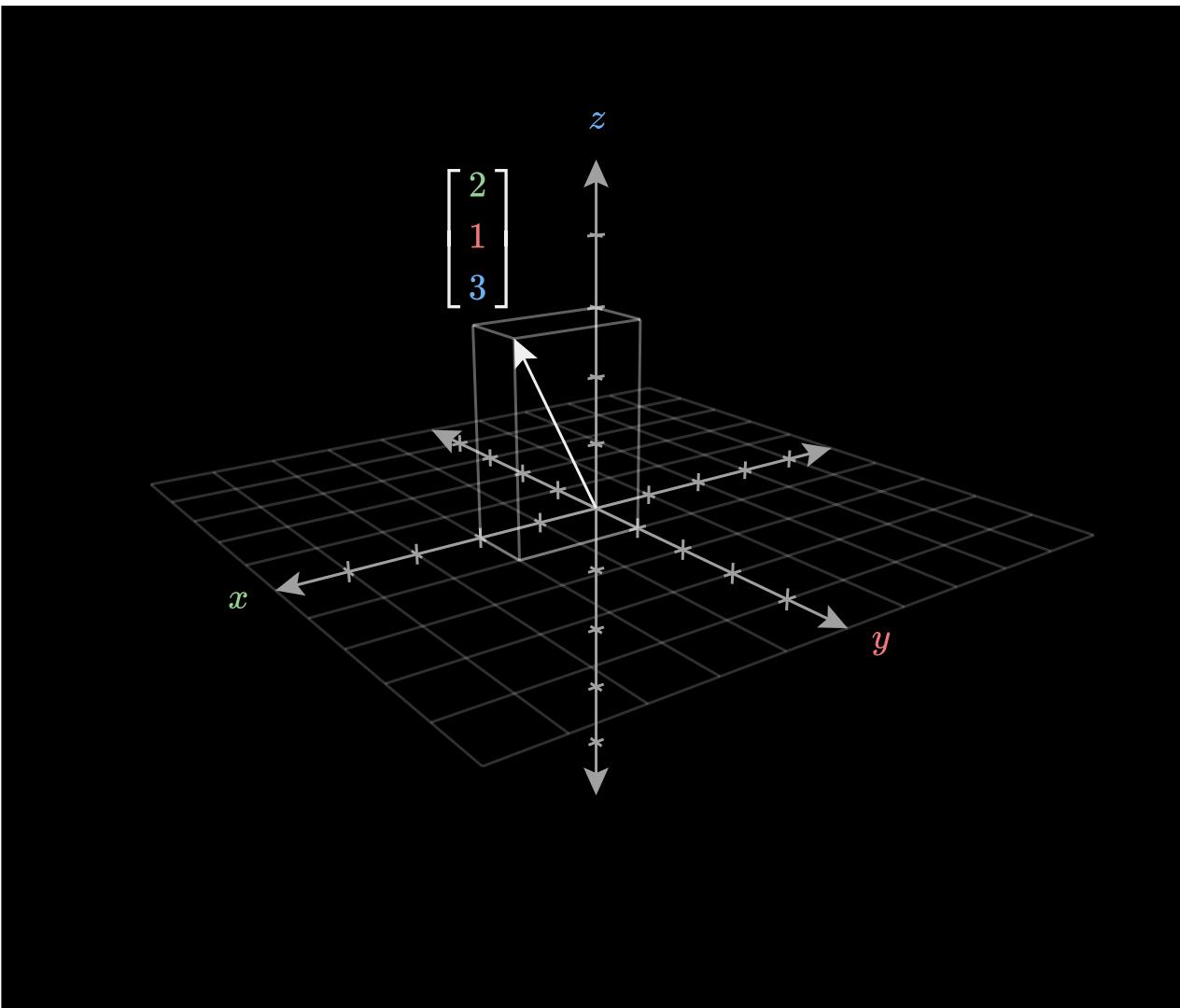


<https://www.3blue1brown.com/lessons/vectors>

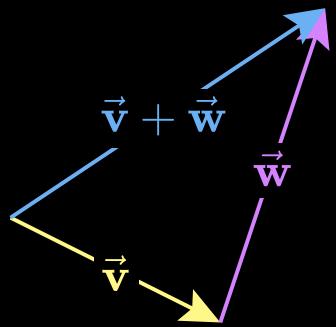
Thinking About Coordinate Systems



3D Coordinate Systems

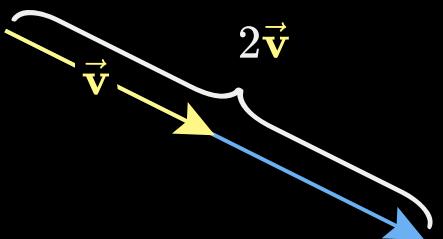


Vector Operations: Addition



$$\vec{v} + \vec{w}$$

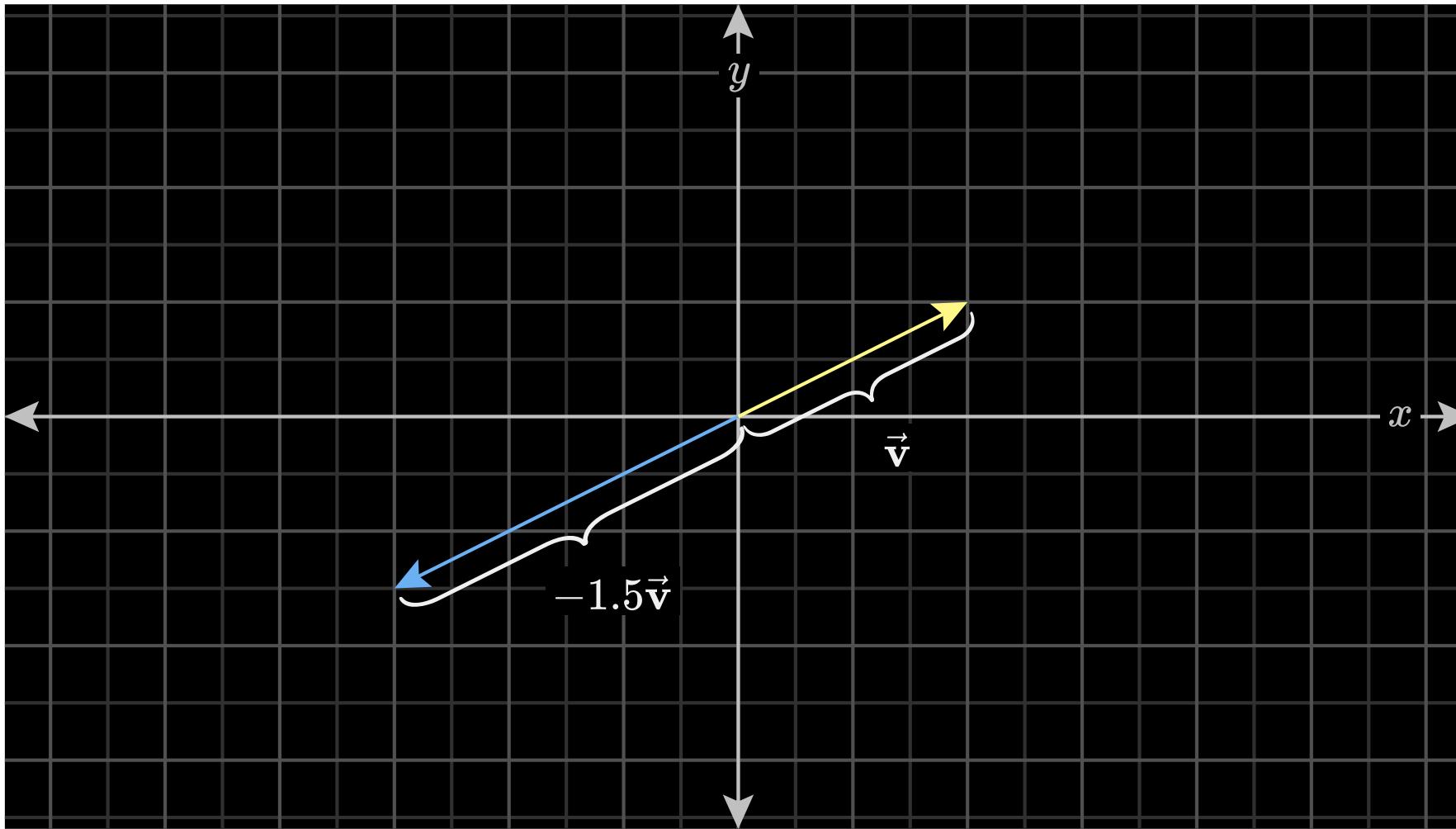
$$\begin{bmatrix} 2 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2+1 \\ -1+3 \end{bmatrix}$$



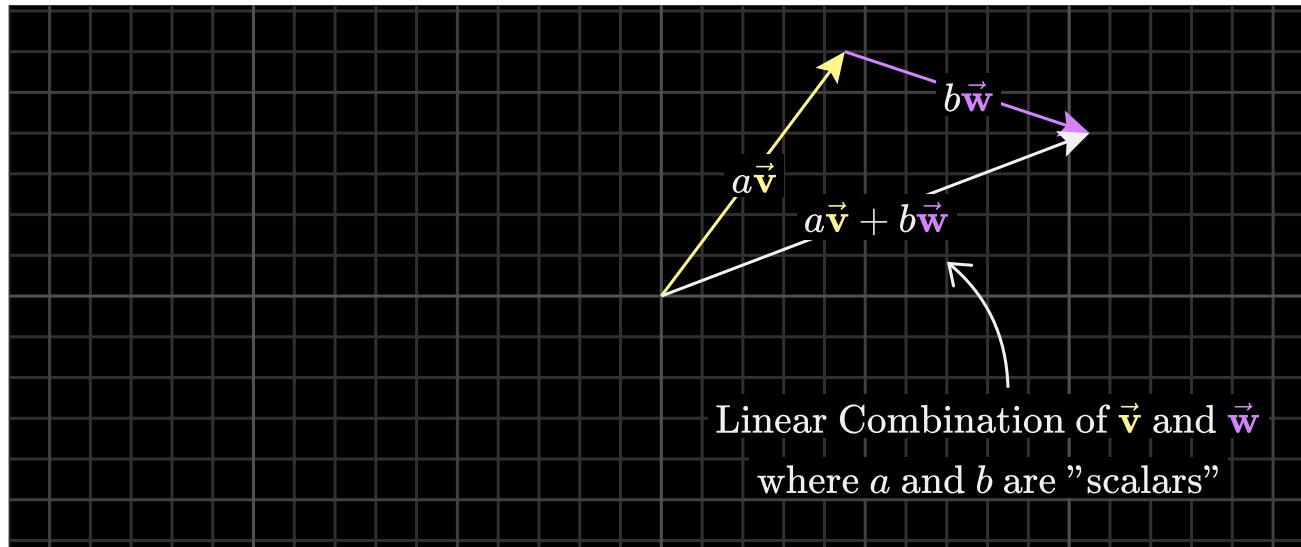
$$2\vec{v}$$

$$2 \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 2(2) \\ 2(-1) \end{bmatrix}$$

Vector Operations: Scaling



Linear Combinations



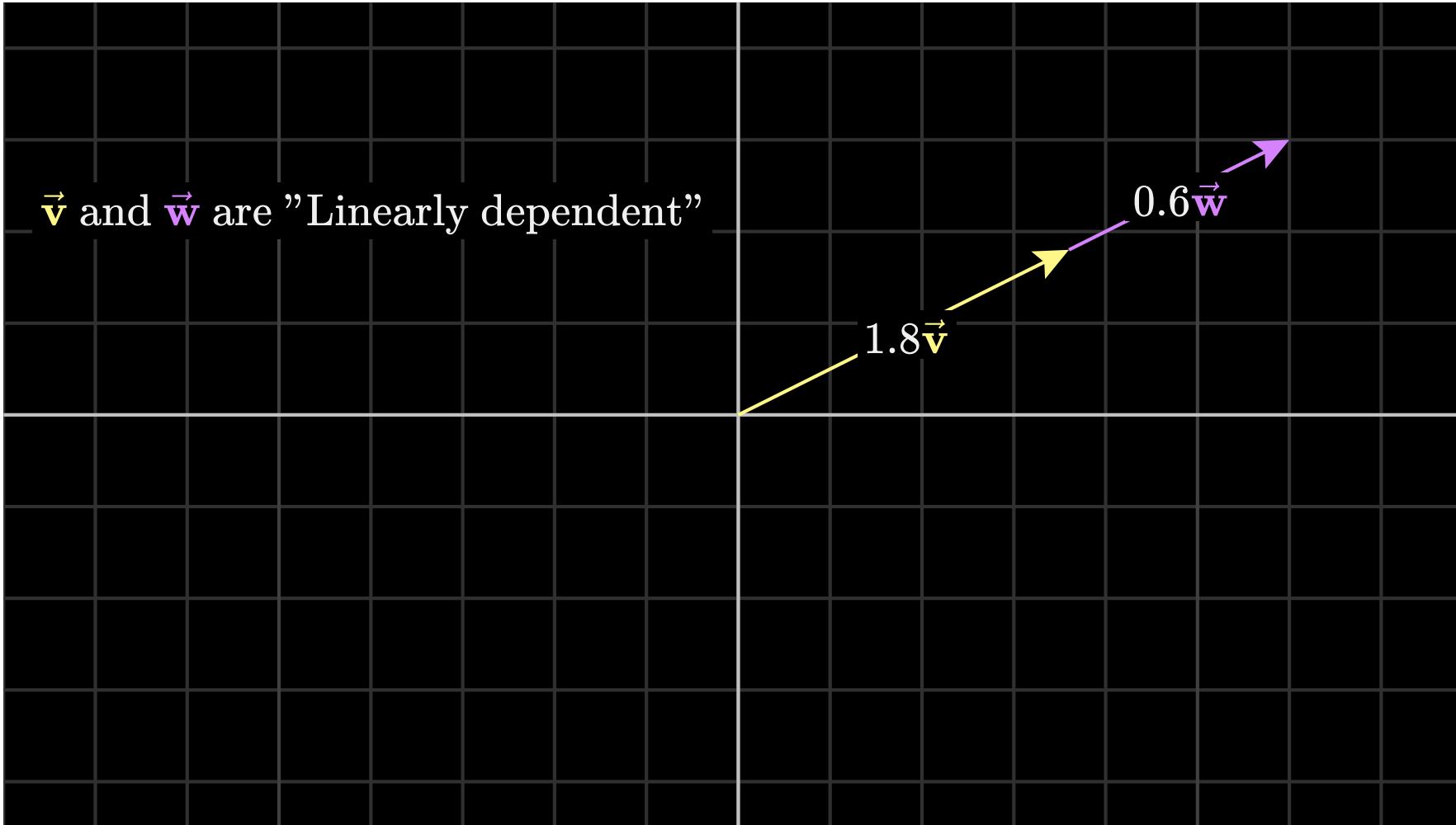
2.2 definition: *linear combination*

A *linear combination* of a list v_1, \dots, v_m of vectors in V is a vector of the form

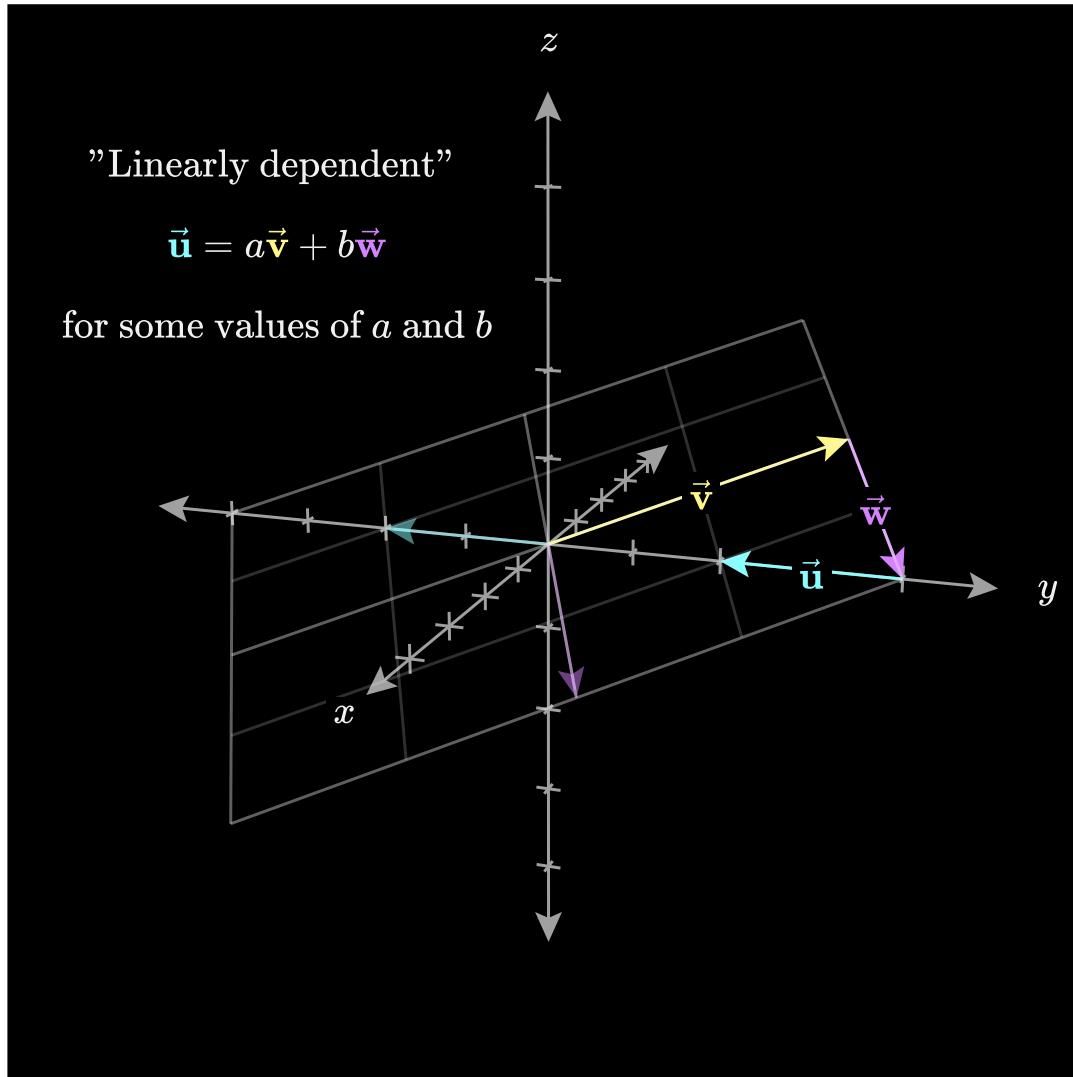
$$a_1v_1 + \dots + a_mv_m,$$

where $a_1, \dots, a_m \in \mathbf{F}$.

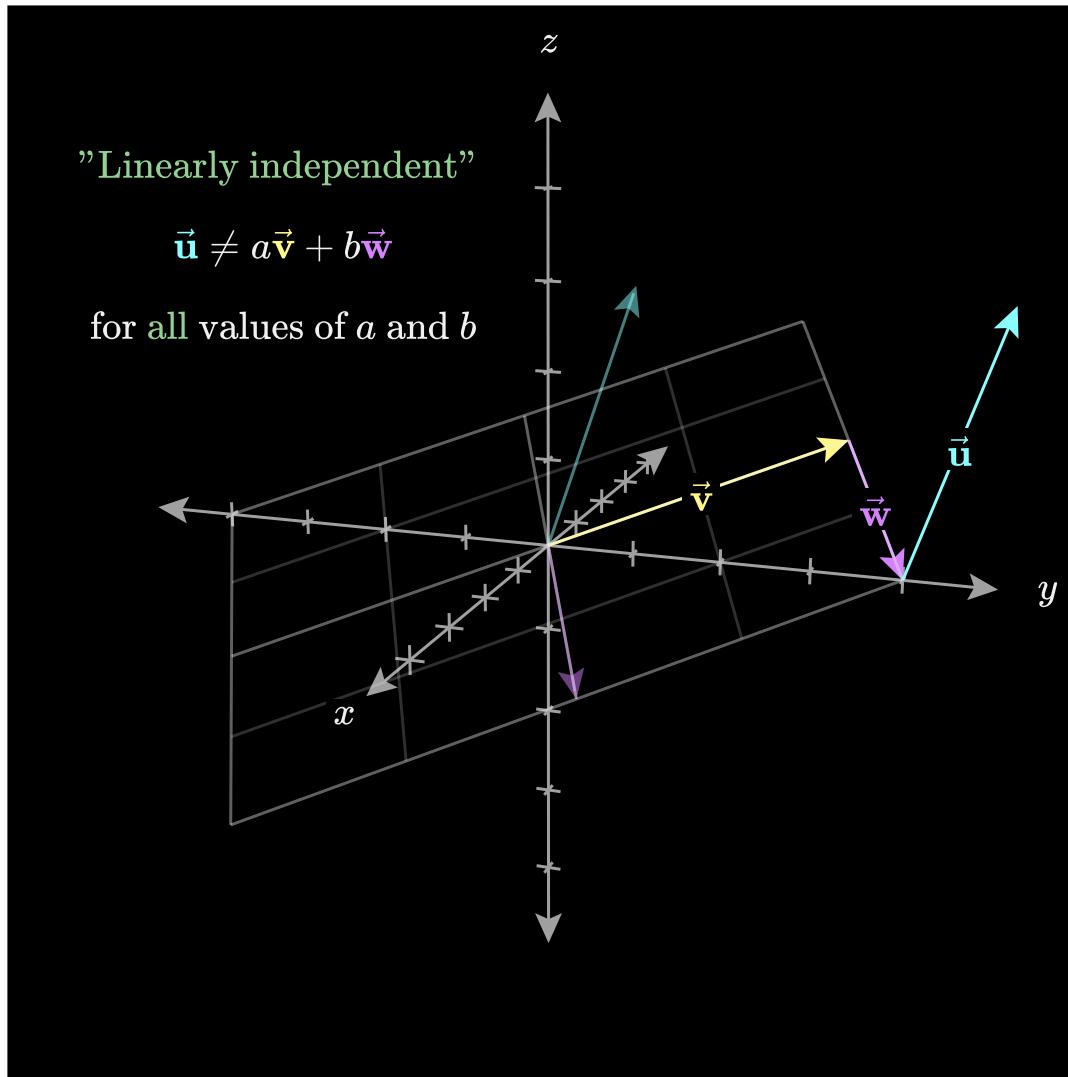
Linear dependence



Linear dependence



Linear independence



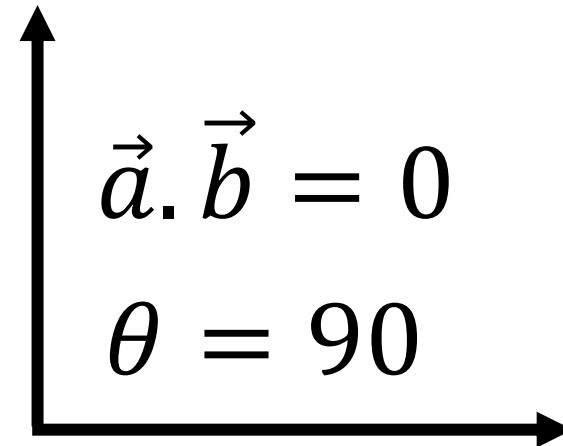
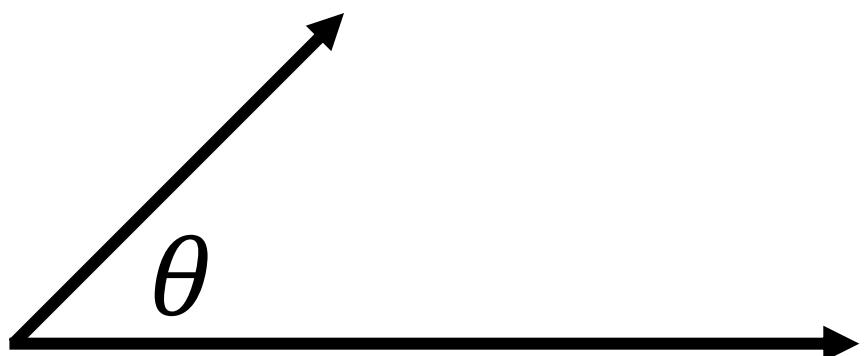
Vector operations: Dot product

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

$$\vec{a} \cdot \vec{b} = a^T b = \sum_{i=1}^n a_i b_i = c$$

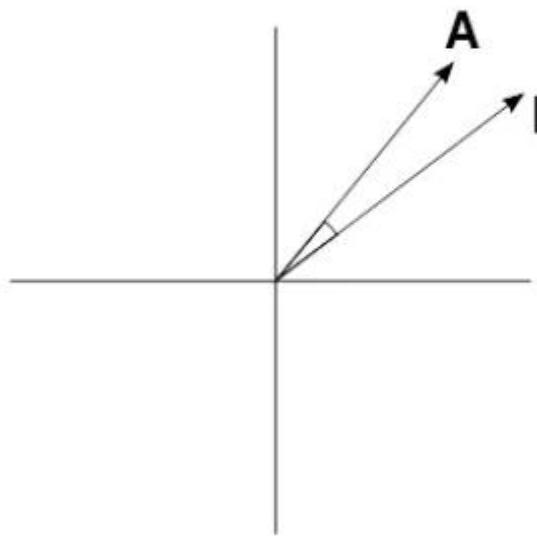
Dot product: a geometric interpretation

$$\vec{a} \cdot \vec{b} = a^T b = \sum_{i=1}^n a_i b_i = |a| |b| \cos(\theta) = c$$

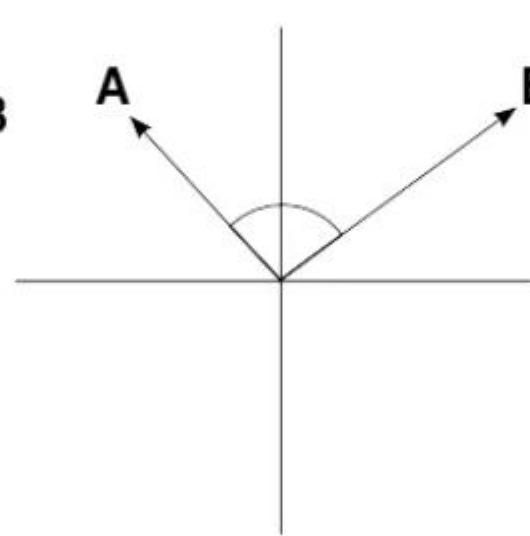


Cosine Similarity

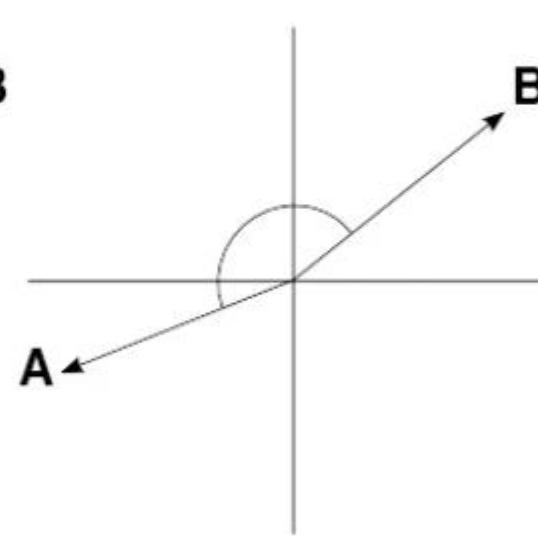
Similar



Unrelated

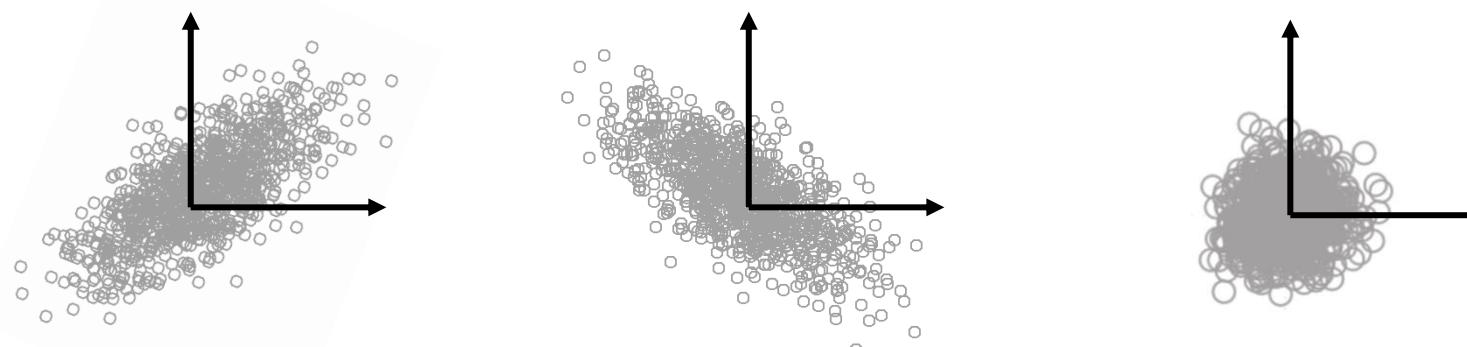


Opposite



Dot product and covariance

$$Cov_{x,y} = \frac{\sum (x_j - \bar{x})(y_j - \bar{y})}{N - 1} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$



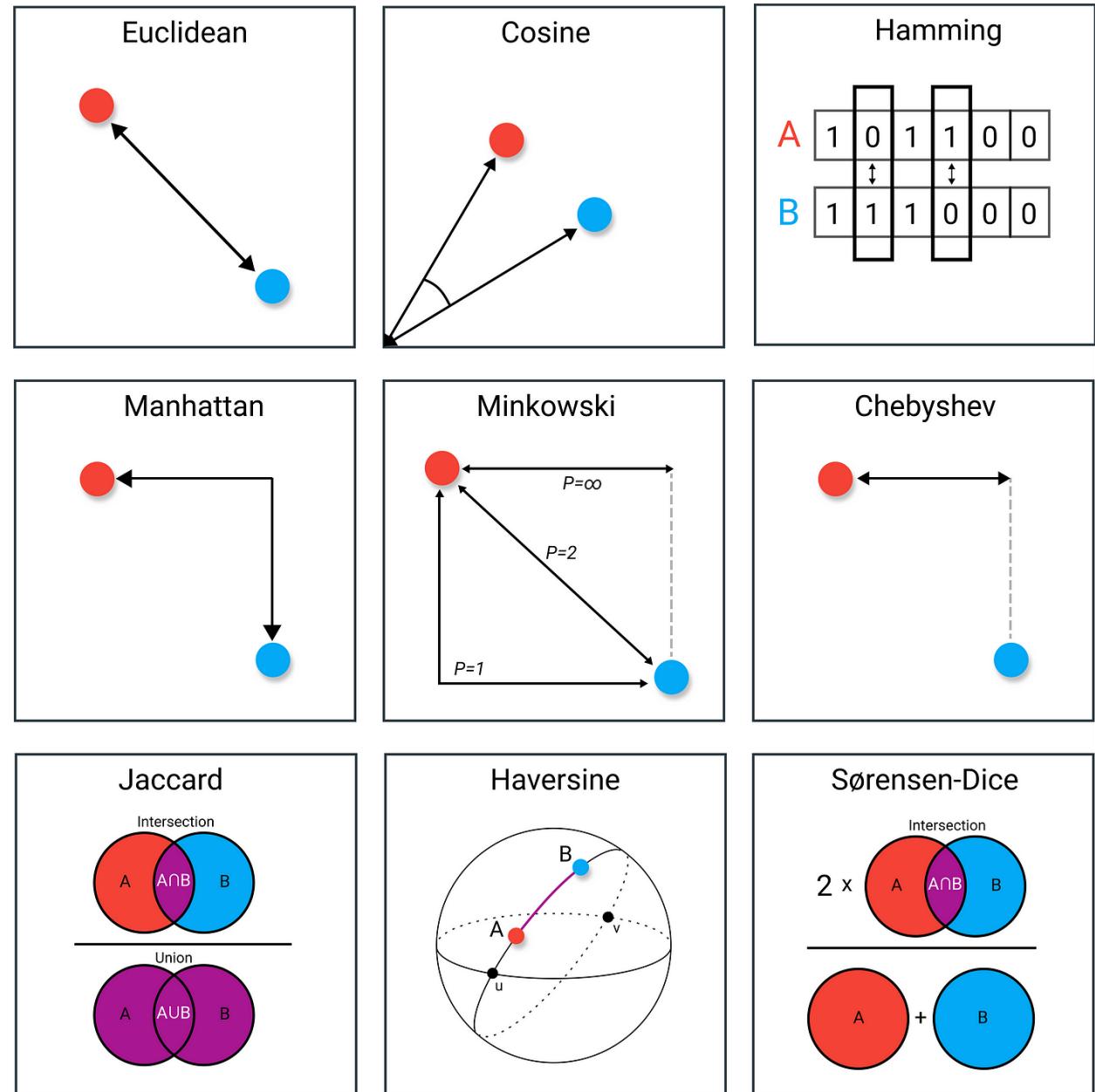
Dot product and correlation

$$Corr_{x,y} = \frac{\sum(x_j - \bar{x})(y_j - \bar{y})}{\sqrt{\sum(x_j - \bar{x})^2 \sum(y_j - \bar{y})^2}}$$

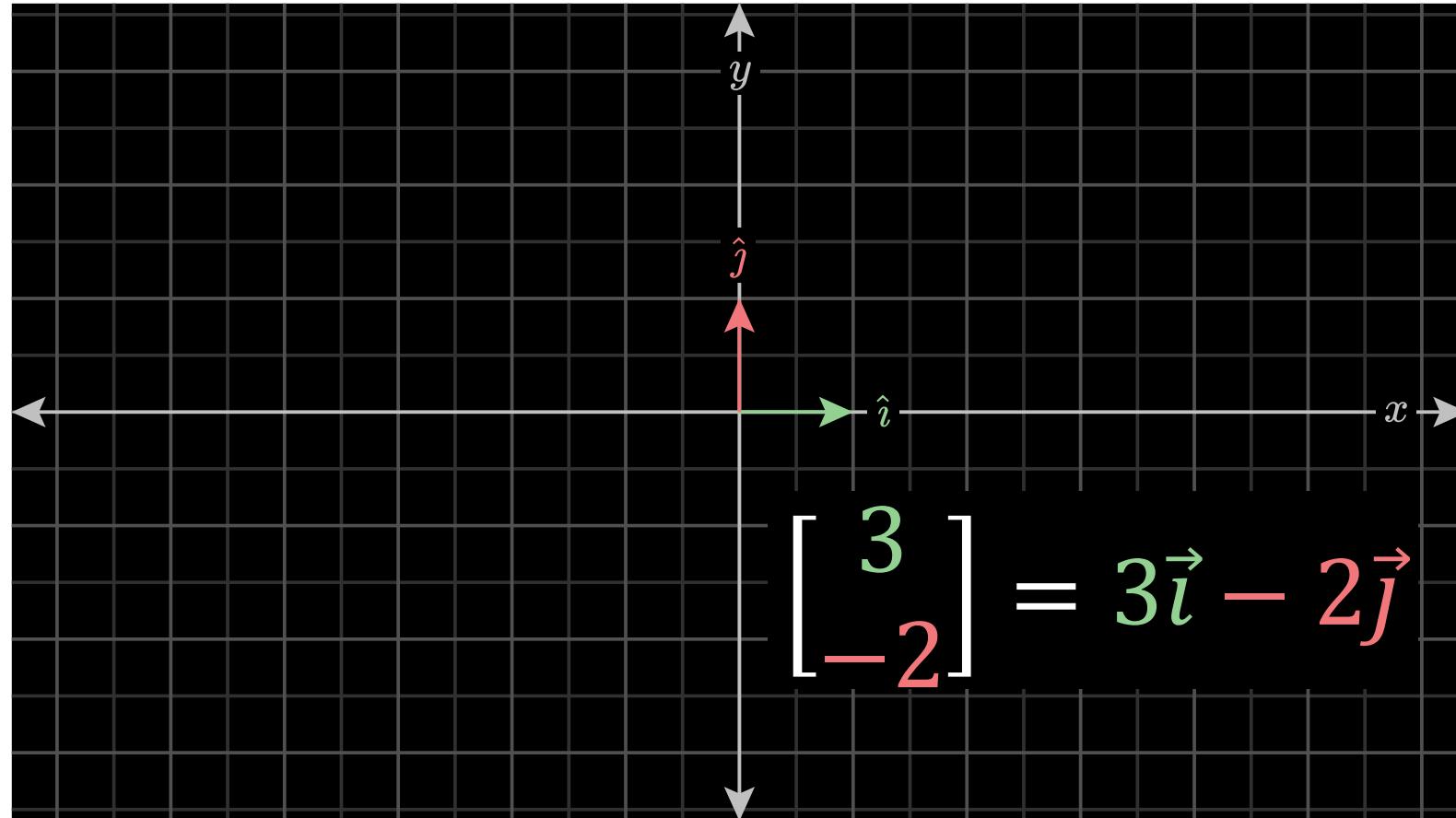
$$= \frac{\sum(x_j - \bar{x})(y_j - \bar{y})}{||x - \bar{x}|| ||y - \bar{y}||} \quad ||x|| = \sqrt{\sum_{i=1}^n x_i^2}$$

$$= \cosine(x - \bar{x}, y - \bar{y})$$

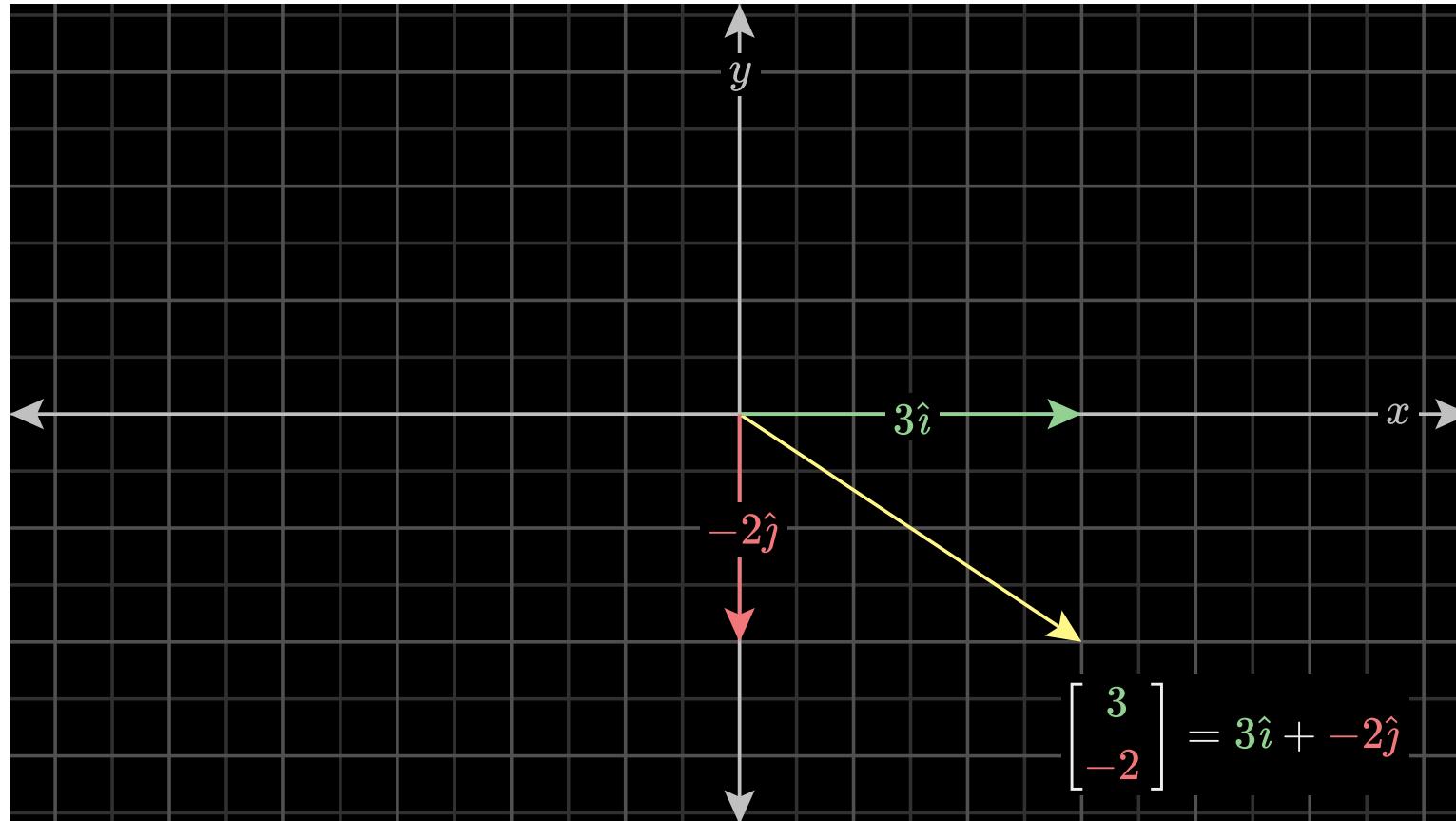
Distance Measures in Data Science



Basis vectors



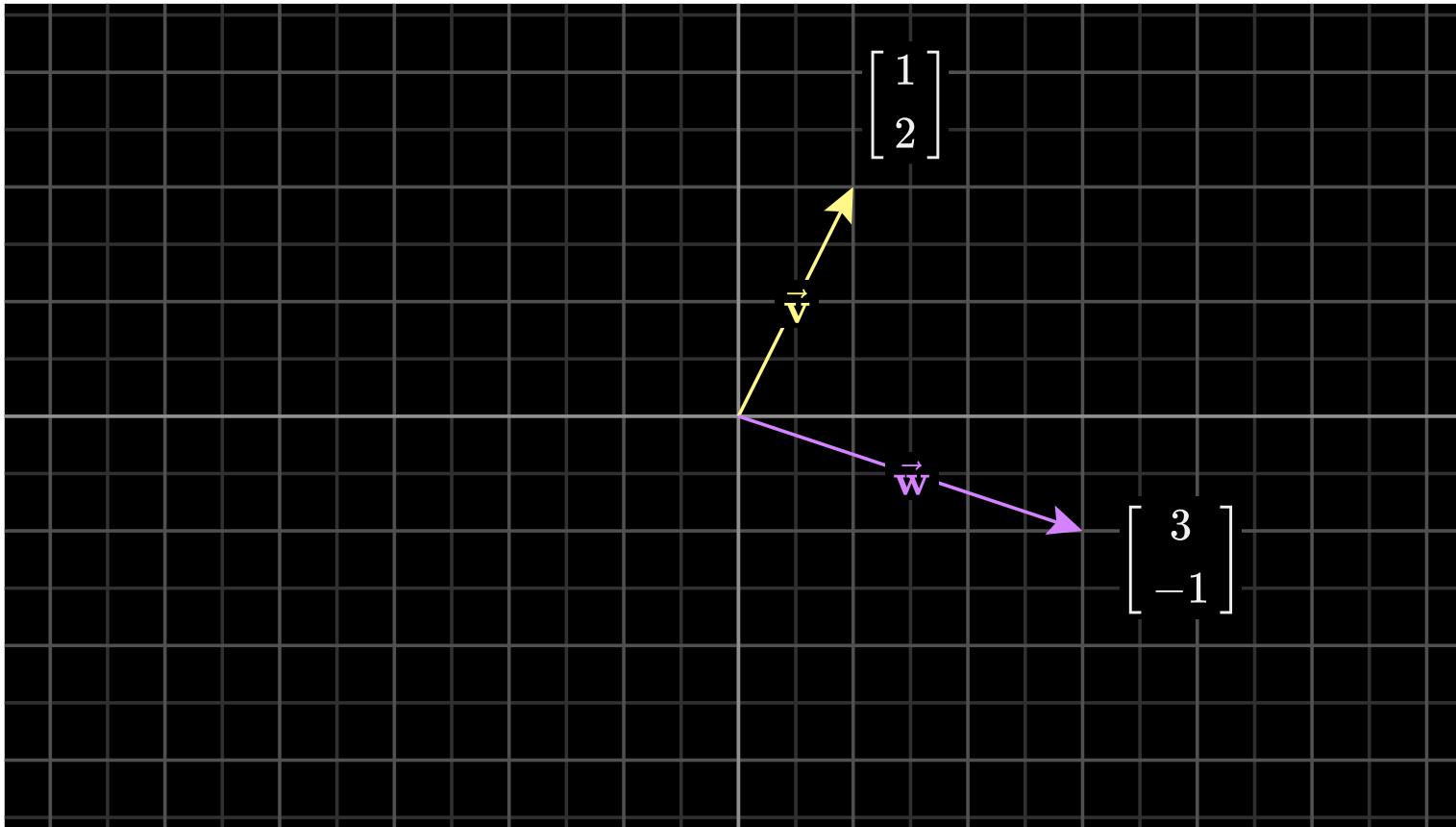
Basis vectors and span



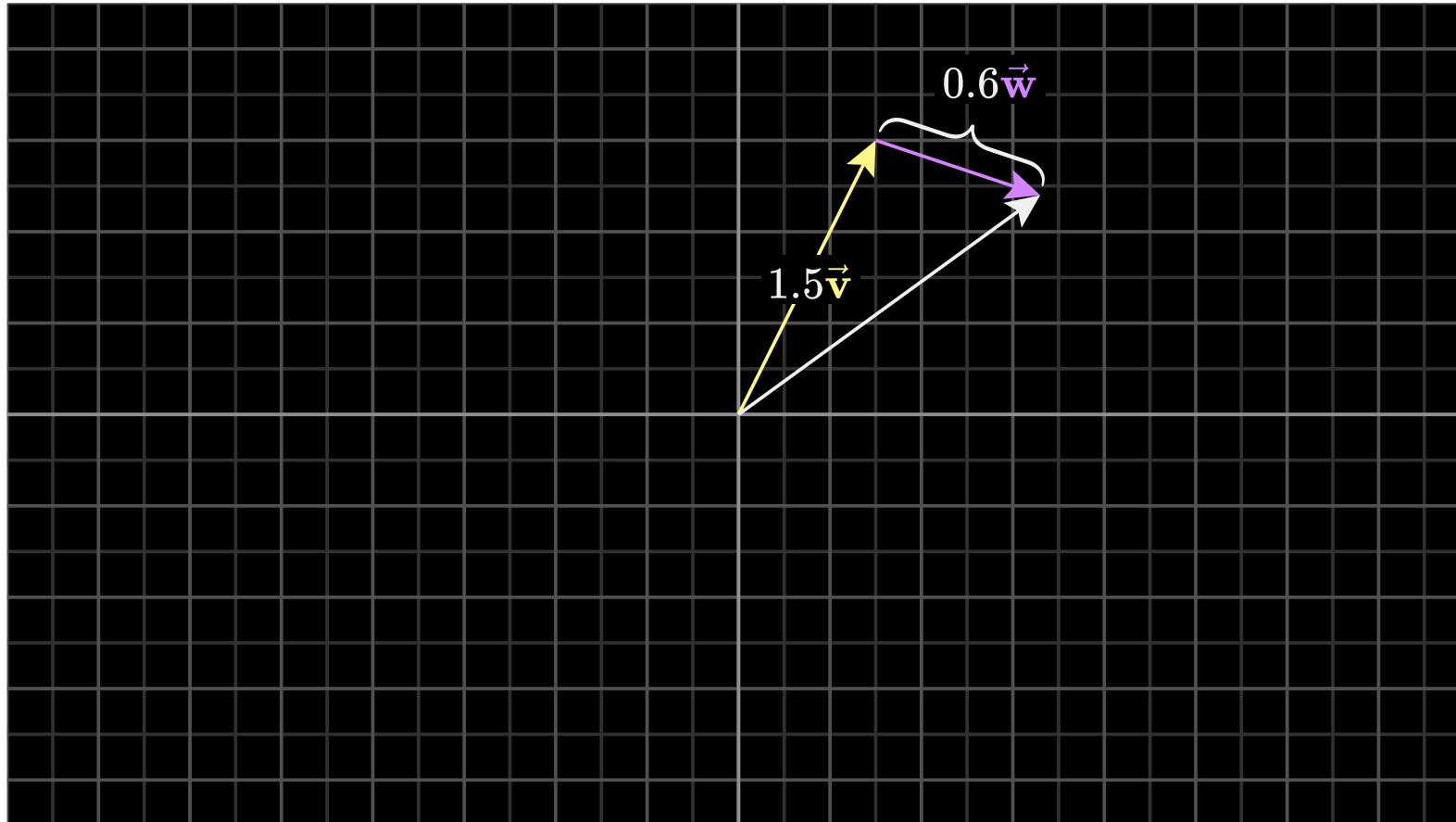
A basis of V is a list of vectors in V that is linearly independent and spans V .

In mathematics, the linear span of a set S of vectors is defined as the set of all linear combinations of the vectors in S .

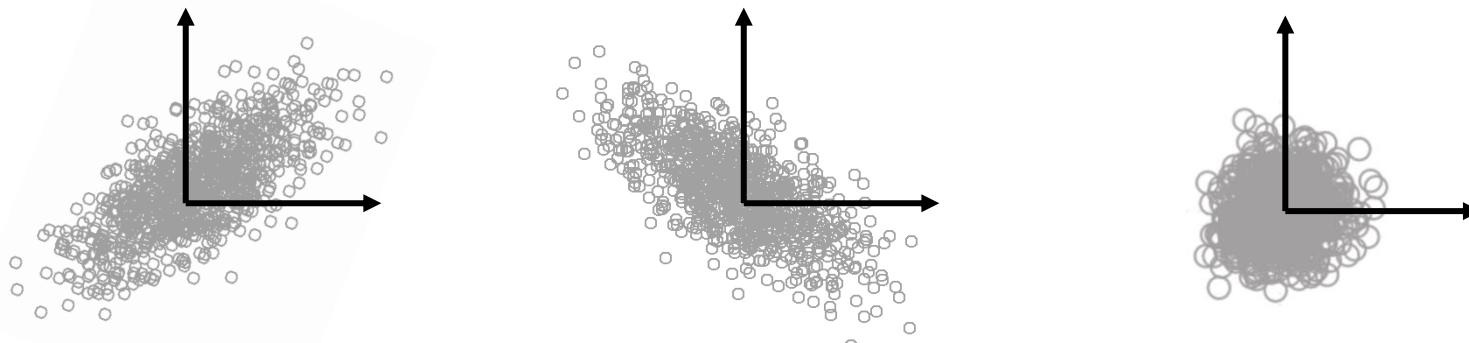
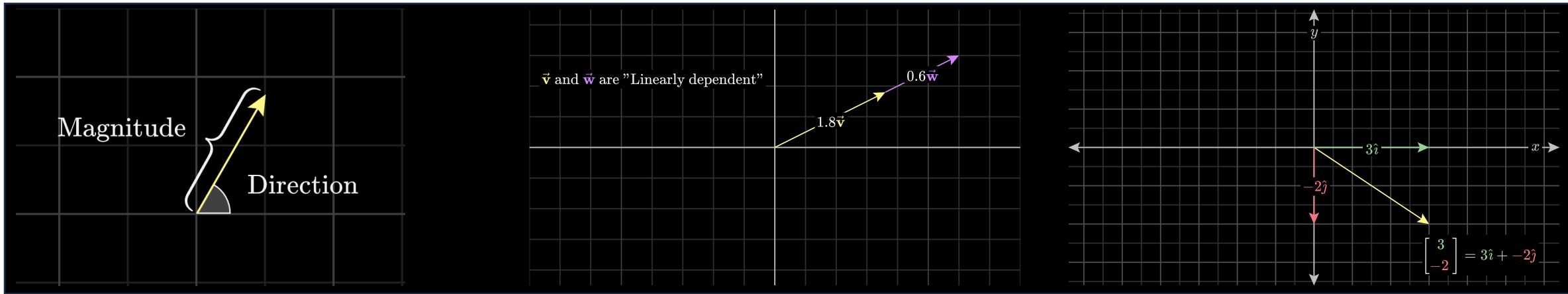
Choosing Different Basis Vectors



Choosing Different Basis Vectors



Summary



Matrix

Matrix

“ 2x2 Matrix ”

$$\begin{bmatrix} 1 & 3 \\ -2 & 0 \end{bmatrix}$$

Where \hat{i} lands

Where \hat{j} lands

Matrix

A rectangular array of numbers is called a matrix. The horizontal arrays of a matrix are called its rows and the vertical arrays are called its columns. A matrix A having m rows and n columns is said to be a matrix of size/ order $m \times n$ and can be represented in either of the following forms:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}_{m \times n}$$

Matrix Dimension

Matrix Operations: Addition

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

a matrix can only be added to another matrix if the two matrices have the same dimensions .

Matrix operations:

Scalar Multiplication

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

Matrix operations: multiplication

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x \begin{bmatrix} a \\ c \end{bmatrix} + y \begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

Matrix operations: multiplication

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & g \\ f & h \end{bmatrix} = \begin{bmatrix} ae + bf & ag + bh \\ ce + df & cg + dh \end{bmatrix}$$

Matrix operations: multiplication

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & g \\ f & h \end{bmatrix} = \begin{bmatrix} ae + bf & ag + bh \\ ce + df & cg + dh \end{bmatrix}$$

Matrix operations: multiplication

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \xrightarrow{\hspace{1cm}} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

You can only multiply matrices if the number of columns of the first matrix is the same as the number of rows as the second matrix: $A_{m \times n} \times B_{n \times p} = C_{m \times p}$

Matrix operations: multiplication

we can operationally define matrix multiplication as the dot product of the rows of the first matrix with the columns of the second matrix.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \xrightarrow{\hspace{1cm}} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

Matrix operations: Transpose

 A

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

 A^T

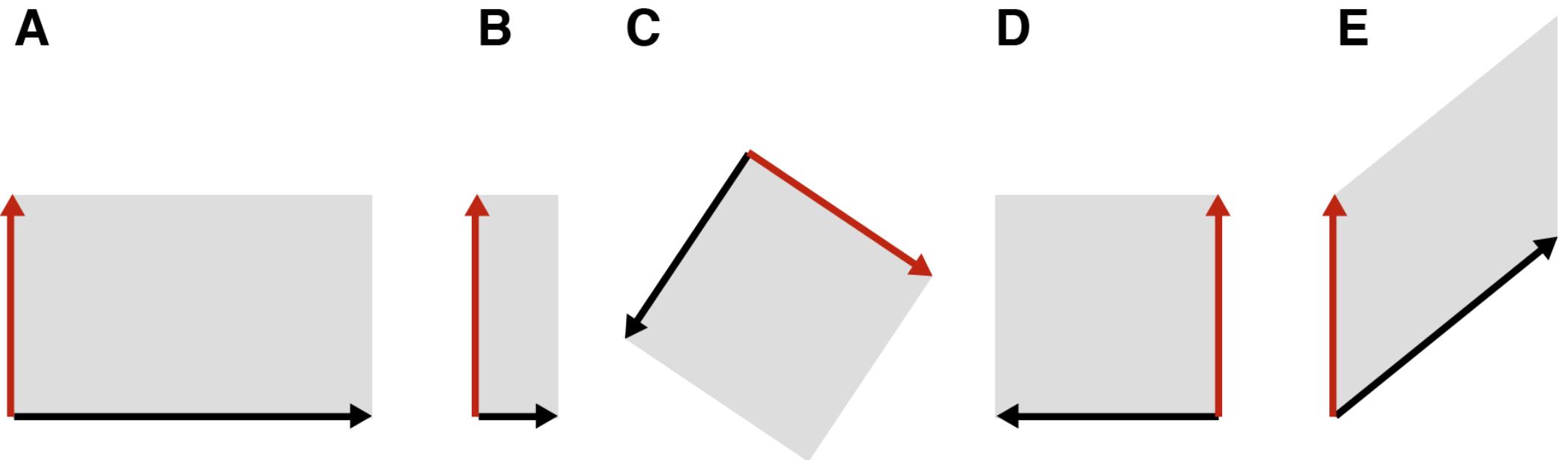
$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

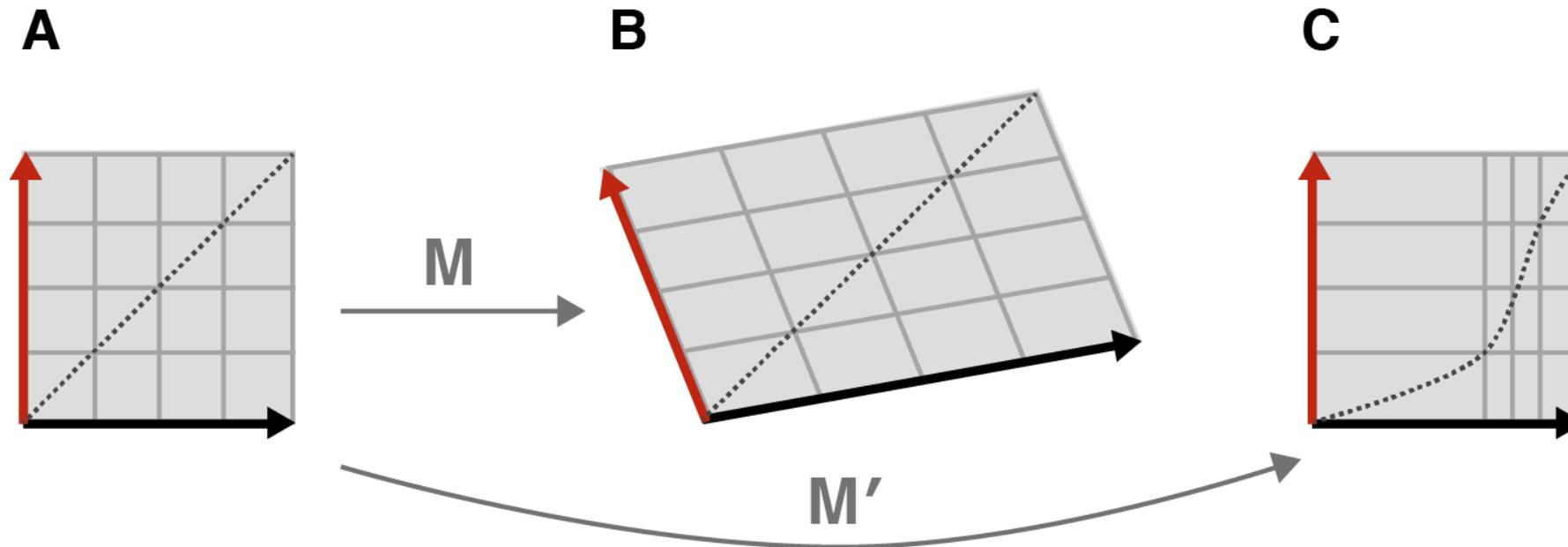
Linear transformations

Transformation



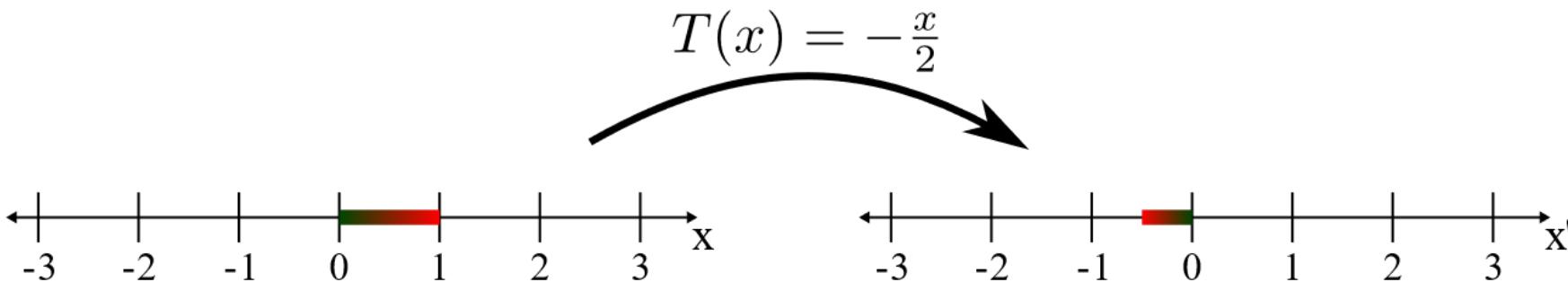
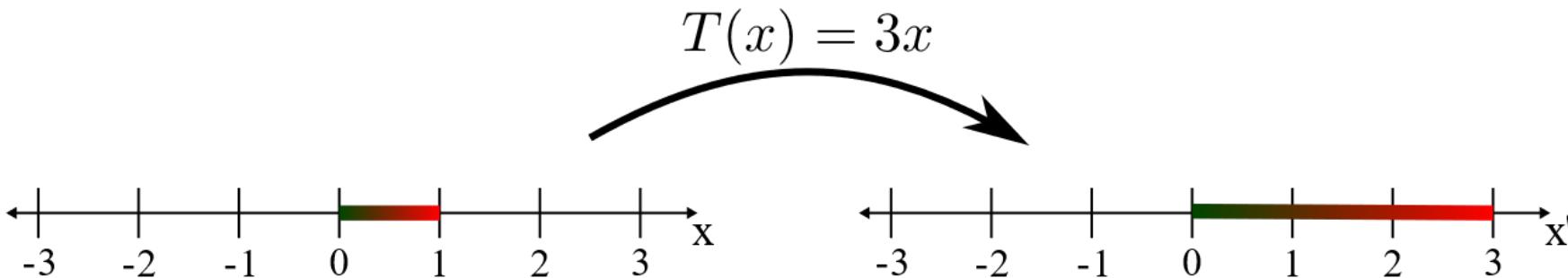
Our original square under different types of transformations: (A) stretched, (B) compressed, (C) rotated, (D) reflected or flipped, and (E) sheared.

Linear and Nonlinear Transformation

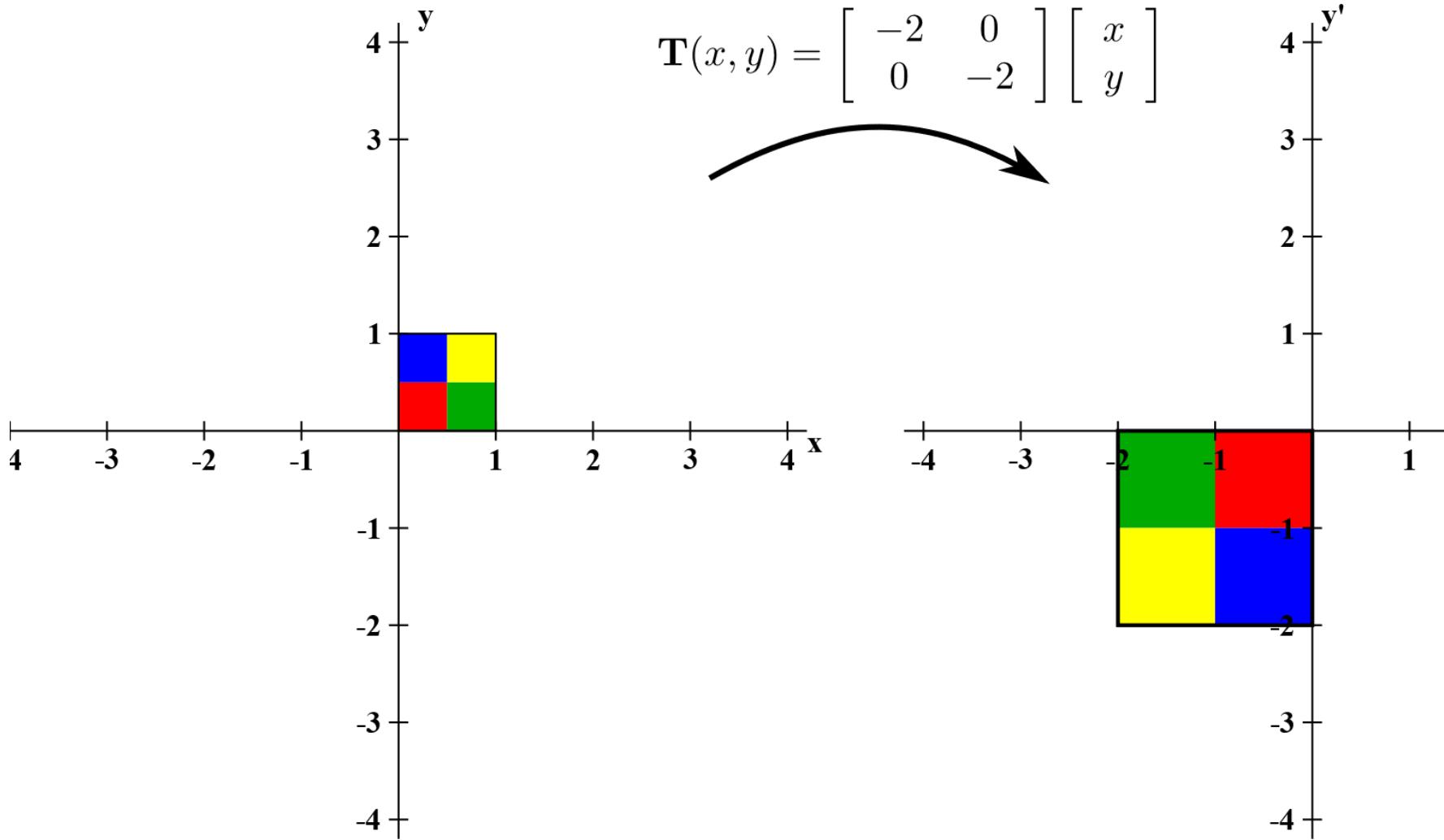


(A) Our original square under a linear transformation M (B) and a nonlinear transformation M (C).

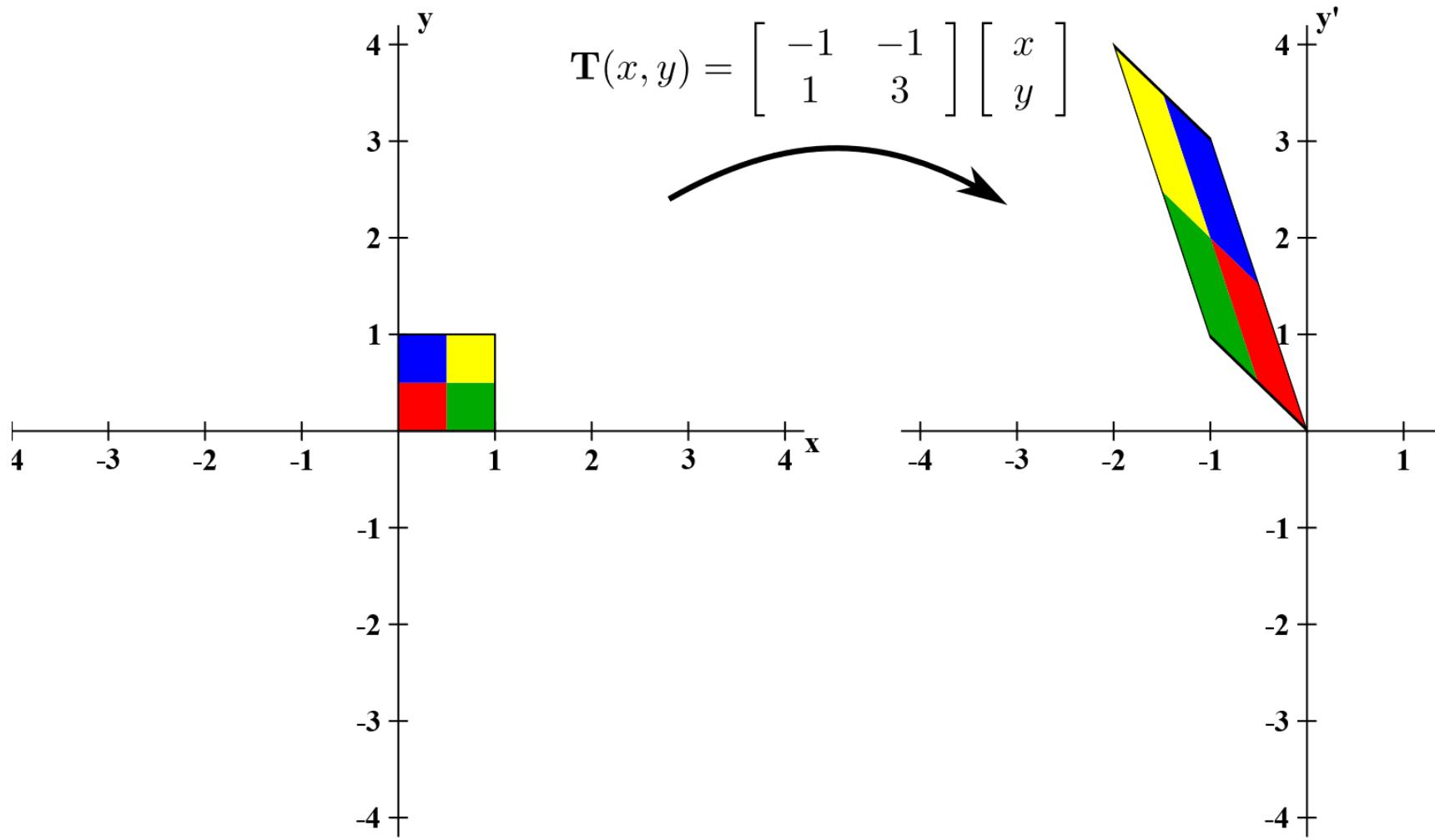
Linear Transformation



Linear Transformation

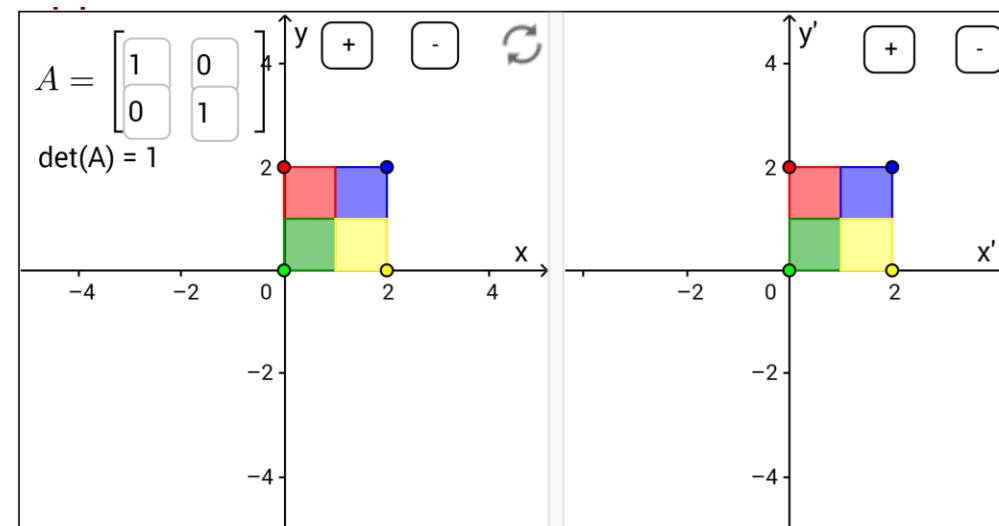


Linear Transformation



Identity Matrix

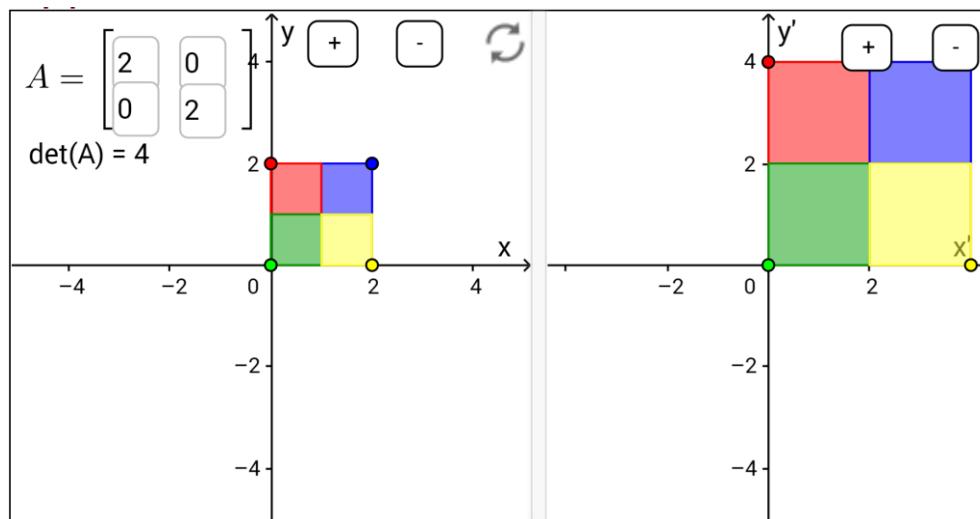
$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a & \cdots & b \\ \vdots & \ddots & \vdots \\ c & \cdots & d \end{bmatrix} = \begin{bmatrix} a & \cdots & b \\ \vdots & \ddots & \vdots \\ c & \cdots & d \end{bmatrix}$$



https://mathinsight.org/applet/linear_transformation_2d
<https://integral-domain.org/lwilliams/Applets/algebra/linearTransformations.php>

Scalar Matrix

$$\begin{bmatrix} C & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C \end{bmatrix} \begin{bmatrix} a & \cdots & b \\ \vdots & \ddots & \vdots \\ c & \cdots & d \end{bmatrix} = \begin{bmatrix} Ca & \cdots & Cb \\ \vdots & \ddots & \vdots \\ Cc & \cdots & Cd \end{bmatrix}$$

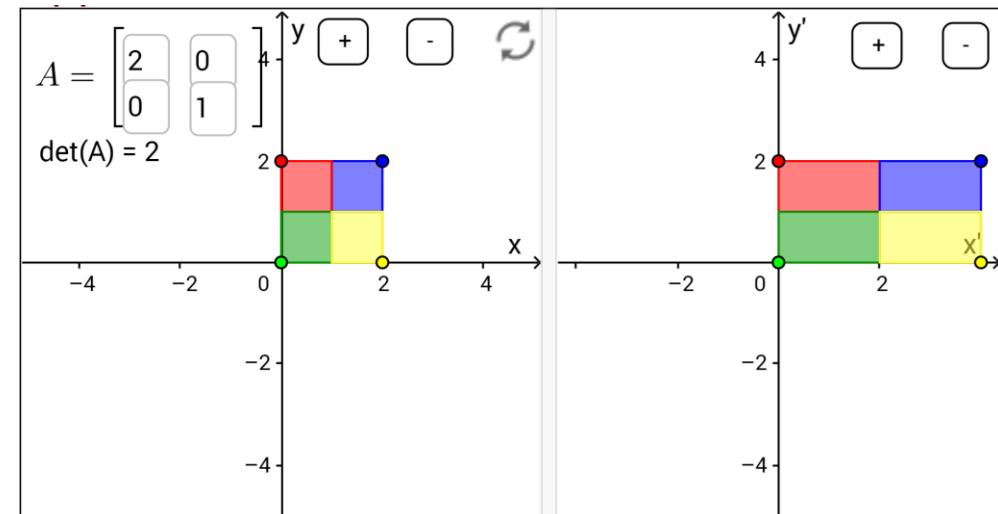
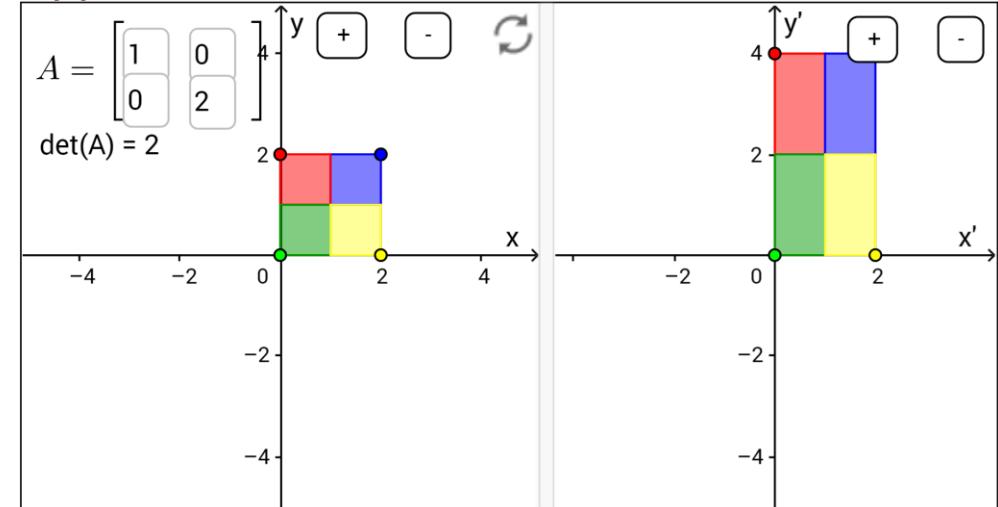


https://mathinsight.org/applet/linear_transformation_2d

<https://integral-domain.org/lwilliams/Applets/algebra/linearTransformations.php>

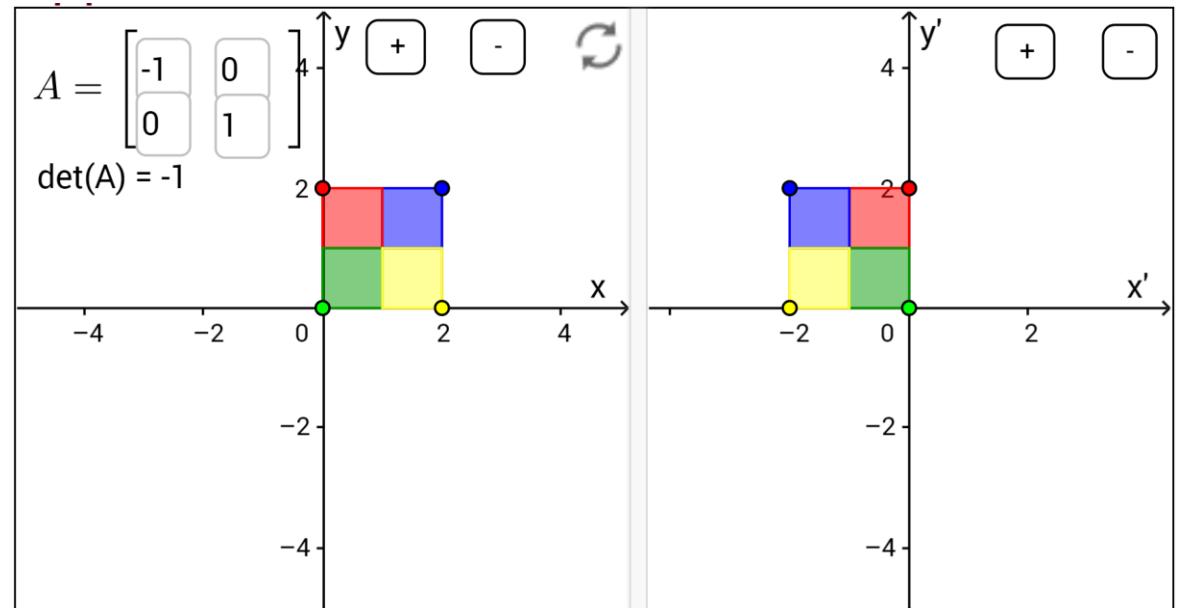
Off by one matrix

$$\begin{bmatrix} 2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} a & \dots & b \\ \vdots & \ddots & \vdots \\ c & \dots & d \end{bmatrix}$$



Off by one matrix

$$\begin{bmatrix} -1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a & \cdots & b \\ \vdots & \ddots & \vdots \\ c & \cdots & d \end{bmatrix}$$



https://mathinsight.org/applet/linear_transformation_2d

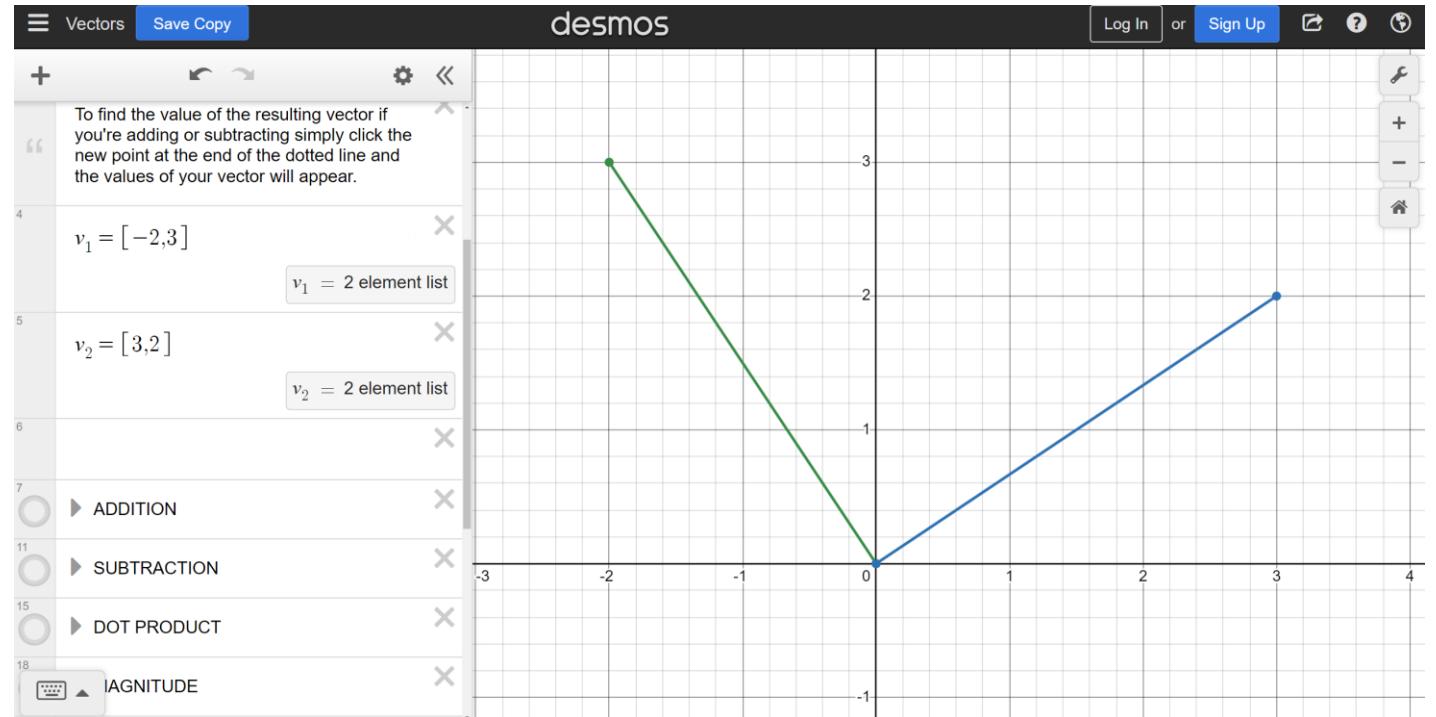
<https://integral-domain.org/lwilliams/Applets/algebra/linearTransformations.php>

Orthogonal Matrix

Square matrix

All column vectors are orthogonal

$$\begin{bmatrix} -2 & 3 \\ 3 & 2 \end{bmatrix}$$

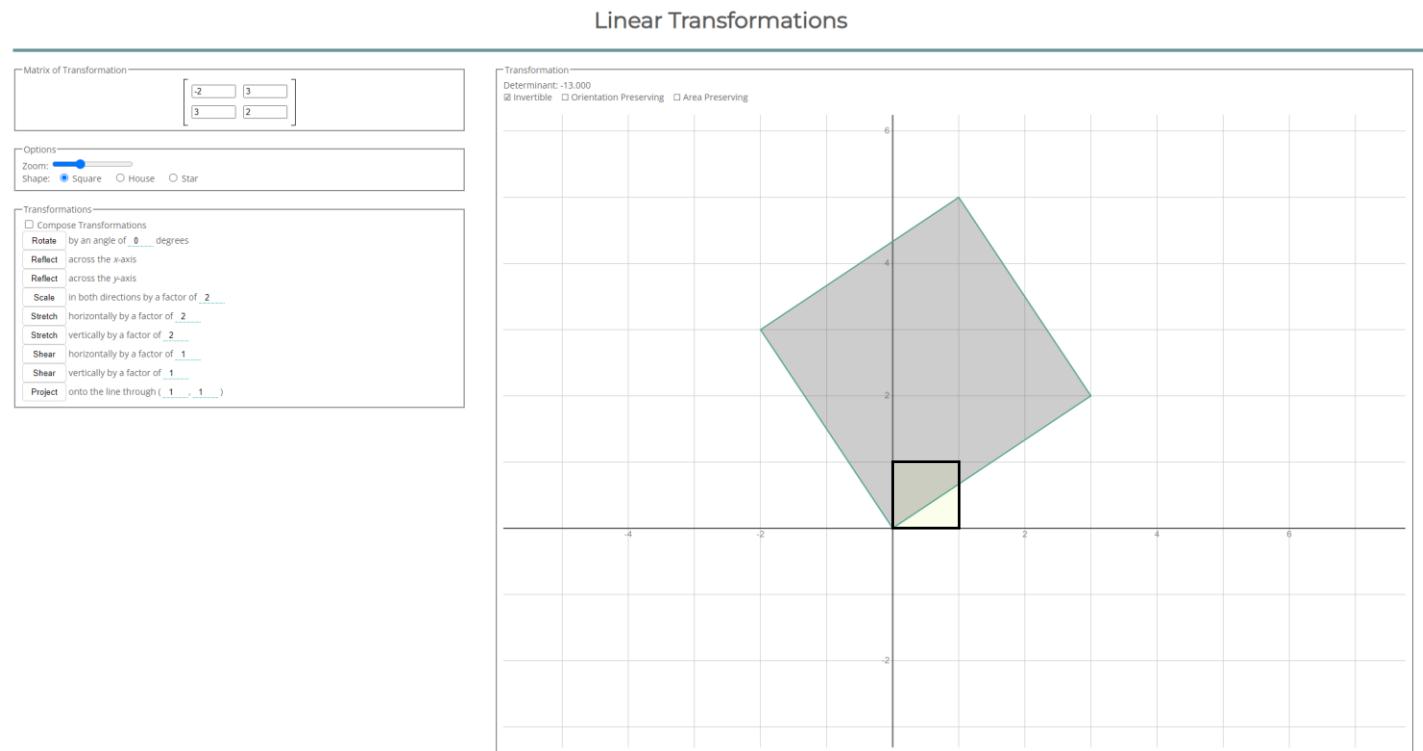


Orthogonal Matrix

Square matrix

All column vectors are orthogonal

$$\begin{bmatrix} -2 & 3 \\ 3 & 2 \end{bmatrix}$$



Visualize Different Matrices part2 | SEE Matrix, Chapter 1

<https://www.desmos.com/calculator/u8wt5rnw9n>

<https://integral-domain.org/lwilliams/Applets/algebra/linearTransformations.php>

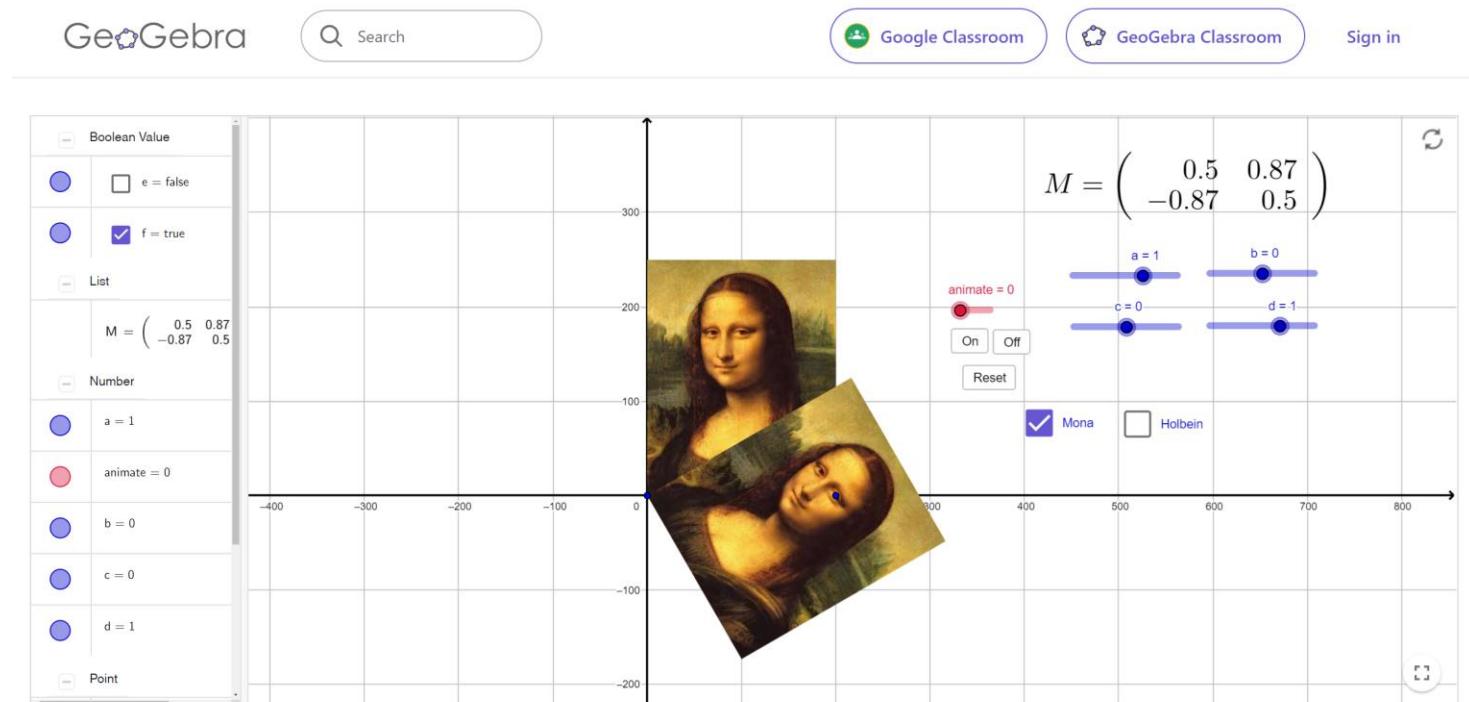
Orthogonal Matrix

Square matrix

All column vectors are orthogonal

All column vectors are unit vectors

$$\begin{bmatrix} 0.5 & \sqrt{3}/2 \\ -\sqrt{3}/2 & 0.5 \end{bmatrix}$$



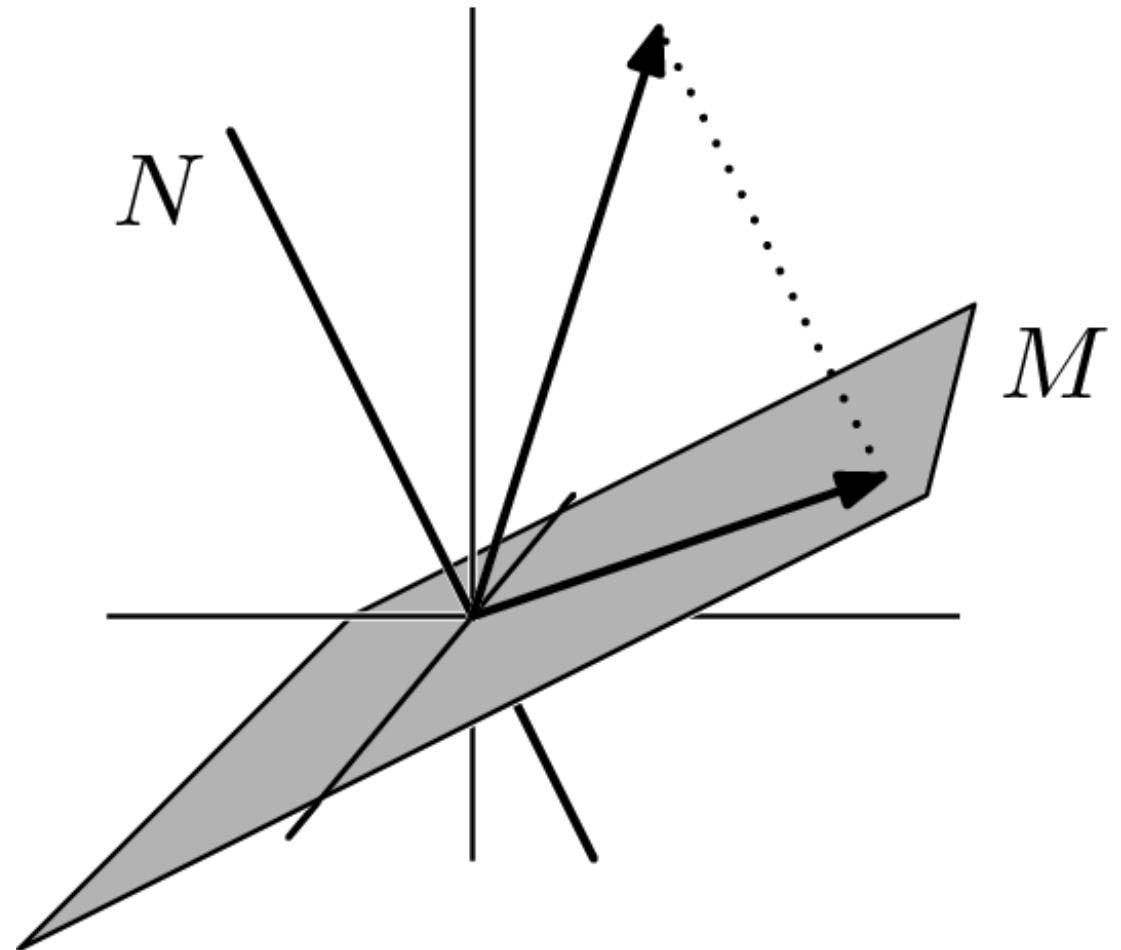
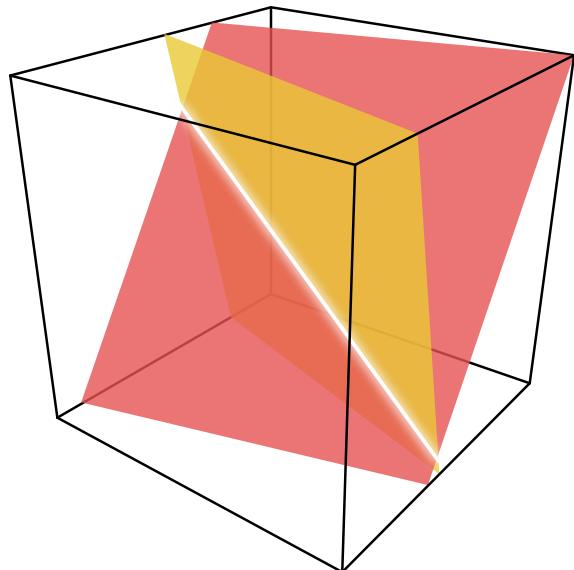
Visualize Different Matrices part2 | SEE Matrix, Chapter 1

<https://www.desmos.com/calculator/u8wt5rnw9n>

<https://www.geogebra.org/m/pDU4peV5>

Projection Matrix

A projection matrix is a special type of square matrix that, when multiplied by a vector, projects that vector onto a certain subspace.



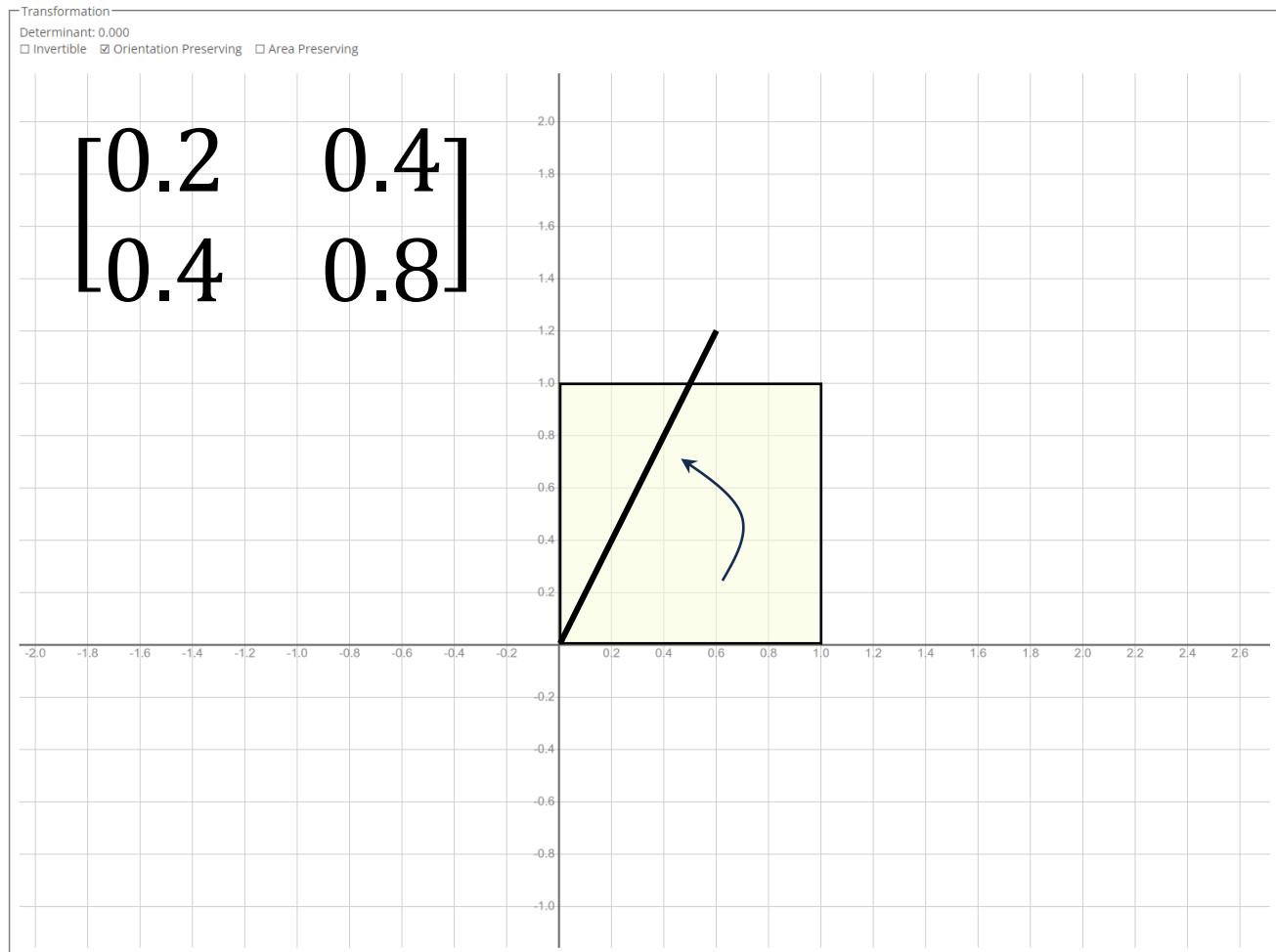
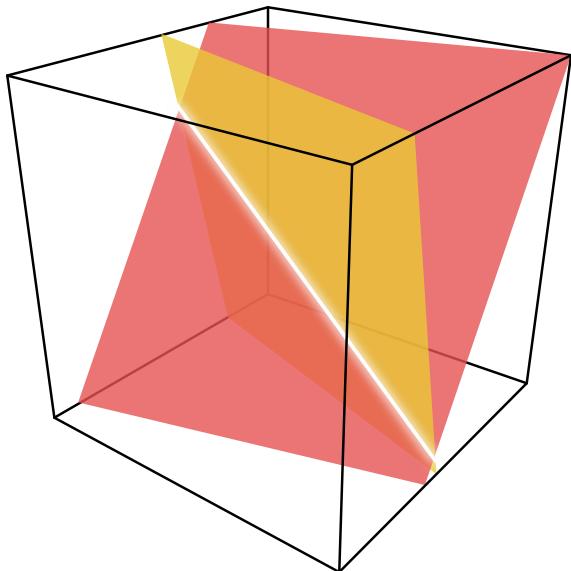
Visualize Different Matrices part2 | SEE Matrix, Chapter 1

<https://www.desmos.com/calculator/u8wt5rnw9n>

<https://integral-domain.org/lwilliams/Applets/algebra/linearTransformations.php>

Projection Matrix

A projection matrix is a special type of square matrix that, when multiplied by a vector, projects that vector onto a certain subspace.



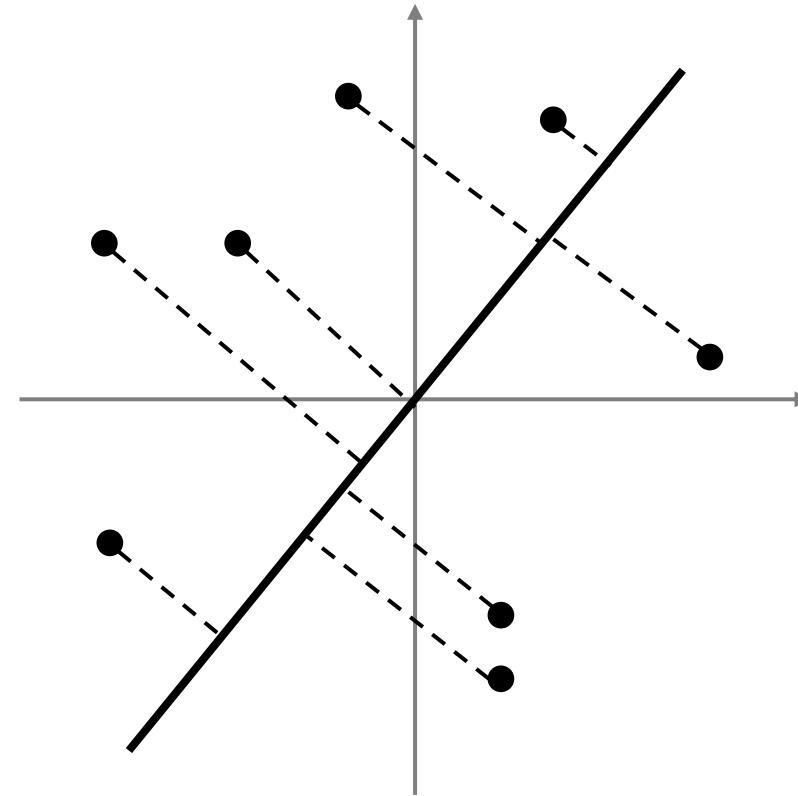
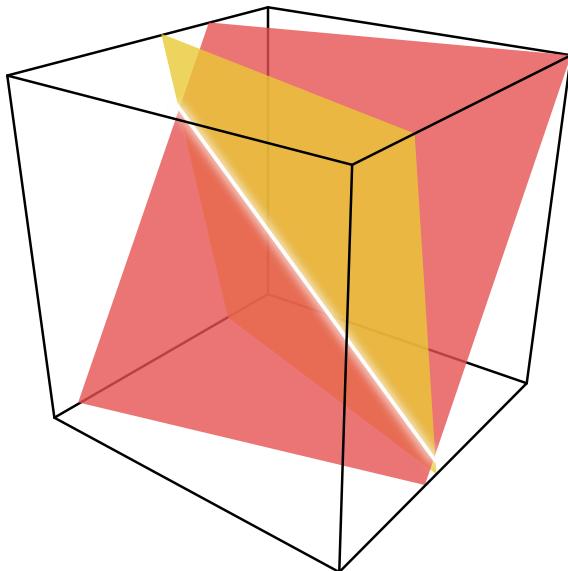
Visualize Different Matrices part2 | SEE Matrix, Chapter 1

<https://www.desmos.com/calculator/u8wt5rnw9n>

<https://integral-domain.org/lwilliams/Applets/algebra/linearTransformations.php>

Projection Matrix

A projection matrix is a special type of square matrix that, when multiplied by a vector, projects that vector onto a certain subspace.

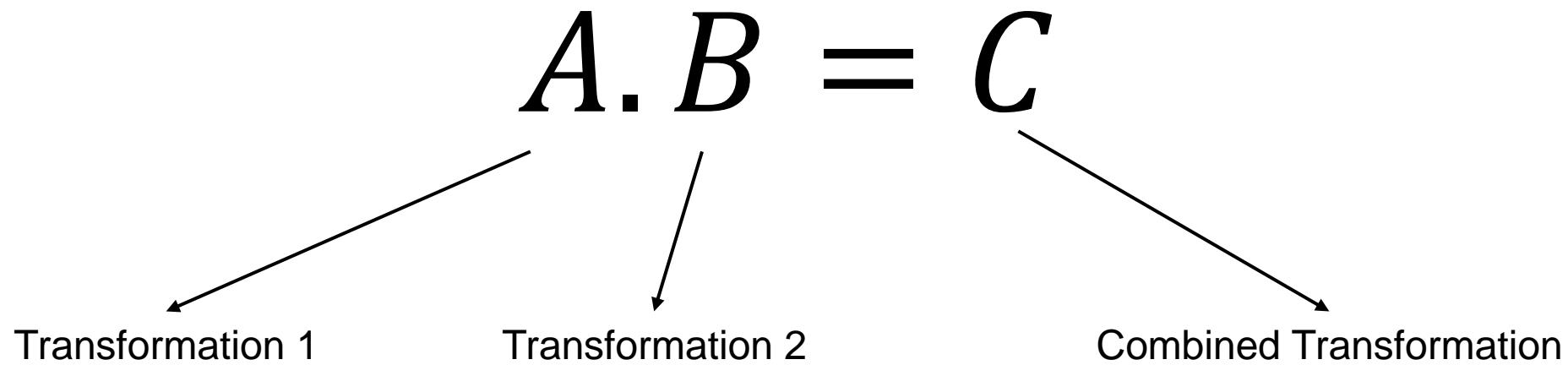


Visualize Different Matrices part2 | SEE Matrix, Chapter 1

<https://www.desmos.com/calculator/u8wt5rnw9n>

<https://integral-domain.org/lwilliams/Applets/algebra/linearTransformations.php>

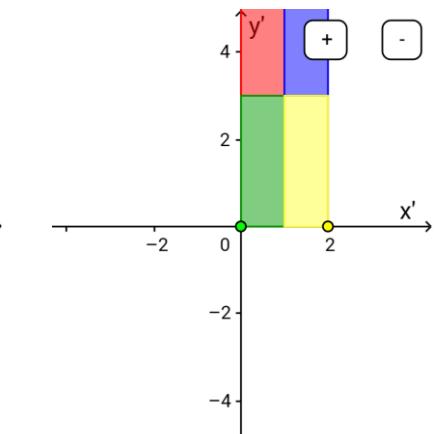
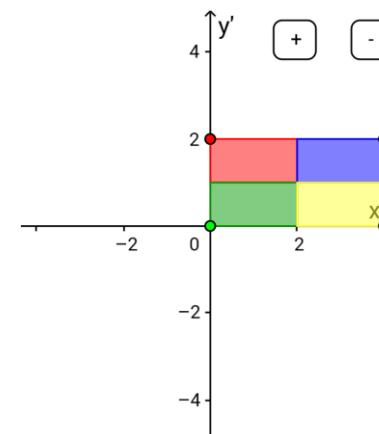
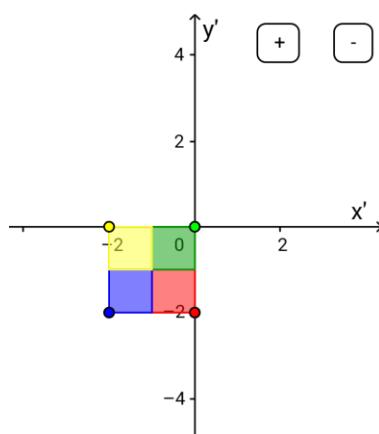
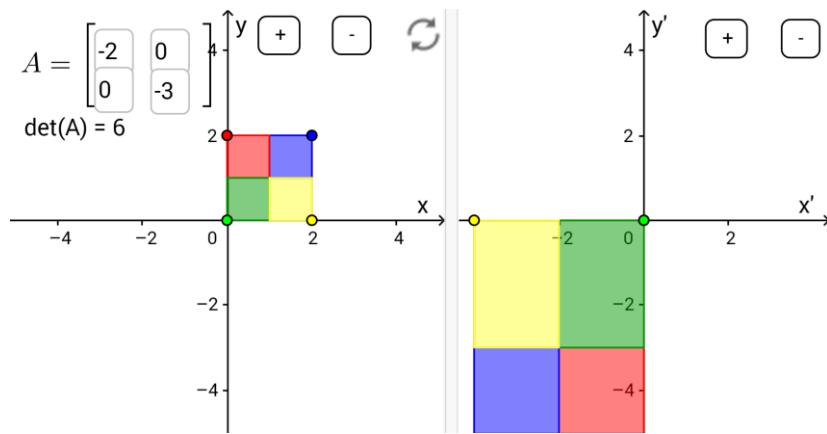
Composition of linear transformations



Matrix multiplication: Composition of Sequential Transformations

Composition of linear transformations

$$\begin{bmatrix} -2 & 0 \\ 0 & -3 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$



Linear transformations is a function

Linear Transformation

$$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

Input

$$L(\vec{\mathbf{v}})$$

$$\begin{bmatrix} 7 \\ 2 \end{bmatrix}$$

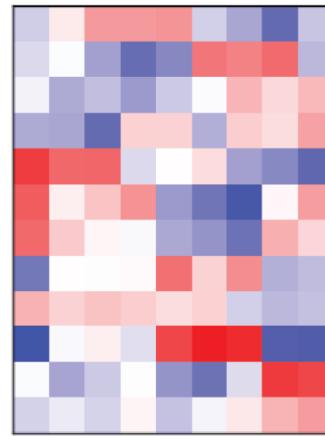
Output

Linear transformations is a function

$$\begin{bmatrix} x_{\text{in}} \\ y_{\text{in}} \end{bmatrix} \rightarrow \text{???} \rightarrow \begin{bmatrix} x_{\text{out}} \\ y_{\text{out}} \end{bmatrix}$$

Input Output

Original Data

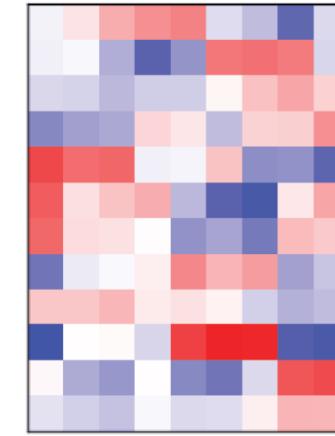


\approx

Components

$$\text{Loadings} \times \text{Components} = \text{Reconstruction}$$

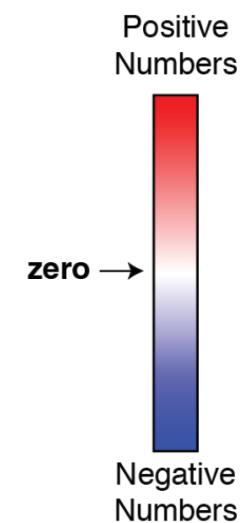
Reconstruction



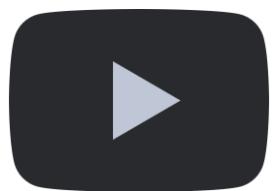
Matrix decomposition

Sum of
Rank-1
Matrices

$$\text{Sum of Rank-1 Matrices} = \text{Matrix 1} + \text{Matrix 2} + \text{Matrix 3}$$



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax

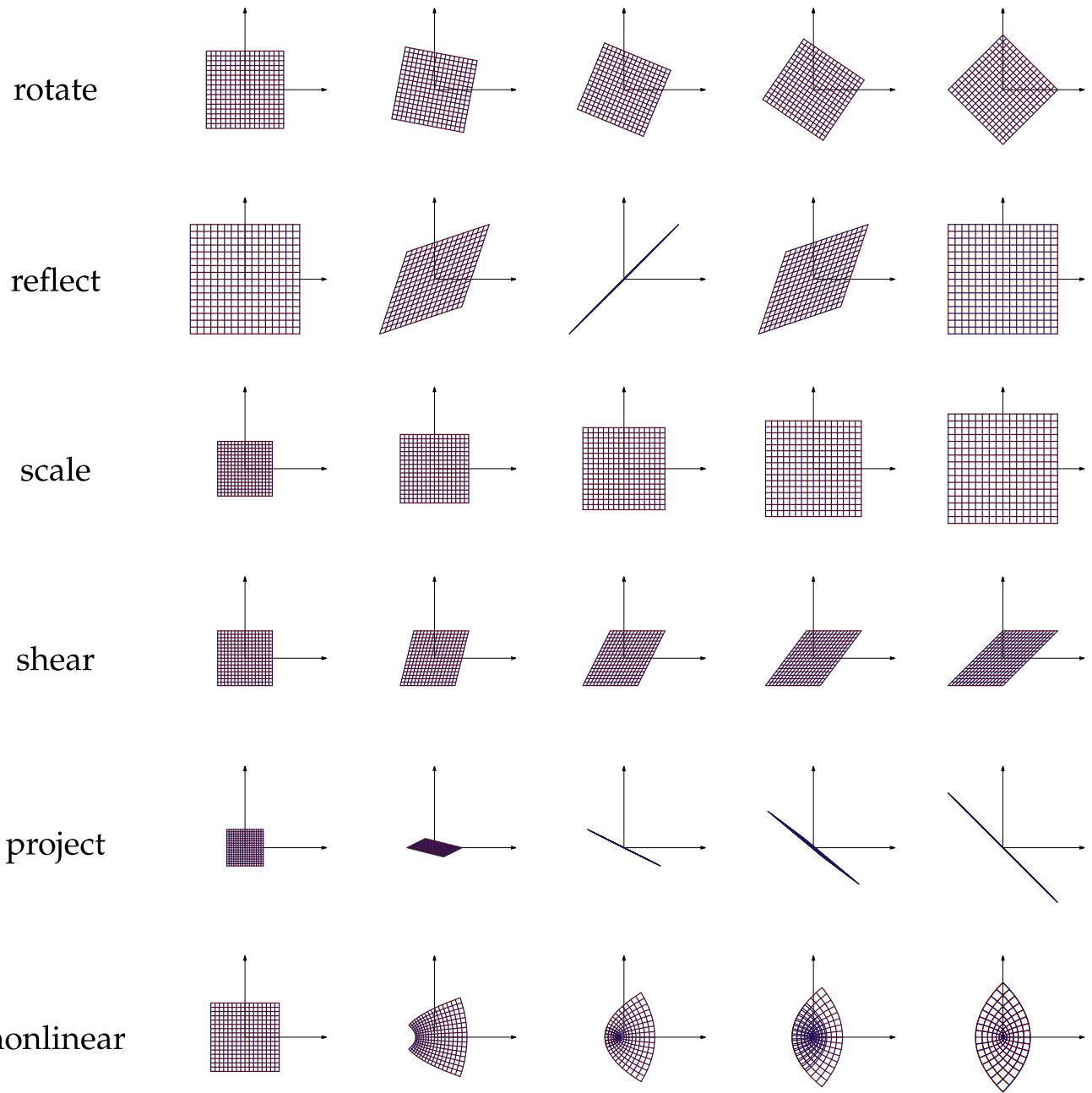


neurosyntax

Session 2

Lecturer: Saman Abbaspoor

Summary: Matrix as a Linear transformations

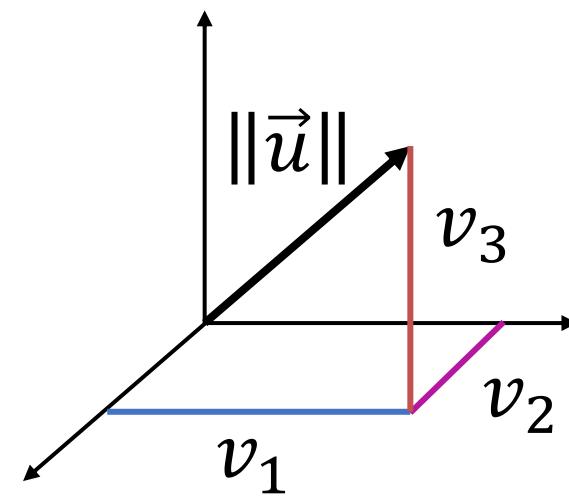
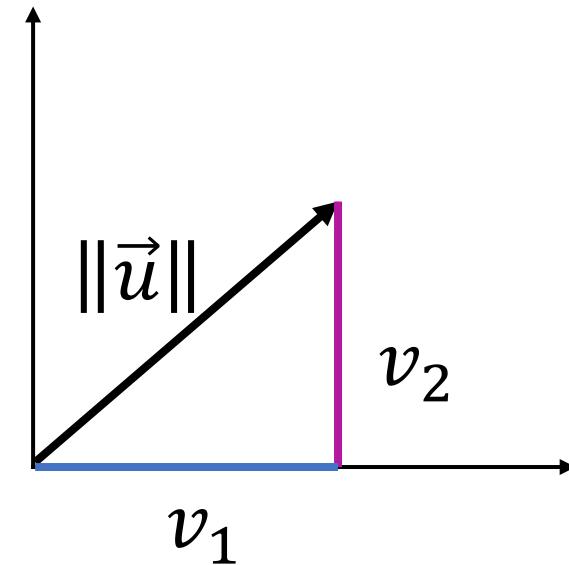


Magnitude of a vector (Euclidean norm)

$$\|\vec{u}\| = \sqrt{v_1^2 + v_2^2}$$

$$\|\vec{u}\| = \sqrt{v_1^2 + v_2^2 + v_3^2}$$

$$\|\vec{u}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \sqrt{\sum_{i=1}^n v_i^2}$$



Euclidean norm and variance

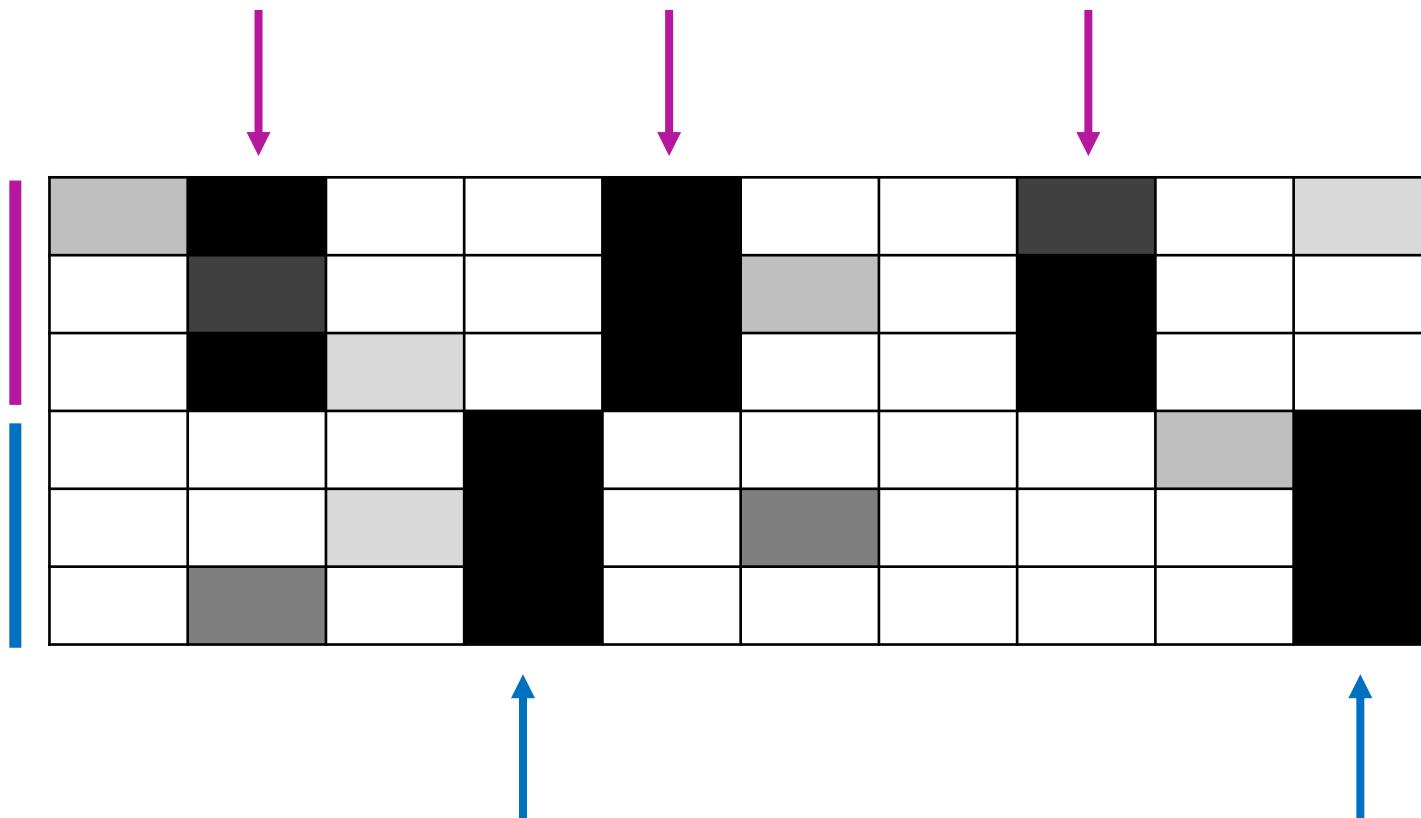
$$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \sqrt{\sum_{i=1}^n v_i^2}$$
$$s^2 = \frac{\sum_{i=1}^n (v_i - \bar{v})^2}{n - 1}$$
$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

$$\frac{1}{n} \|\vec{v}\|^2 = s^2$$

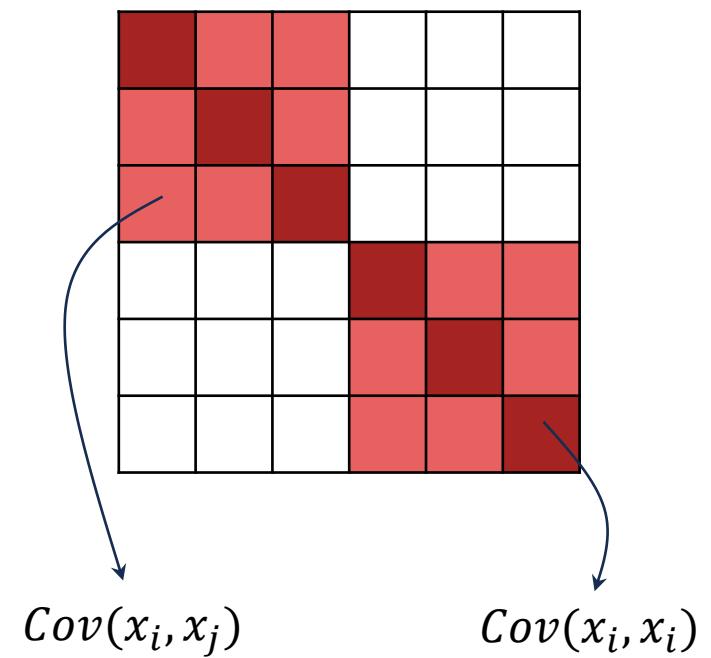
Matrix decomposition and principal component analysis (PCA)

Patterns: correlated structures

Data matrix

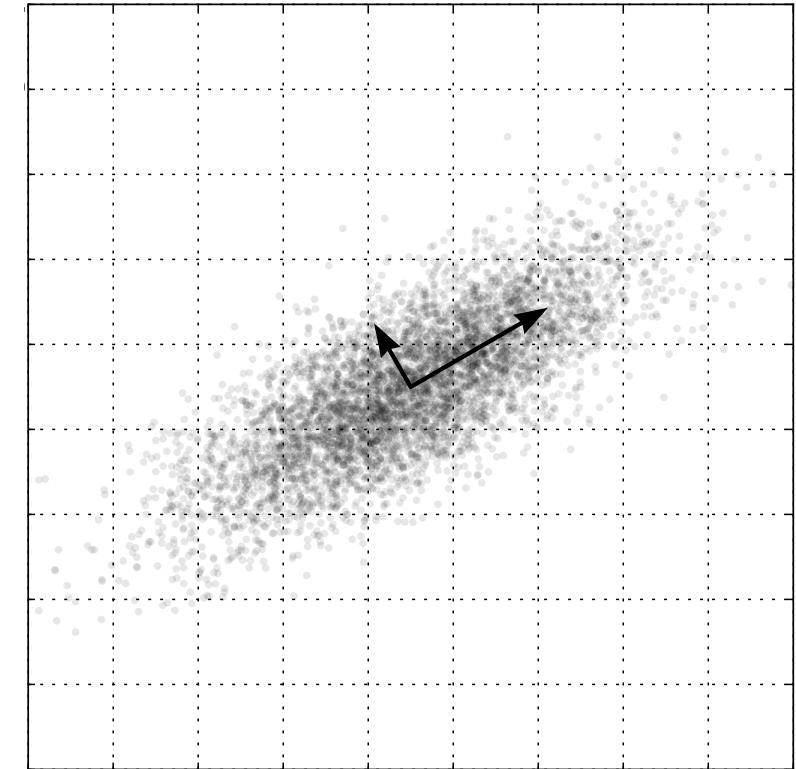


Covariance
matrix

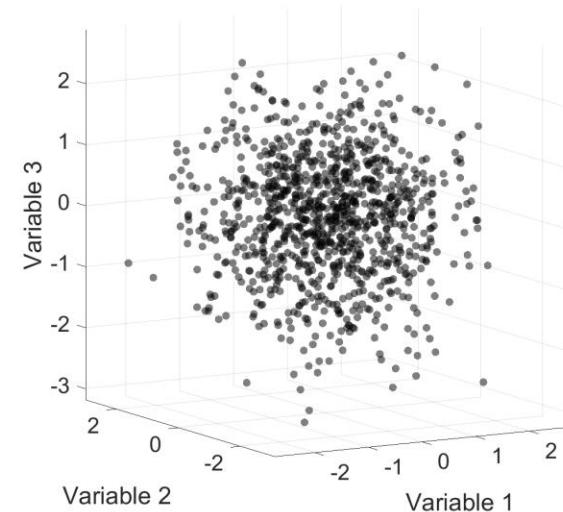
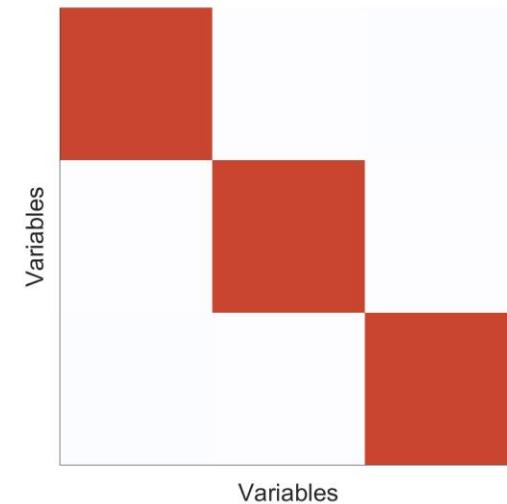
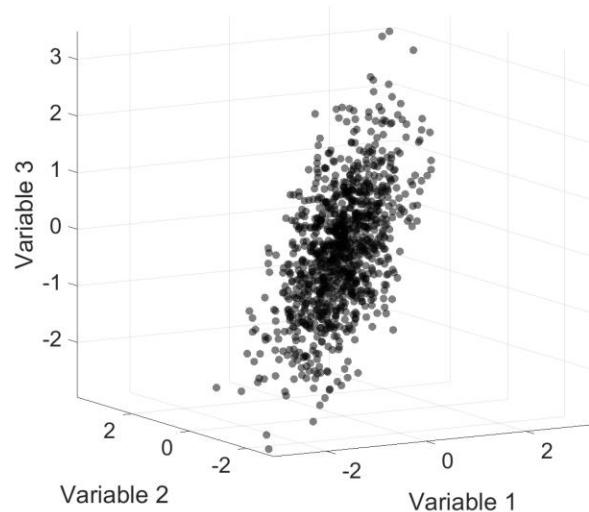
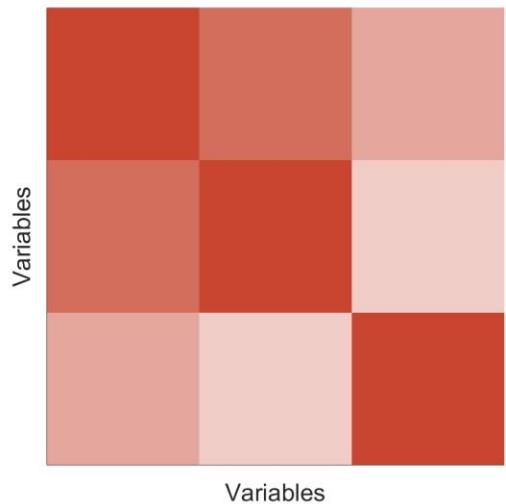


Principal component analysis (PCA)

Eigendecomposition of the
data **covariance** matrix

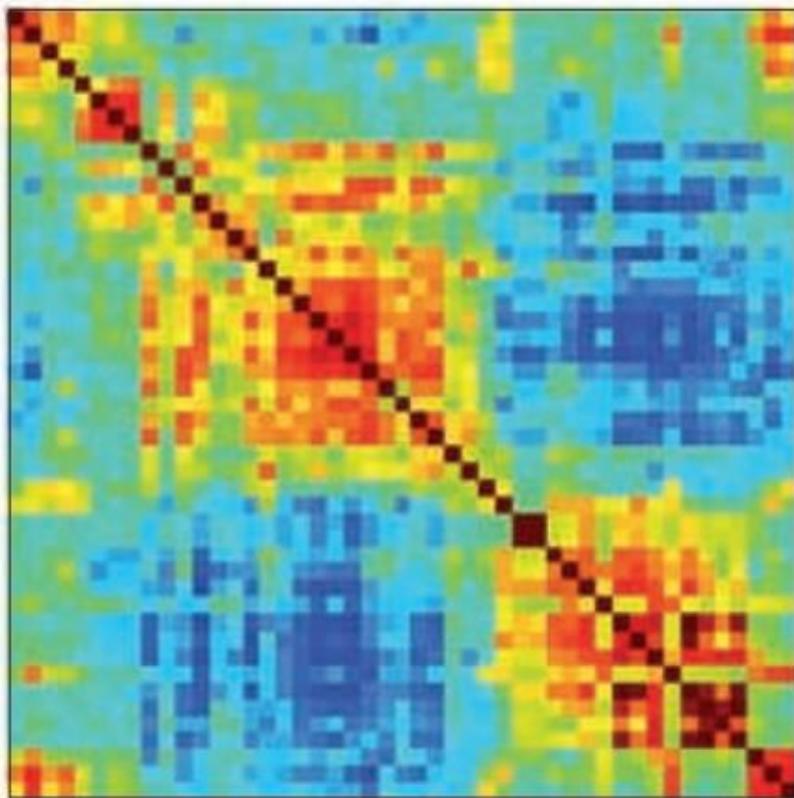


Correlation space



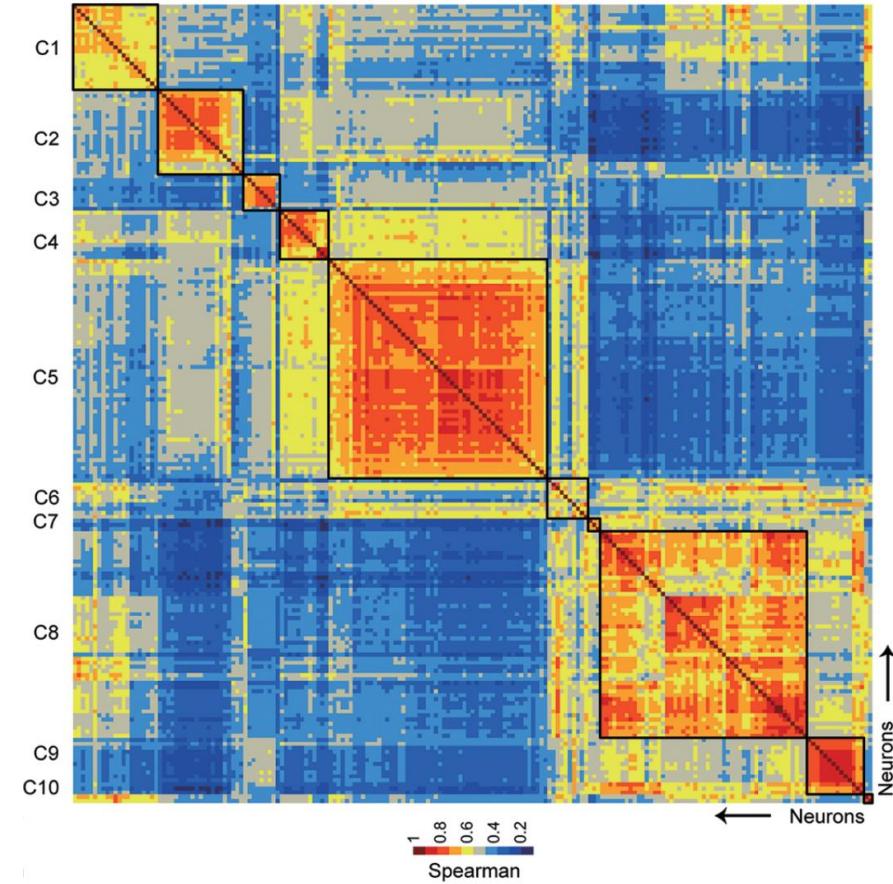
Examples of covariance/correlation matrices

Functional Connectivity in the Brain



Bullmore and Sporns, Nature Reviews 2009

Gene expression profile in neurons



Li et al., Cell Research 2016

Compute covariance/correlation matrix

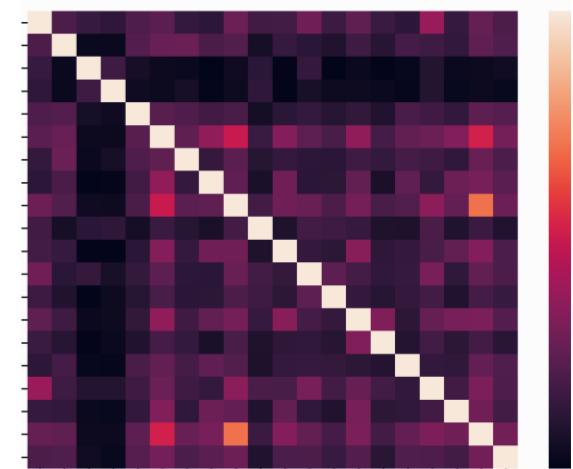
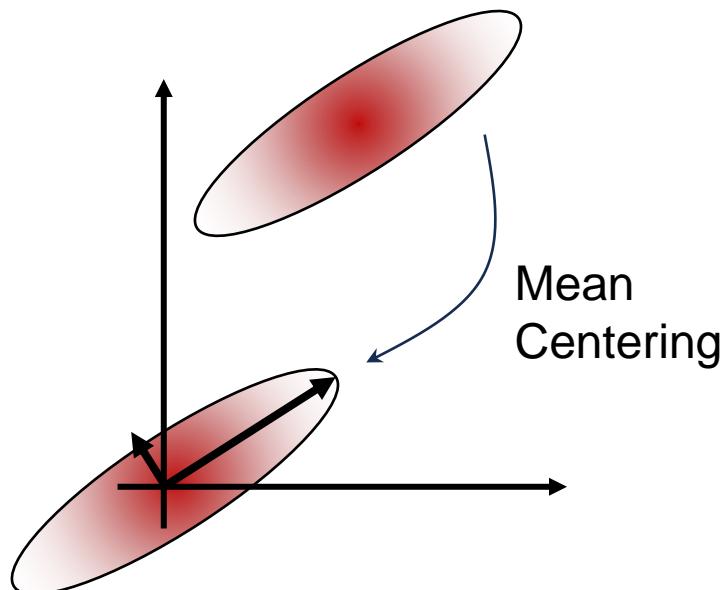
$$\bar{X} = \frac{1}{n} \sum_{j=1}^n X_j$$

$$B = X - \bar{X}$$

$B = \text{zscore}(x)$ [for correlation]

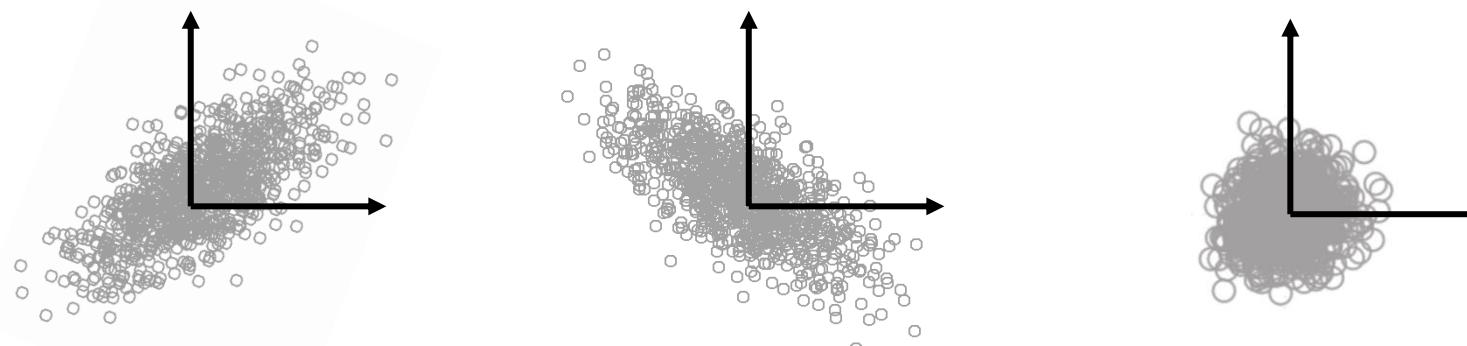
$$C = B^T B$$

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_3 & y_3 \end{bmatrix}$$



Dot product and covariance

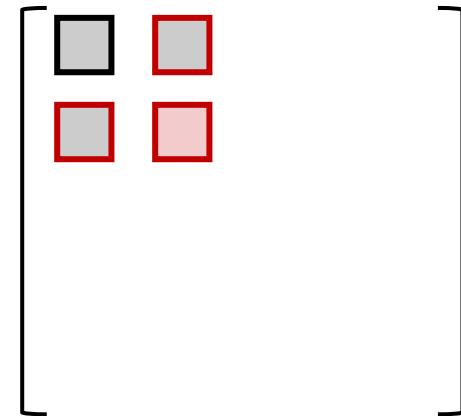
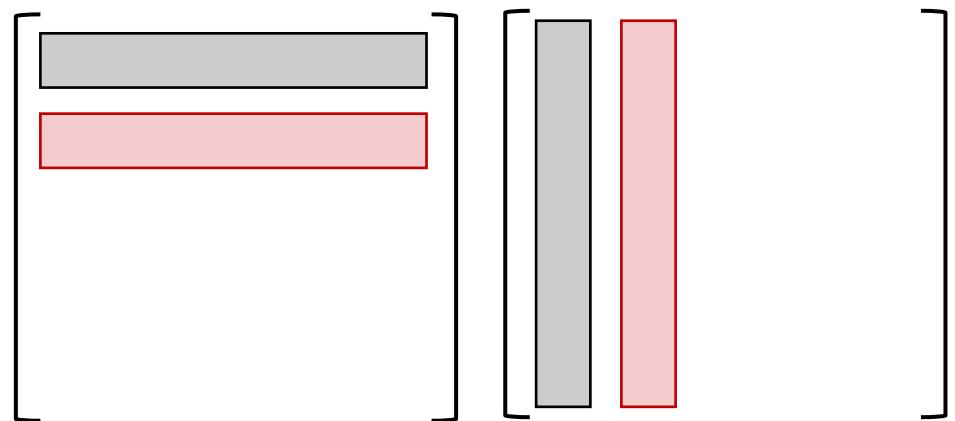
$$Cov_{x,y} = \frac{\sum (x_j - \bar{x})(y_j - \bar{y})}{N - 1} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}$$



Covariance Matrix

$$Cov_{x,y} = \frac{\sum (x_j - \bar{x})(y_j - \bar{y})}{N - 1}$$

$$B = X - \bar{X}$$



$$C = B^T B$$

Dot product and correlation

$$Corr_{x,y} = \frac{\sum(x_j - \bar{x})(y_j - \bar{y})}{\sqrt{\sum(x_j - \bar{x})^2 \sum(y_j - \bar{y})^2}}$$

$$= \frac{\sum(x_j - \bar{x})(y_j - \bar{y})}{||x - \bar{x}|| ||y - \bar{y}||} \quad ||x|| = \sqrt{\sum_{i=1}^n x_i^2}$$

$$= \cosine(x - \bar{x}, y - \bar{y})$$

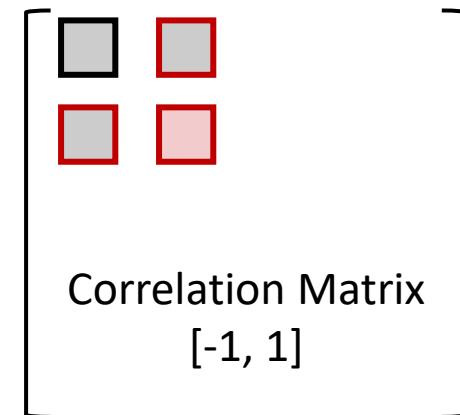
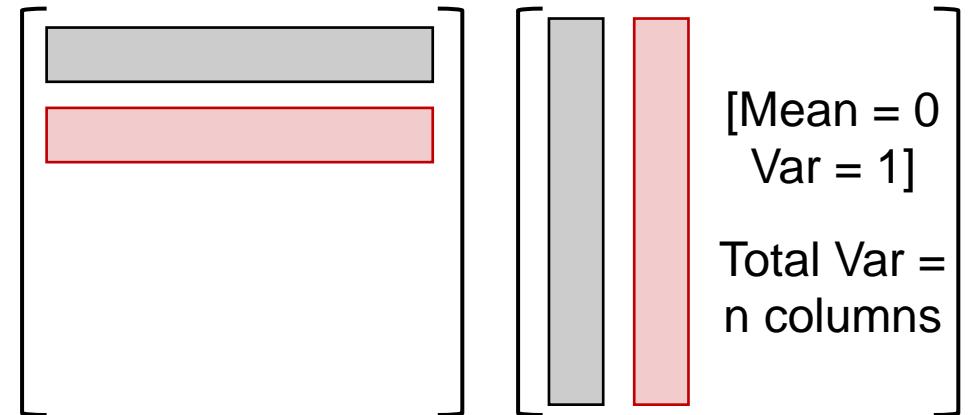
Correlation Matrix

$$Corr_{x,y} = \frac{\sum(x_j - \bar{x})(y_j - \bar{y})}{\sqrt{\sum(x_j - \bar{x})^2 \sum(y_j - \bar{y})^2}}$$

$$\bar{X} = \frac{1}{n} \sum_{j=1}^n X_j$$

$$B = zscore(X) = \frac{x - \bar{x}}{\sigma}$$

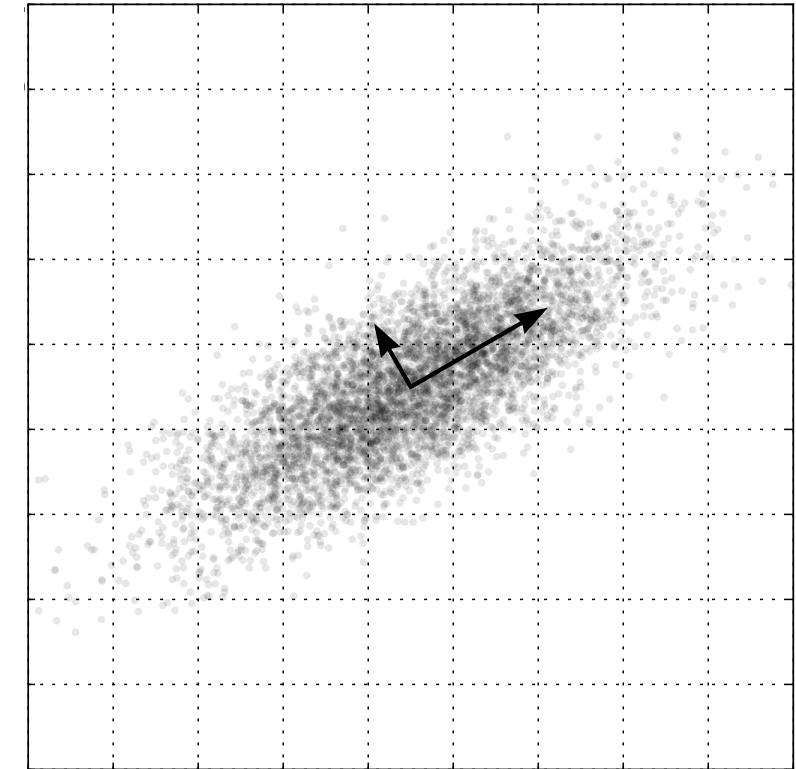
$$C = B^T B$$



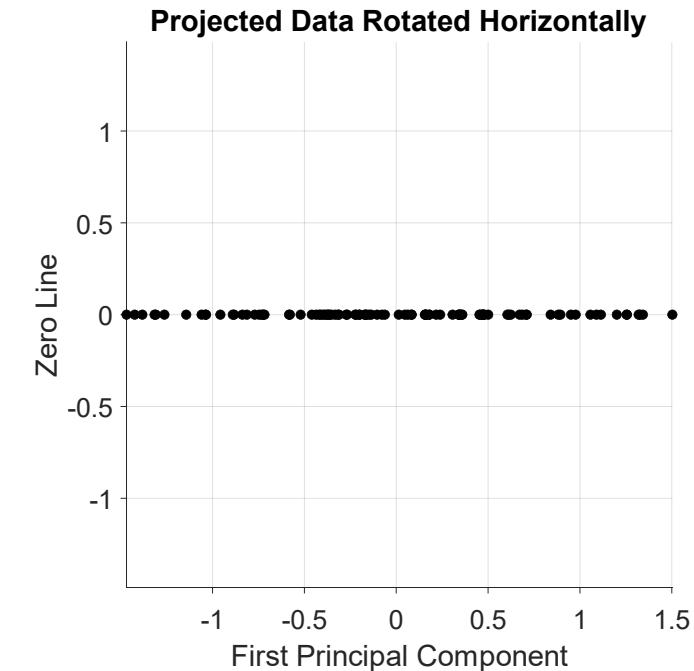
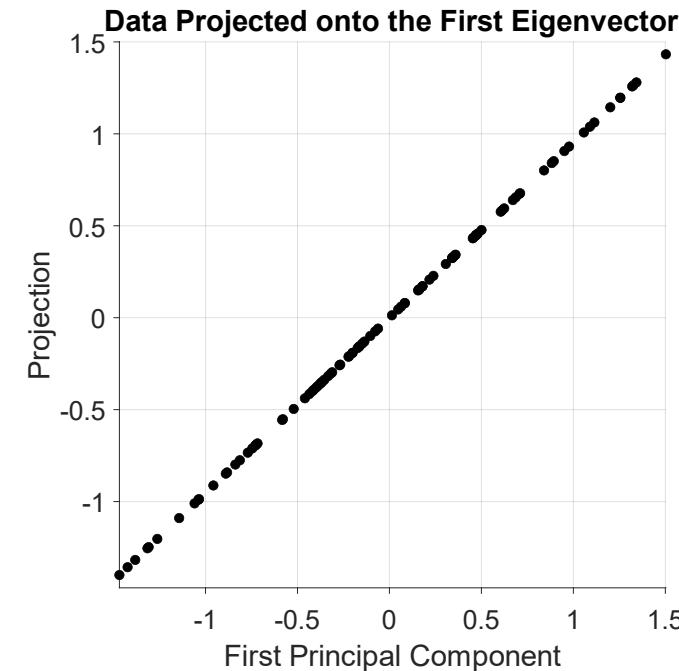
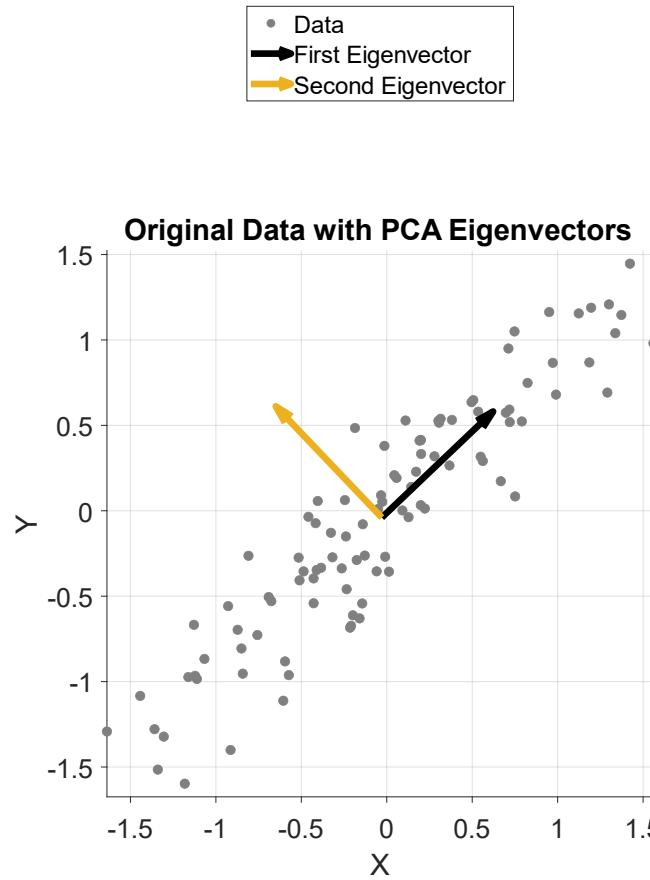
$$C = B^T B$$

Principal component analysis (PCA)

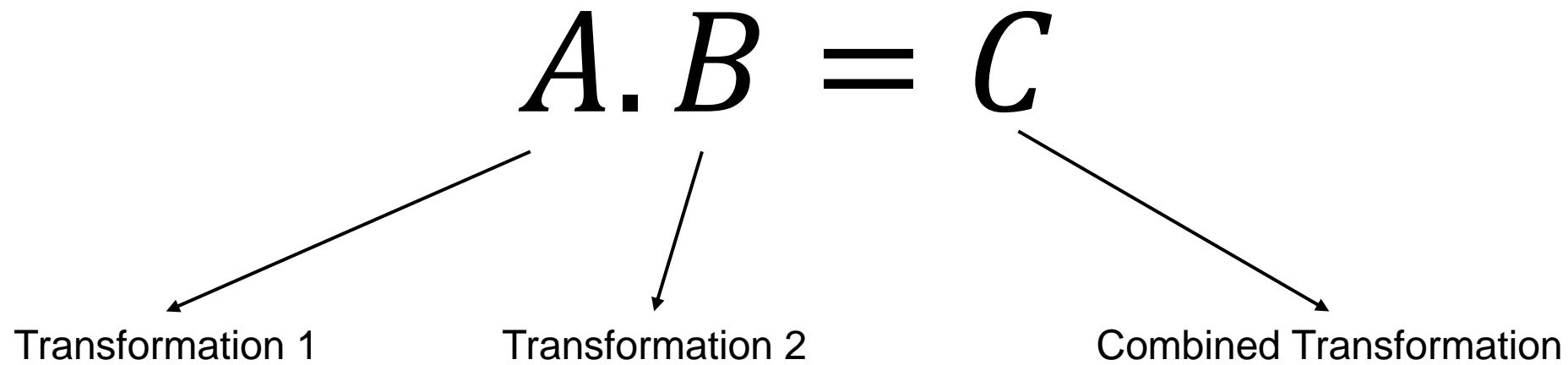
Eigendecomposition of the
data **covariance** matrix



Principal component analysis (PCA)



Composition of linear transformations



Matrix multiplication: Composition of Sequential Transformations

Matrix decomposition

$$[X] = [\quad] [\quad] [\quad]$$

Combined Transformation
(square matrix)

Simple Transformation

Eigendecomposition

$$A = V\Lambda V^T$$

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ | & | & | \\ V_1 & V_2 & V_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_3 \end{bmatrix} \begin{bmatrix} & & \\ | & | & | \\ V_1 & V_2 & V_3 \\ | & | & | \end{bmatrix}^T$$

Eigenvectors
(Orthogonal)

Eigenvalues
(Diagonal)

Eigendecomposition

$$A\boldsymbol{v} = \lambda\boldsymbol{v}$$

$$A\boldsymbol{v} = \lambda\boldsymbol{v} = \lambda I\boldsymbol{v}$$

$$(A - \lambda I)\boldsymbol{v} = 0$$

$$\det(A - \lambda I) = 0$$

For more details check **Session 2 - Differential equations and dynamical systems**
on the Youtube of Neurosyntax academy:
<https://www.youtube.com/@neurosyntaxacademy>

Eigenvectors of a symmetric matrix

Let A be a square matrix. Suppose that A has an orthonormal eigenbasis $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n$, with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$. What can we say about A ?

Let Q be the matrix with columns $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n$. Let D be the diagonal matrix with diagonal $\lambda_1, \lambda_2, \dots, \lambda_n$. Then we have

$$A = QDQ^{-1}.$$

But, since $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n$ is orthonormal, we know that Q is orthogonal, and thus $Q^{-1} = Q^T$. So we have

$$A = QDQ^T.$$

We deduce that

$$A^T = (QDQ^T)^T = (Q^T)^T D^T Q^T = QDQ^T = A.$$

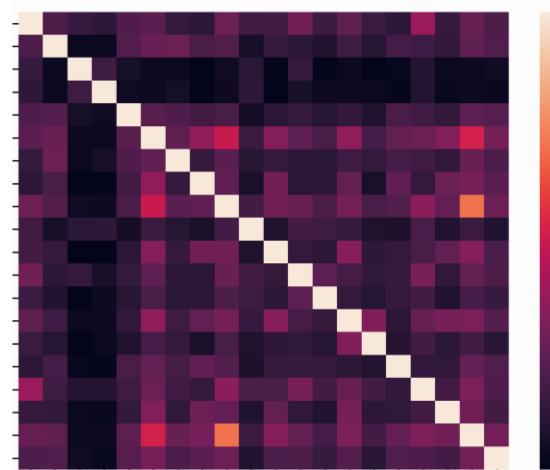
In other words, A is symmetric!

We have thus shown: Let A be a square matrix. If A has an orthonormal eigenbasis $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n$ then A is symmetric, meaning that $A = A^T$.

The big result about symmetric matrices is that the reverse is true:

The spectral theorem: If A is a symmetric $n \times n$ matrix, then A has an orthonormal eigenbasis.

The easy part of this is that, if A has an eigenbasis, then it has an orthonormal eigenbasis.



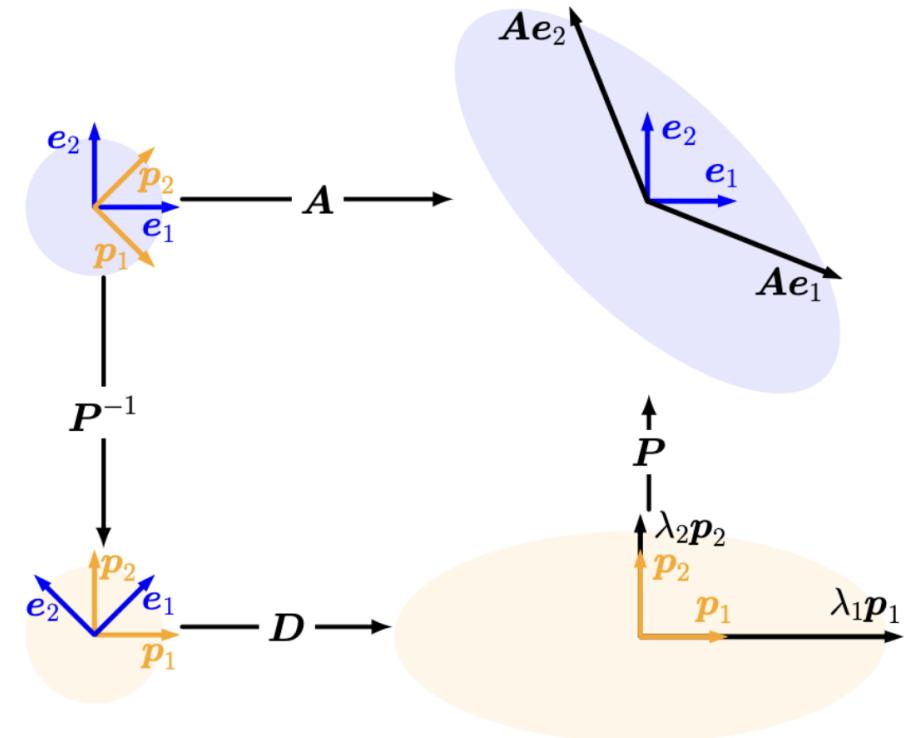
Geometric intuitions of Eigendecomposition

$$A = PDP^{-1}$$

Top-left to bottom-left: P^{-1} performs a basis change.

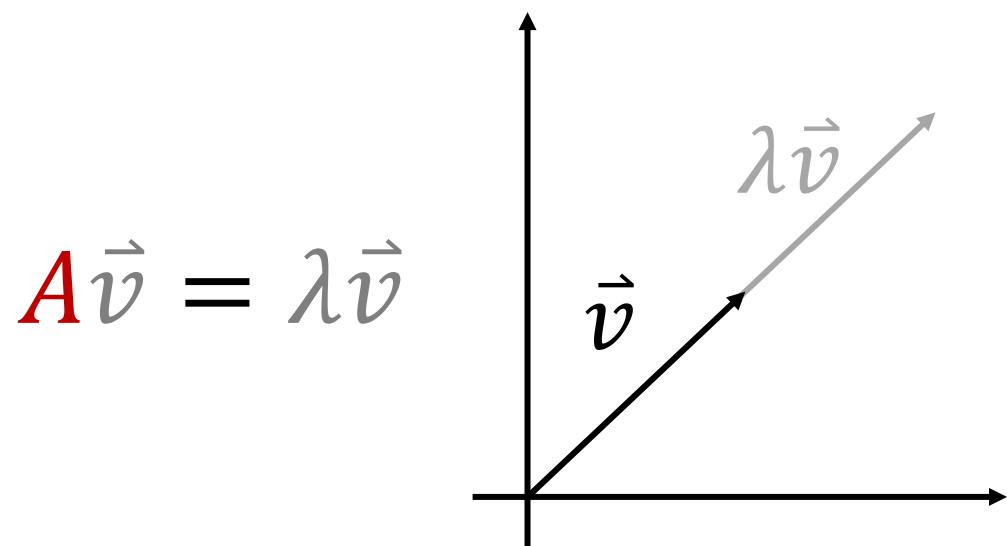
Bottom-left to bottom-right: D performs a scaling.

Bottom-right to top-right: P undoes the basis change.



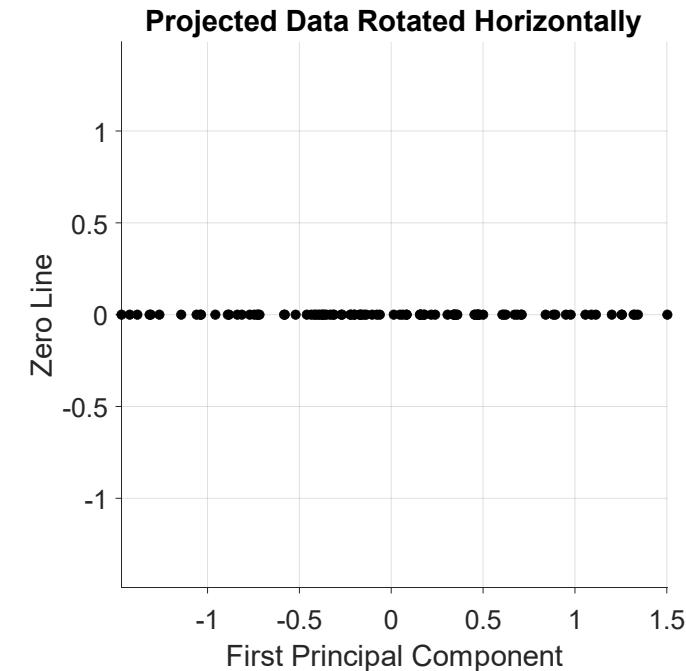
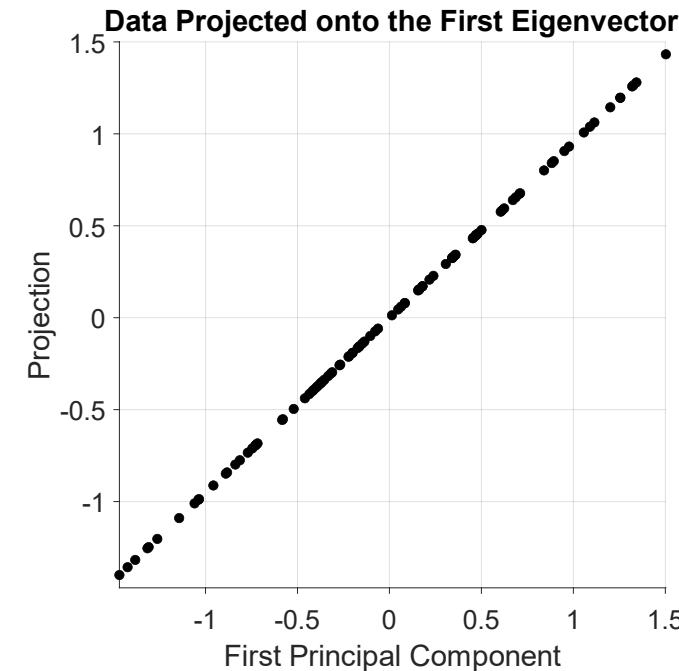
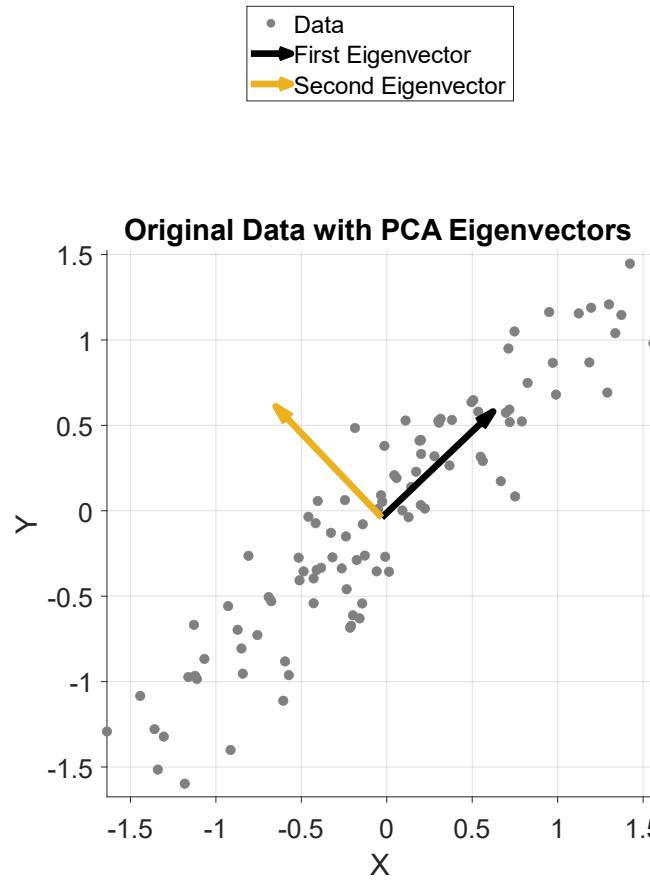
Eigendecomposition

$$\begin{bmatrix} \text{Red Grid} \end{bmatrix} = \begin{bmatrix} V_1 & V_2 & V_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_3 \end{bmatrix}$$



Each eigenvalue in Principal Component Analysis (PCA) indeed tells us how much of the total data variance is explained after projecting the data onto its corresponding eigenvector.

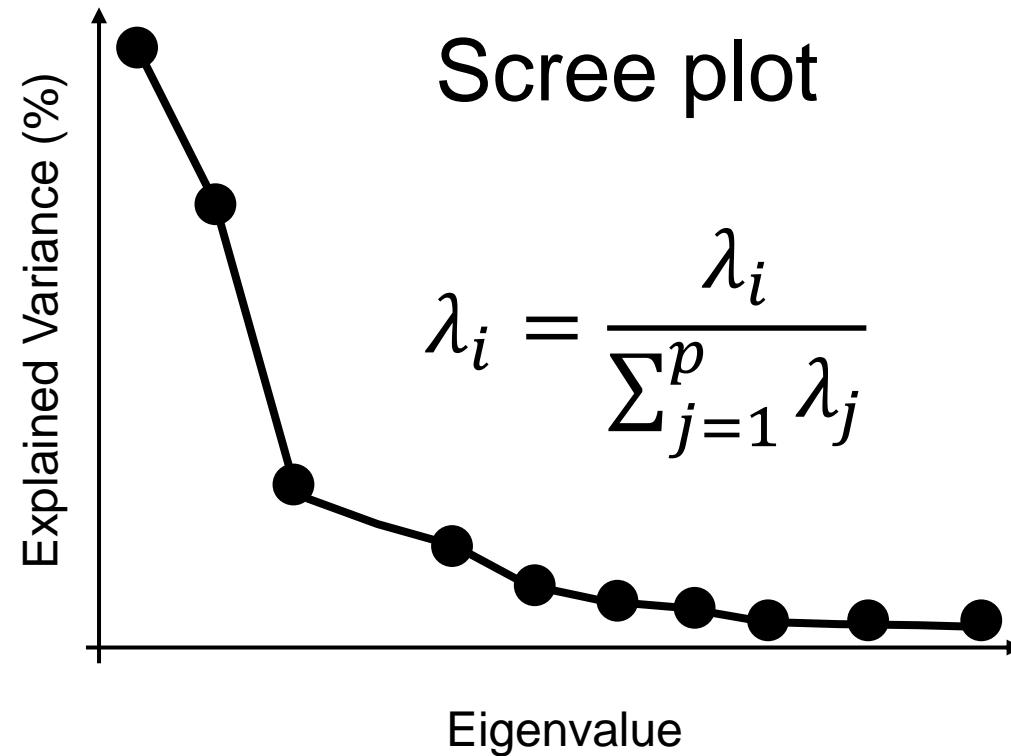
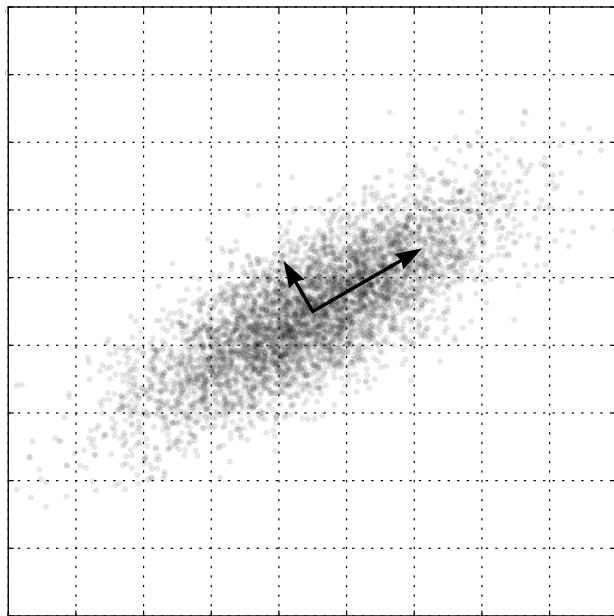
Principal component analysis (PCA)



Explained Variance

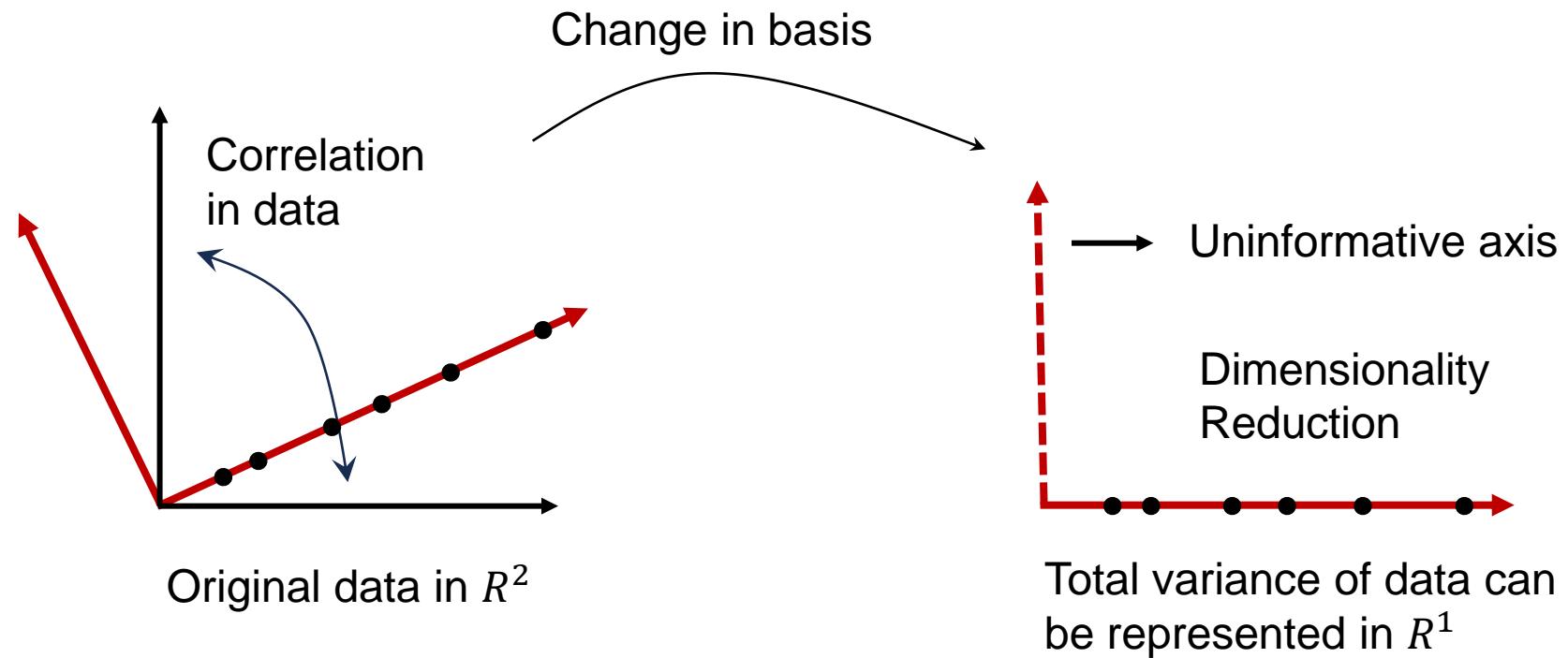
$$\begin{bmatrix} \text{Matrix} \end{bmatrix} = \begin{bmatrix} V_1 & V_2 & V_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} V_1 & V_2 & V_3 \end{bmatrix}^T$$

$$EV = 1 - \frac{Var[Y - \hat{Y}]}{Var[Y]}$$

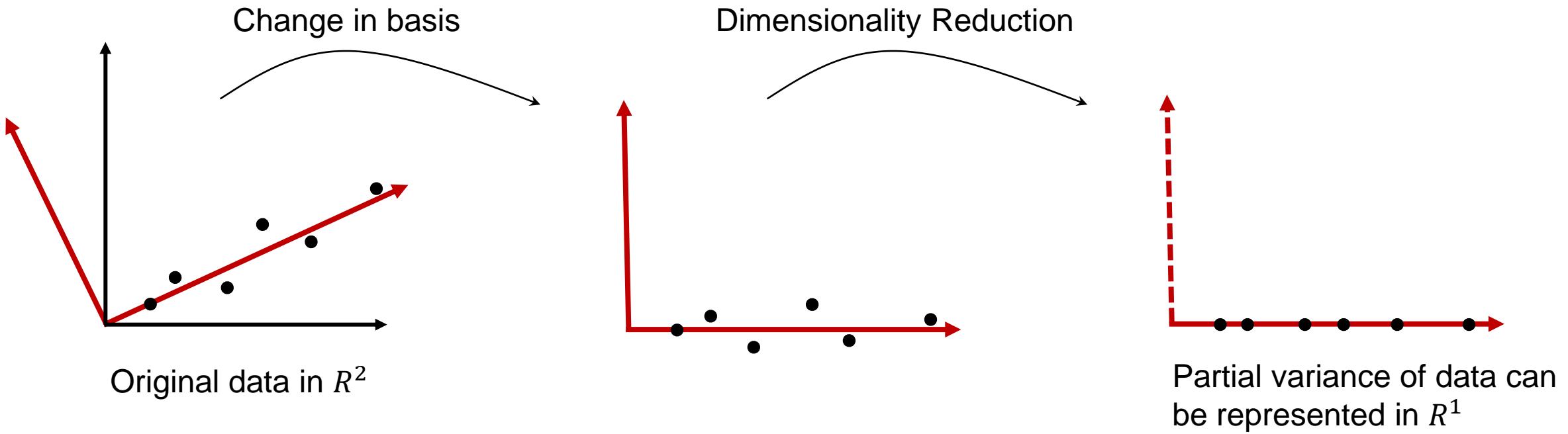


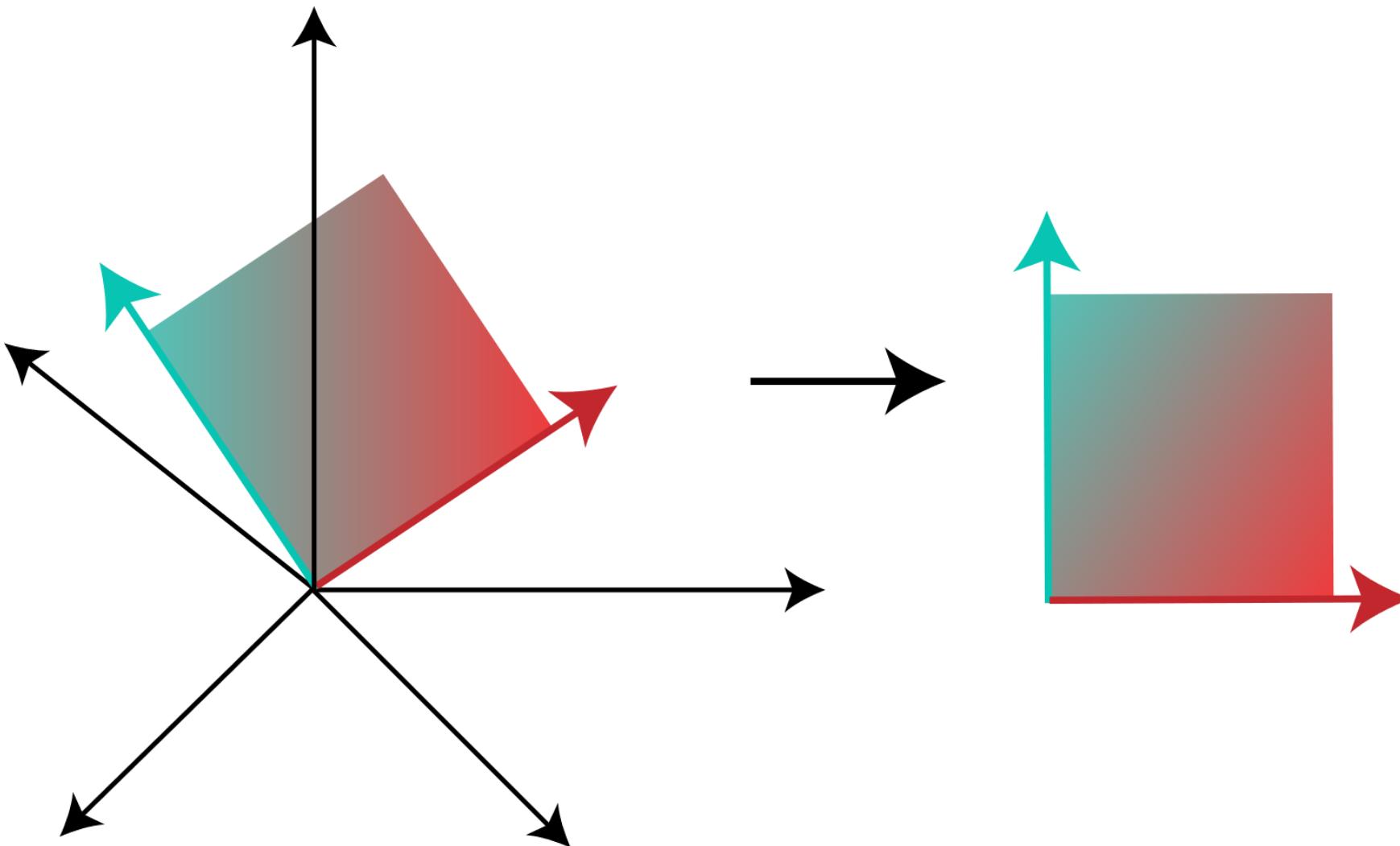
Dimensionality Reduction

Component analysis, dimensionality reduction and change in basis vectors



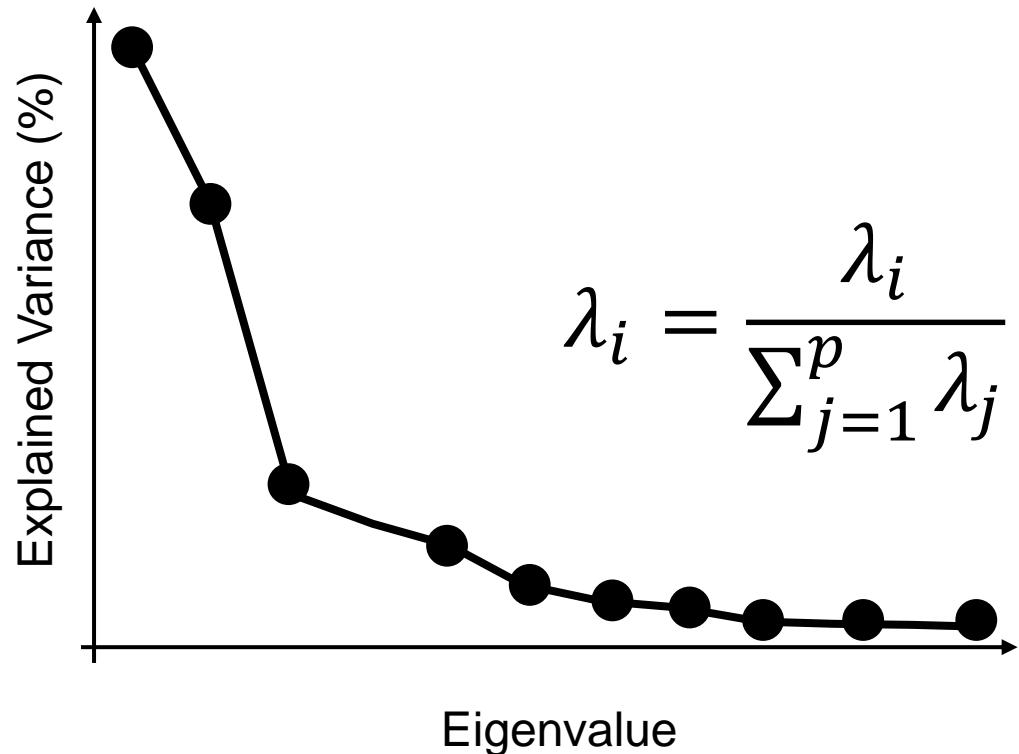
Component analysis, dimensionality reduction and change in basis vectors





Dimensionality Reduction

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ V_1 & V_2 & V_3 \\ & & \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \\ & \lambda_3 \end{bmatrix} \begin{bmatrix} & & \\ V_1 & V_2 & V_3 \\ & & \end{bmatrix}^T$$

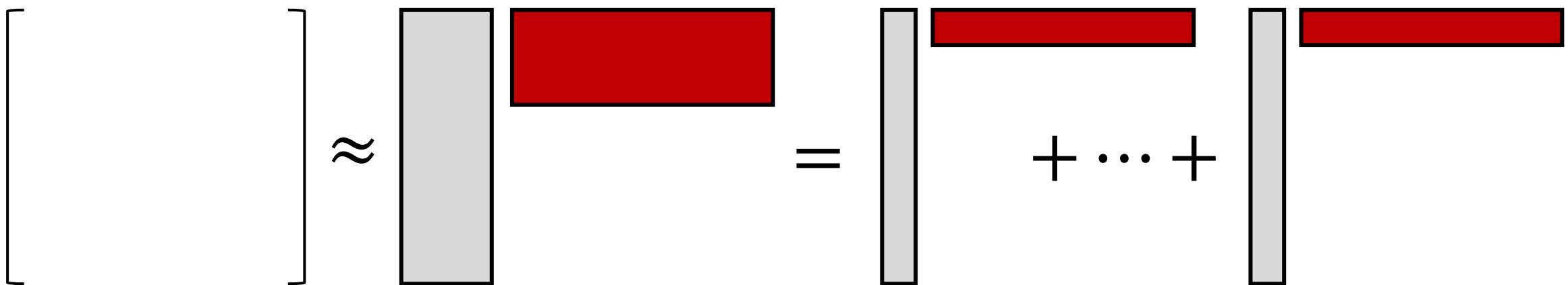


$$X_{n \times m} V_{m \times k} = R_{n \times k}$$
$$k < m$$

Matrix Decomposition and Dimensionality Reduction

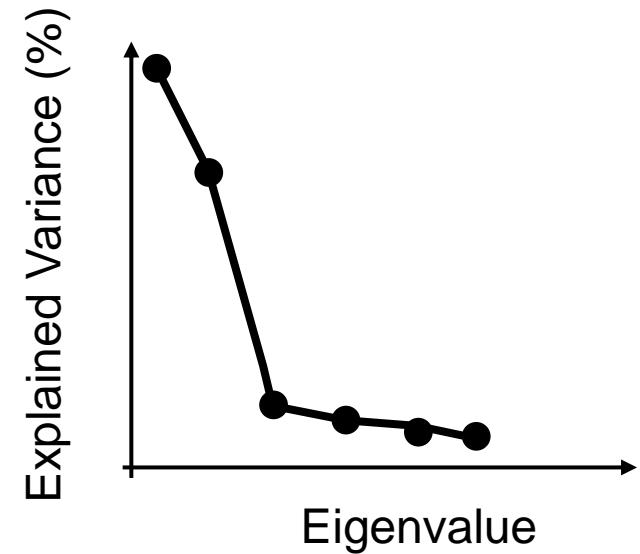
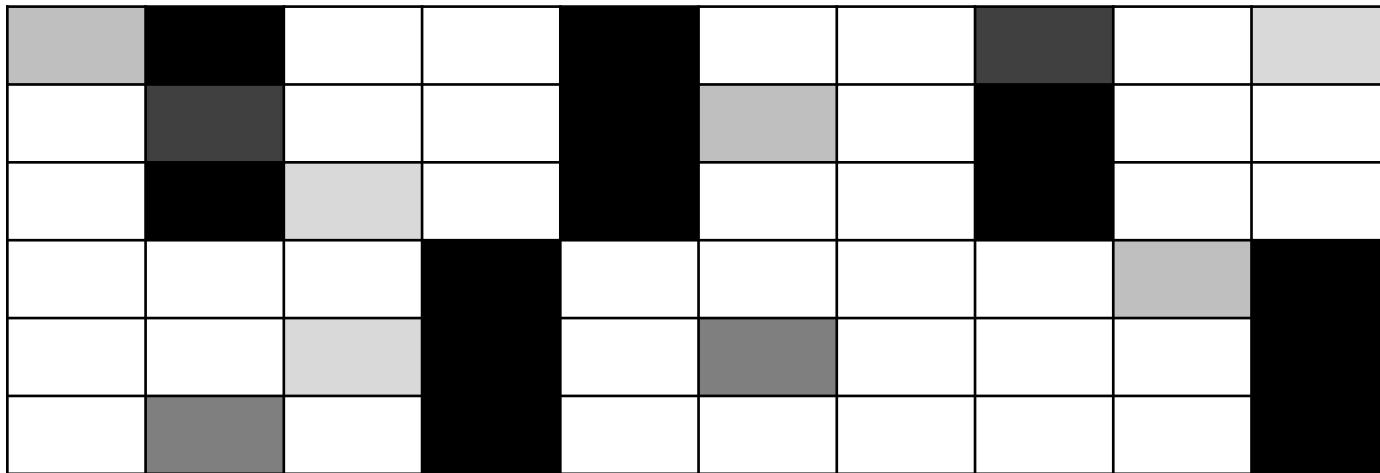
$$x_{ij} \approx \sum_{r=1}^R \lambda_i^r v_j^r$$

$$A \quad \Lambda \quad V^T$$



Dimensionality Reduction

$$D_{6 \times t} \longrightarrow D_{t \times 6}^T$$

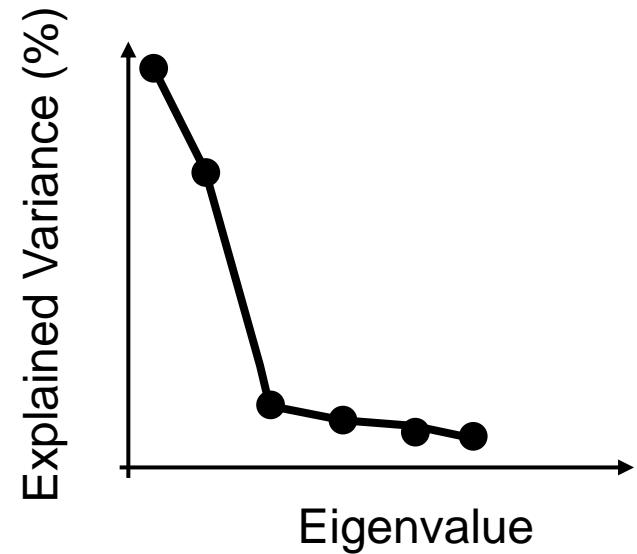
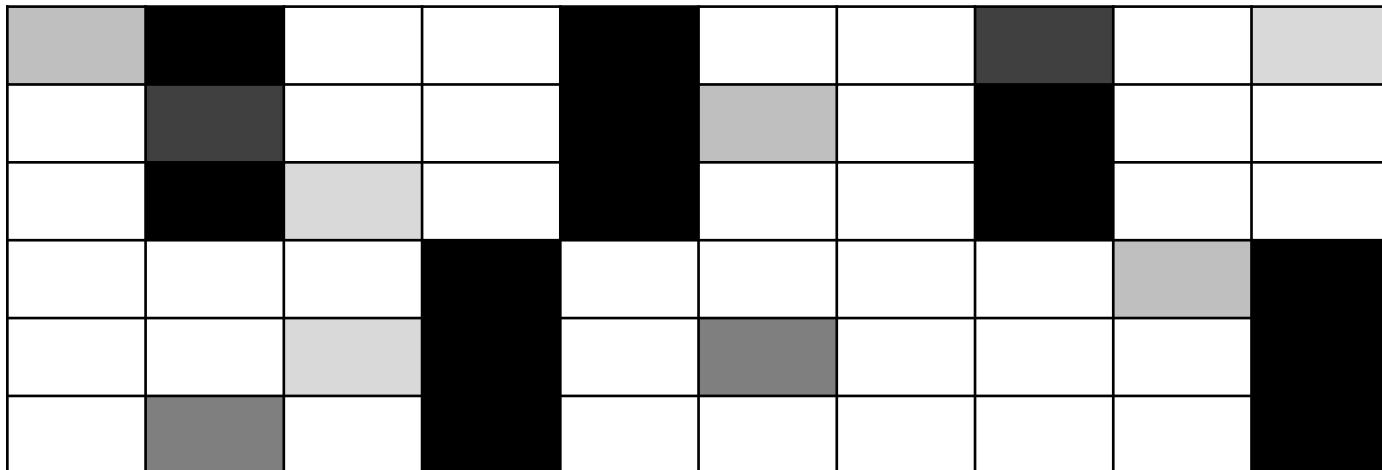


$$C_{n \times n} = V_{n \times n} \Lambda_{n \times n} V_{n \times n}^T$$

$$\widehat{D}_{t \times r} = D_{t \times n}^T V_{n \times r}; r < n$$

Dimensionality Reduction

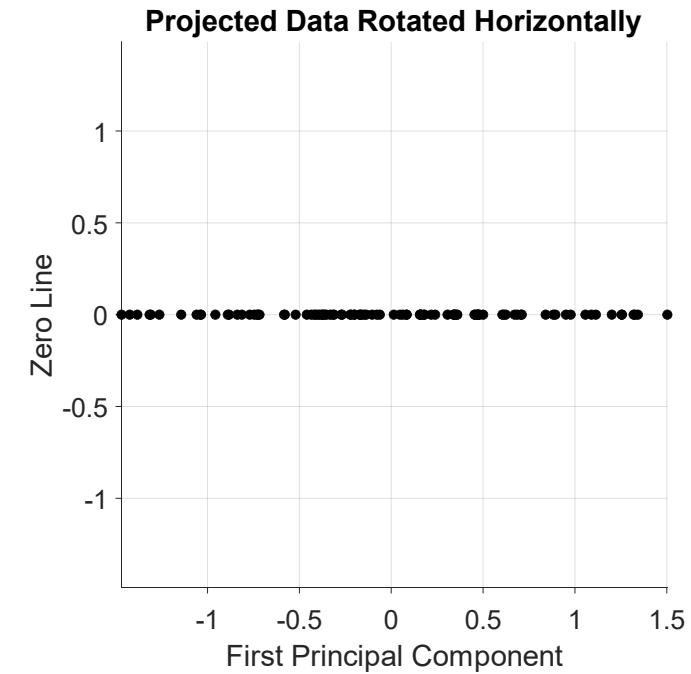
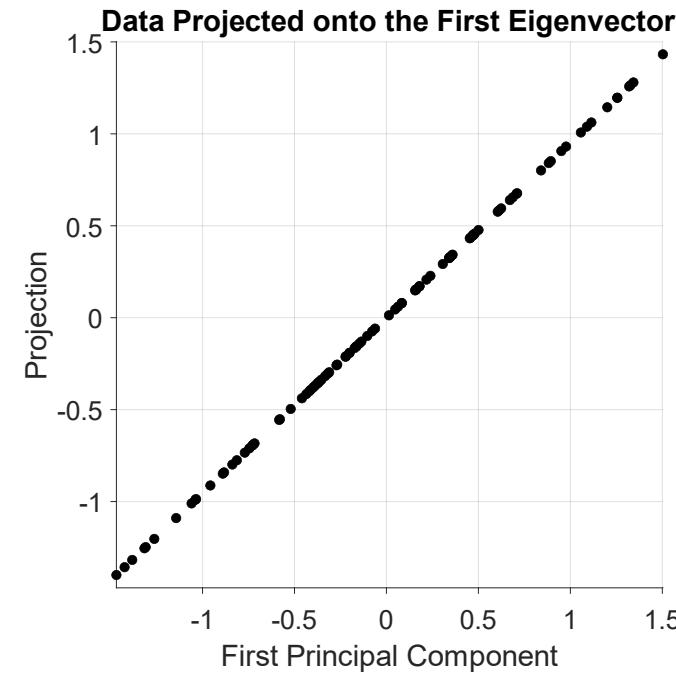
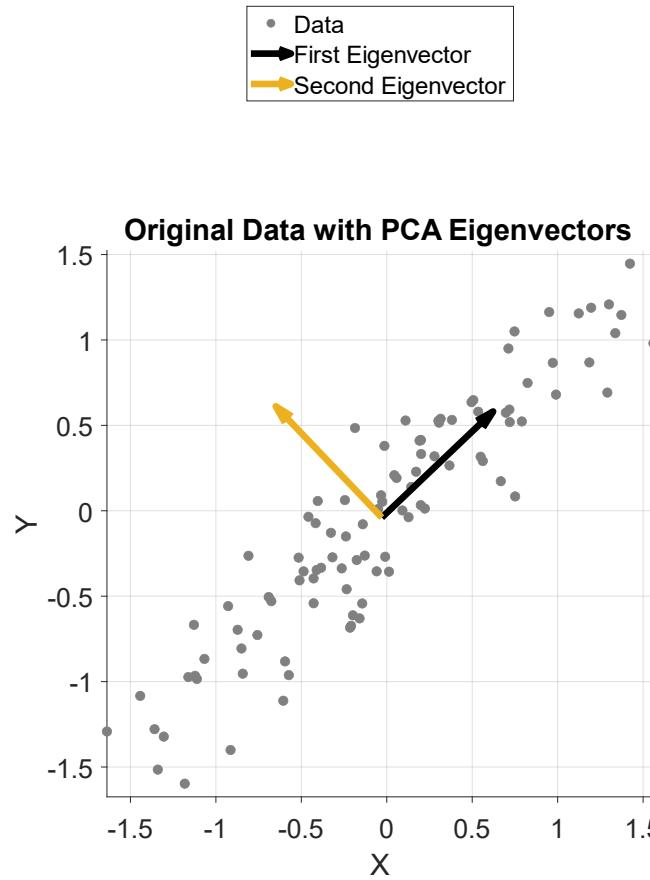
$$D_{6 \times t} \longrightarrow D_{t \times 6}^T$$



$$C_{6 \times 6} = V_{6 \times 6} \Lambda_{6 \times 6} V_{6 \times 6}^T$$

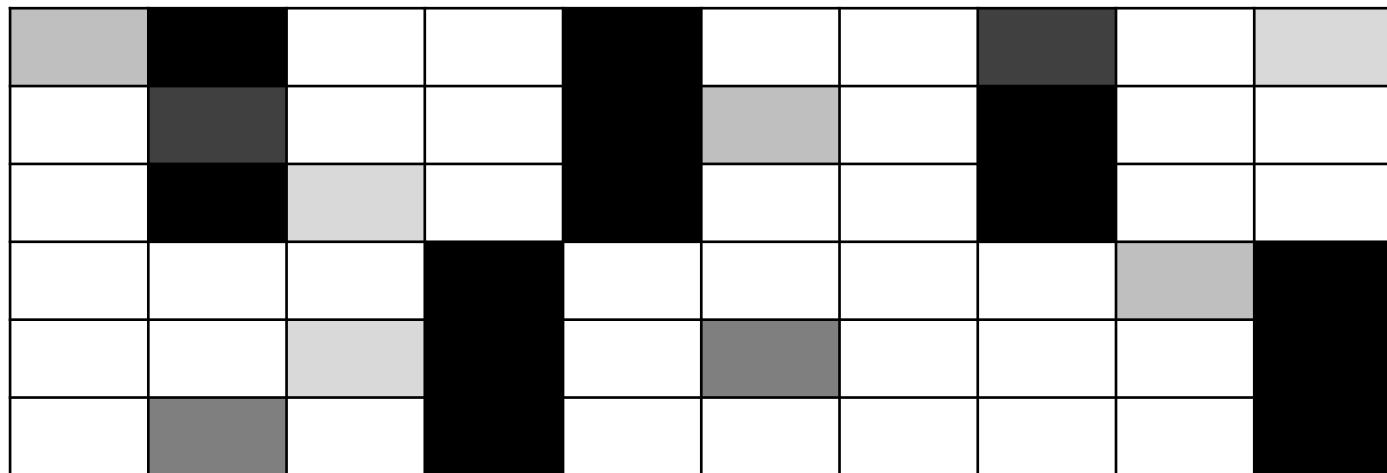
$$\widehat{D}_{t \times 2} = D_{t \times 6}^T V_{6 \times 2}$$

Principal component analysis (PCA)

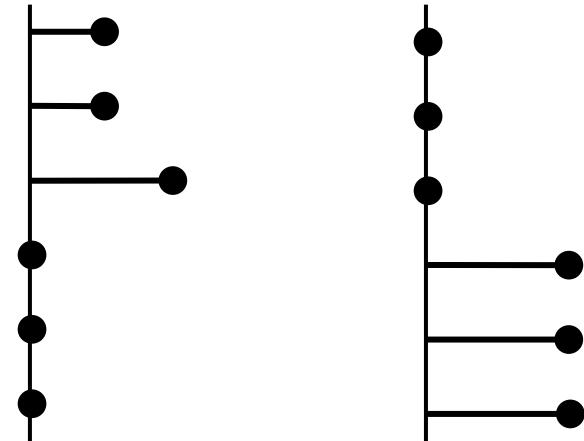


Dimensionality Reduction

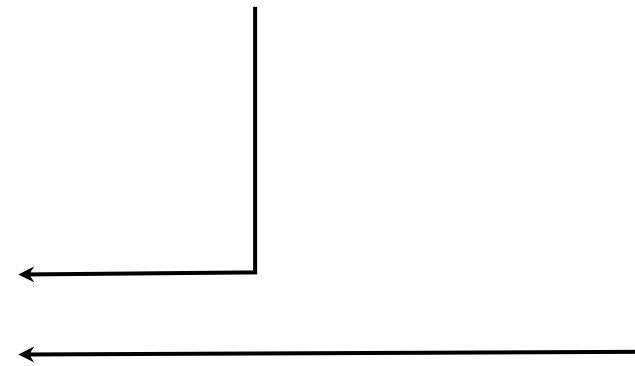
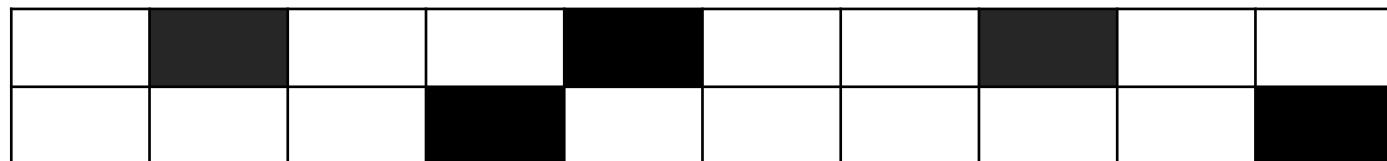
$$D_{6 \times t} \longrightarrow D_{t \times 6}^T$$



Eigenvectors



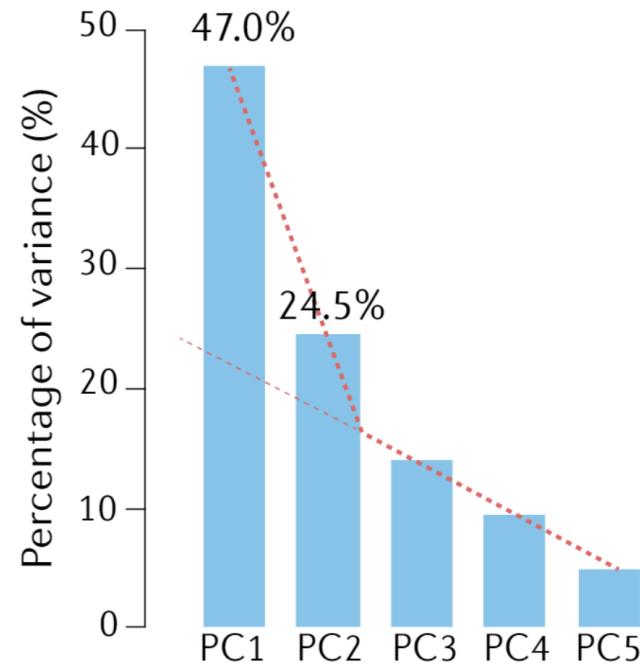
$$\widehat{D}_{t \times 2} = D_{t \times 6}^T V_{6 \times 2}$$



Truncation

In Component Analysis, truncation refers to the process of reducing the dimensionality of the data by selecting only a subset of principal components. This is typically done by retaining the principal components that capture the most variance and discarding those that contribute the least.

Elbow Rule



Marchenko–Pastur distribution

We will now turn our attention to rectangular matrices. Let

$$X = (\underline{X}_1, \underline{X}_2, \dots, \underline{X}_n) \in \mathbb{R}^{p \times n}$$

where X_{ij} are iid, $E(X_{ij}) = 0$, $E(X_{ij}^2) = 1$ and $p = p(n)$.

Define

$$S_n = \frac{1}{n} X X^T \in \mathbb{R}^{p \times p}$$

and let

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_p$$

denote the eigenvalues of the matrix S_n .

Define the random spectral measure by

$$\mu_n = \frac{1}{p} \sum_{i=1}^p \delta_{\lambda_i}$$

We are now ready to state the Marchenko-Pastur law

Theorem 1. Let S_n, μ_n be as above. Assume that $p/n \xrightarrow{n \rightarrow \infty} y \in (0, 1]$. Then we have

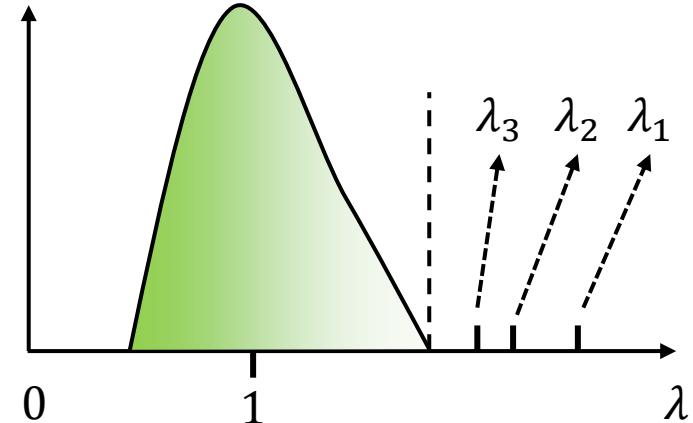
$$\mu_n(\cdot, \omega) \Rightarrow \mu \text{ a.s}$$

where μ is a deterministic measure whose density is given by

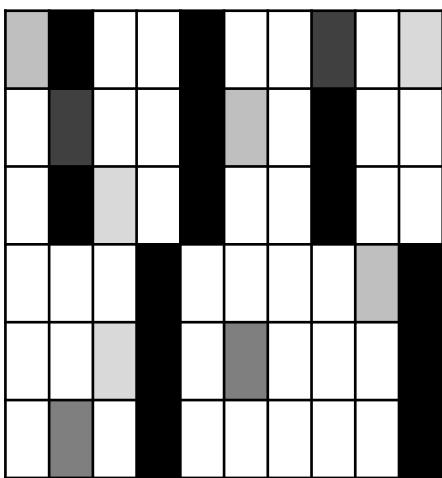
$$\frac{d\mu}{dx} = \frac{1}{2\pi xy} \sqrt{(b-x)(x-a)} \mathbf{1}_{(a \leq x \leq b)} \quad (1)$$

Here a and b are functions of y given by

$$a(y) = (1 - \sqrt{y})^2, \quad b(y) = (1 + \sqrt{y})^2$$

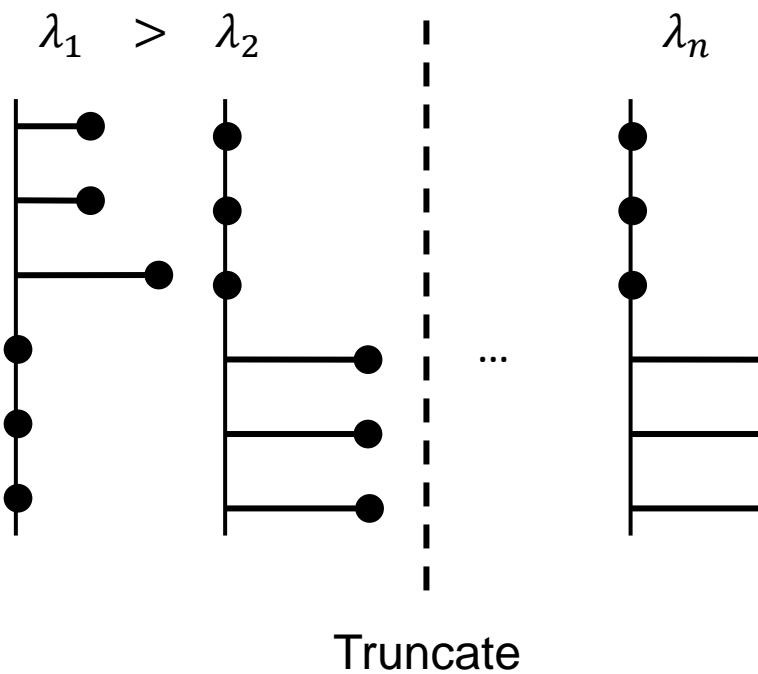


Data Matrix (S^*N)



PCA

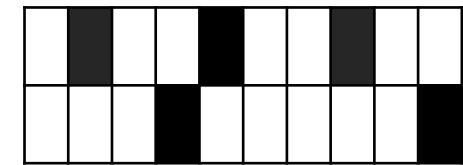
Eigenvectors/Factors/Loadings



Latent Variables (S^R)

$R < N$

Projection



PCA: Uniqueness

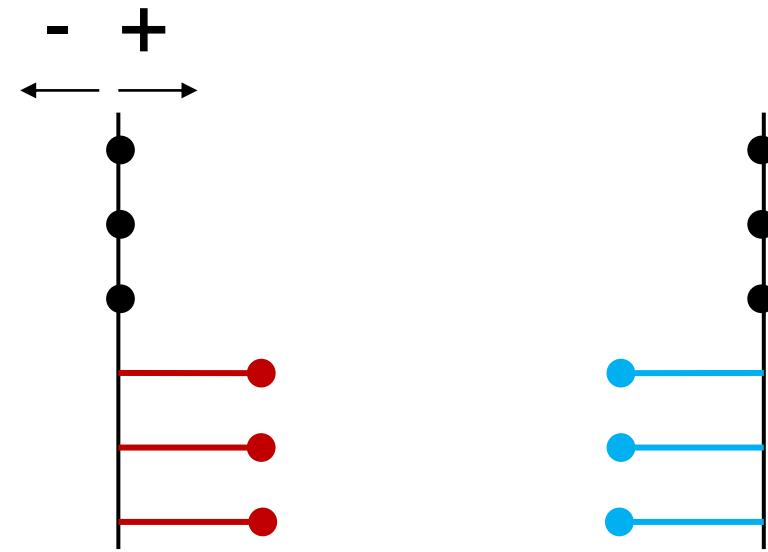
In PCA, the eigenvectors (principal components), the uniqueness of these components is up to a sign, which means:

1. Eigenvectors:

- Each eigenvector can be multiplied by -1 without changing the orientation of the principal component it represents. Both ν and $-\nu$ are valid eigenvectors corresponding to the same eigenvalue.

2. Eigenvalues:

- Eigenvalues are unique in their magnitude, but the associated eigenvectors' direction (positive or negative) can vary.



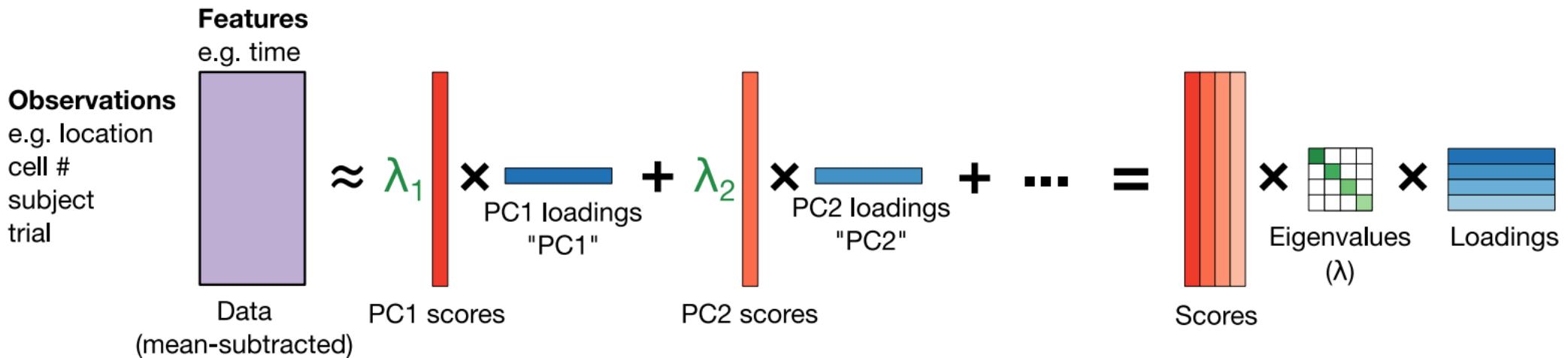
The highest absolute weight was positive

$$\text{if } \operatorname{sgn}(w_{|\max|}) > 0 \rightarrow V \times (1)$$

$$\text{if } \operatorname{sgn}(w_{|\max|}) < 0 \rightarrow V \times (-1)$$

Principal component analysis (PCA) summary

a



How to compute PCA

b

$$\text{Data transposed (mean-subtracted)} \times \text{Data (mean-subtracted)} = \text{Covariance matrix} = \text{Eigenvectors (loadings) transposed} \times \text{Eigenvalues} (\lambda) \times \text{Eigenvectors (loadings)}$$

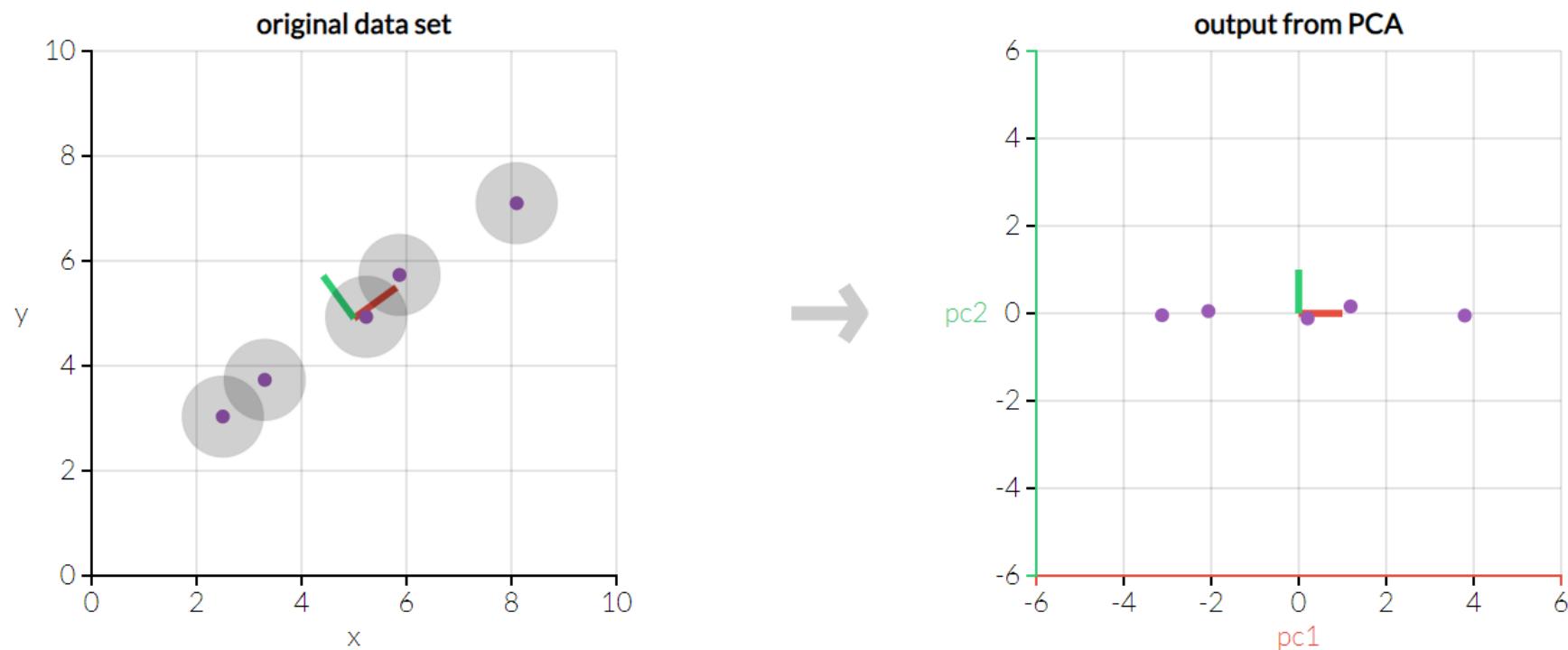
Labels include "Data transposed (mean-subtracted)" and "Data (mean-subtracted)" under the first two matrices, and "Covariance matrix" under the third. Labels for the eigenvectors and eigenvalues are placed between the second and third matrices.

c

$$\text{Data (mean-subtracted)} \times \text{Eigenvectors (loadings) transposed} = \text{Scores}$$

Labels include "Data (mean-subtracted)" under the first matrix, "Eigenvectors (loadings) transposed" under the second, and "Scores" under the third.

Interactive visualization



<https://setosa.io/ev/principal-component-analysis/>

PCA

Assumptions and limitations

Limitations of principal components in quantitative genetic association models for human studies

Yiqi Yao^{1†}, Alejandro Ochoa^{1,2*}

¹Department of Biostatistics and Bioinformatics, Duke University, Durham, United States; ²Duke Center for Statistical Genetics and Genomics, Duke University, Durham, United States

Abstract Principal Component Analysis (PCA) and the Linear Mixed-effects Model (LMM), sometimes in combination, are the most common genetic association models. Previous PCA-LMM comparisons give mixed results, unclear guidance, and have several limitations, including not varying the number of principal components (PCs), simulating simple population structures, and inconsistent use of real data and power evaluations. We evaluate PCA and LMM both varying number of PCs in realistic genotype and complex trait simulations including admixed families, subpopulation trees, and real multiethnic human datasets with simulated traits. We find that LMM without PCs usually performs best, with the largest effects in family simulations and real human datasets and traits without environment effects. Poor PCA performance on human datasets is driven by large numbers of distant relatives more than the smaller number of closer relatives. While PCA was known to fail on family data, we report strong effects of family relatedness in genetically diverse human datasets, not avoided by pruning close relatives. Environment effects driven by geography and ethnicity are better modeled with LMM including those labels instead of PCs. This work better characterizes the severe limitations of PCA compared to LMM in modeling the complex relatedness structures of multiethnic human data for association studies.

*For correspondence:
alejandro.ochoa@duke.edu

Present address: BenHealth



The screenshot shows the IEEE Xplore digital library interface. At the top, there are navigation links for "Browse", "My Settings", "Help", and "Institutional Sign In". Below the header is a search bar with a dropdown set to "All" and a magnifying glass icon. To the right of the search bar is an "ADVANCED SEARCH" link. The main content area displays a search result for a conference paper. The title is "Limitations of principal component analysis as a method to detect neuronal assemblies". Below the title, it says "Publisher: IEEE" and "Cite This". There is also a red "PDF" button. The authors listed are Camila Sardeto Deolindo, Ana Carolina Bione Kunicki, Fabricio Lima Brasil, Renan Cipriano Moioli, and All Authors. To the right of the author list are several small icons for sharing or saving the document. Below the title, there are two boxes: one for "1 Cites in Paper" and another for "205 Full Text Views".

Limitations of principal component analysis as a method to detect neuronal assemblies

Publisher: IEEE Cite This PDF

Camila Sardeto Deolindo ; Ana Carolina Bione Kunicki ; Fabricio Lima Brasil ; Renan Cipriano Moioli All Authors

1
Cites in
Paper

205
Full
Text Views



Abstract

Document Sections

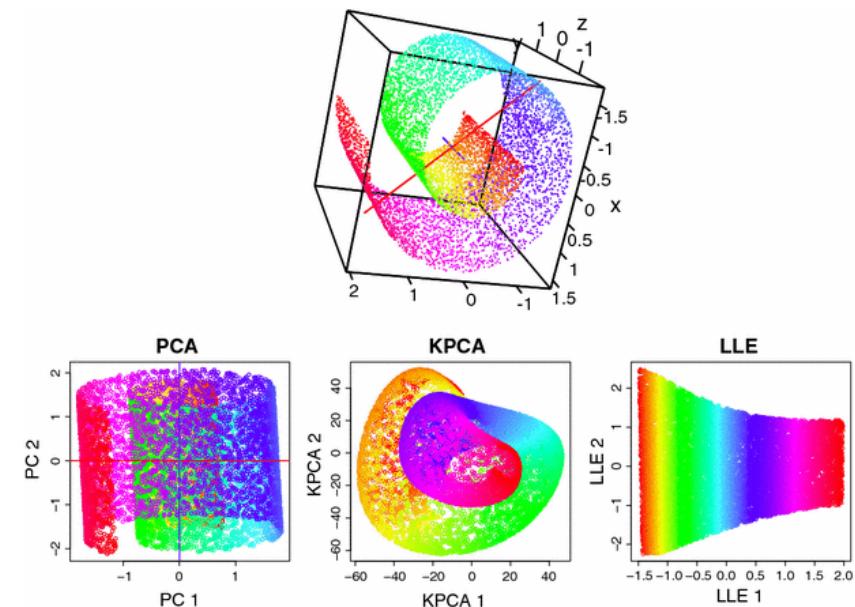
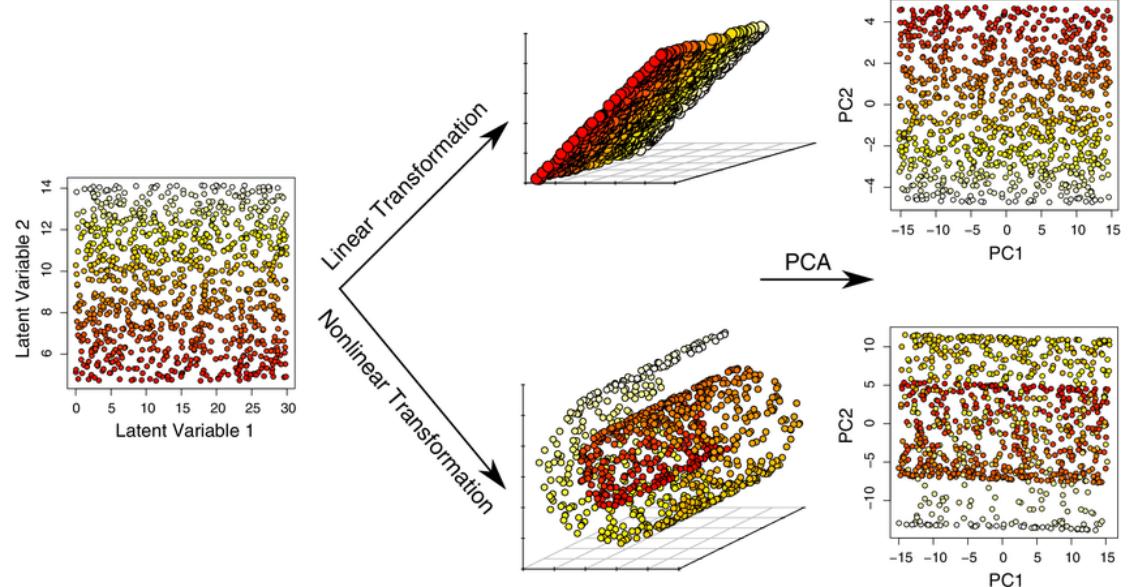
- I. Introduction
- II. Theoretical Background
- III. Methods
- IV. Results
- V. Discussion

Abstract:

The anatomical and functional characterization of neuronal assemblies (NA) is a major challenge in neuroscience. Principal component analysis (PCA) is a widely used method for feature detection, however, when dealing with neuronal data analysis, its limitations have not yet been fully understood. Our work complements previous PCA studies which, in general, characterize NAs based solely on excitatory neuronal interactions. We analysed the performance of PCA in two neglected scenarios: assemblies containing patterns of neural interactions (1) with inhibition and (2) with delays. The analyses considered two types of artificially generated data, one drawn from a traditional Poissonian model, and the other drawn from a latent multivariate Gaussian model; in both models, data from a behaving Wistar rat was used for parameter tuning. Our results highlight scenarios in which neglecting complex interactions between neurons can lead to false conclusions when using PCA to detect NAs. Also, we reinforce the importance of more realistic simulations in the evaluation of neuronal signal processing algorithms.

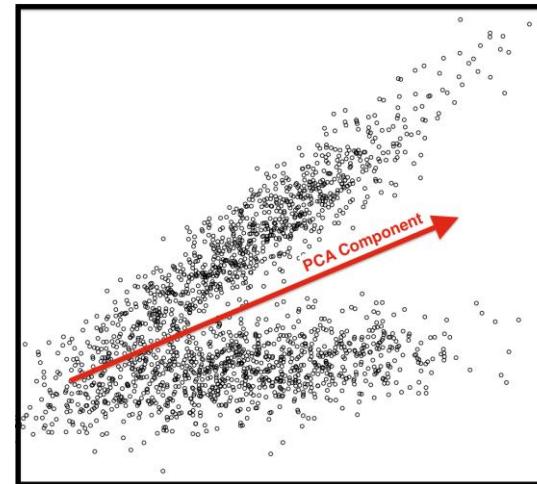
PCA Assumptions

- 1. Linearity:** PCA assumes that the relationships between variables are linear.
- 2. Orthogonality:** Principal components are orthogonal (uncorrelated) to each other.
- 3. Large Variance Indicates Importance:** It assumes that components with larger variance are more important.
- 4. Gaussian Distribution:** Assumes the data is Gaussian distributed, although this is not strictly necessary.

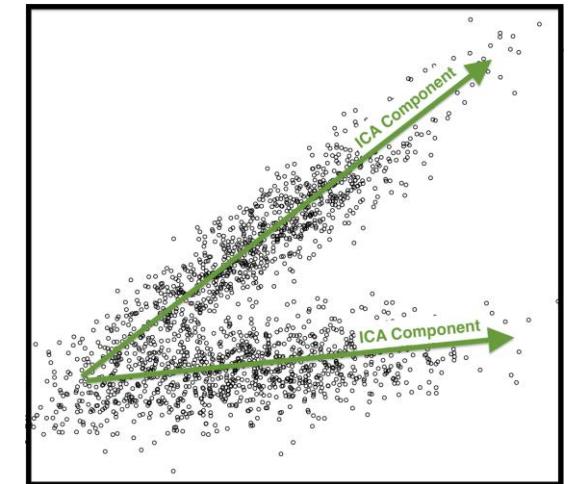


PCA Assumptions

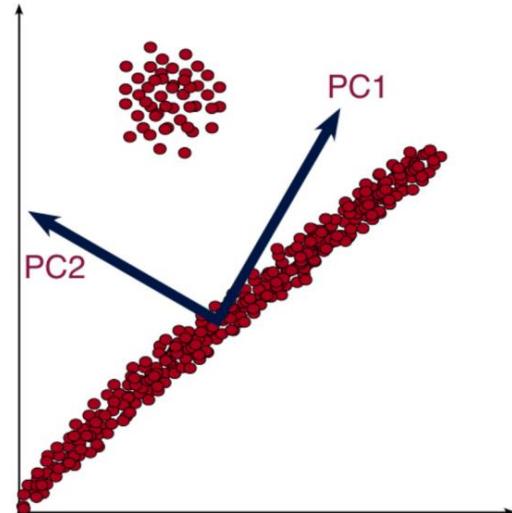
1. Linearity: PCA assumes that the relationships between variables are linear.



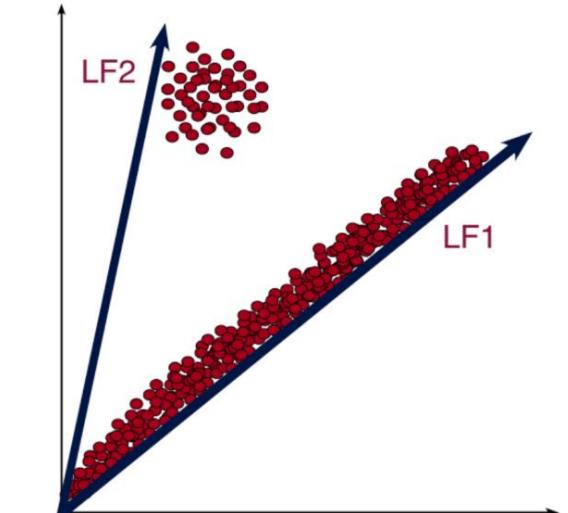
2. Orthogonality: Principal components are orthogonal (uncorrelated) to each other.



3. Large Variance Indicates Importance: It assumes that components with larger variance are more important.



4. Gaussian Distribution: Assumes the data is Gaussian distributed, although this is not strictly necessary.



PCA Assumptions

- 1. Linearity:** PCA assumes that the relationships between variables are linear.

- 2. Orthogonality:** Principal components are orthogonal (uncorrelated) to each other.

- 3. Large Variance Indicates Importance:** It assumes that components with larger variance are more important.

- 4. Gaussian Distribution:** Assumes the data is Gaussian distributed, although this is not strictly necessary.

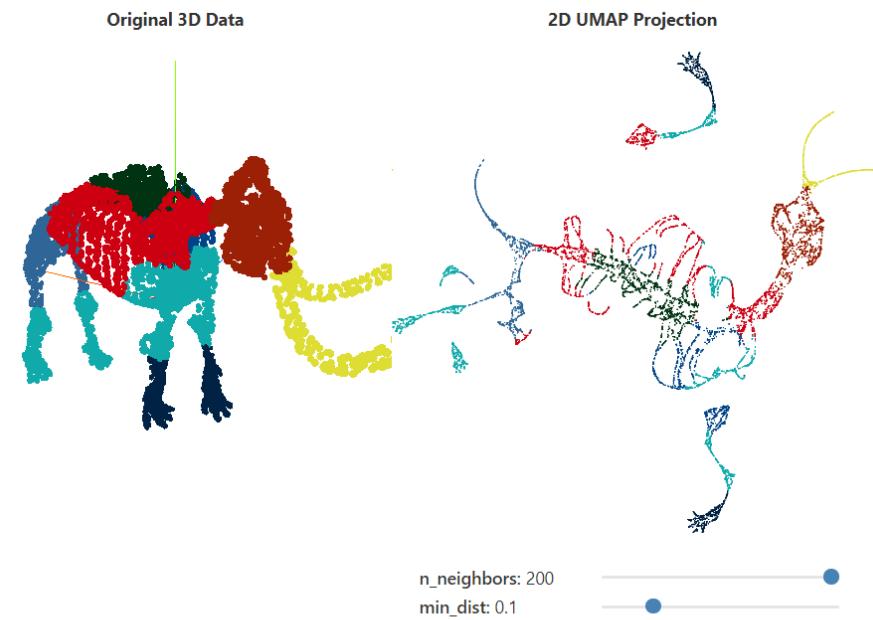
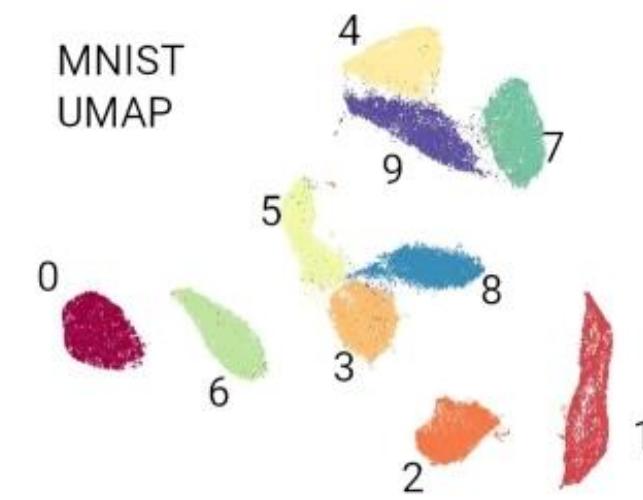
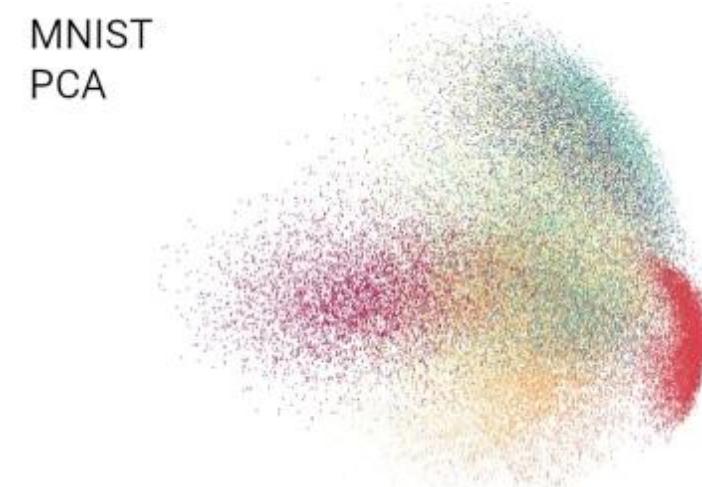


Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` and `min_dist` parameters.

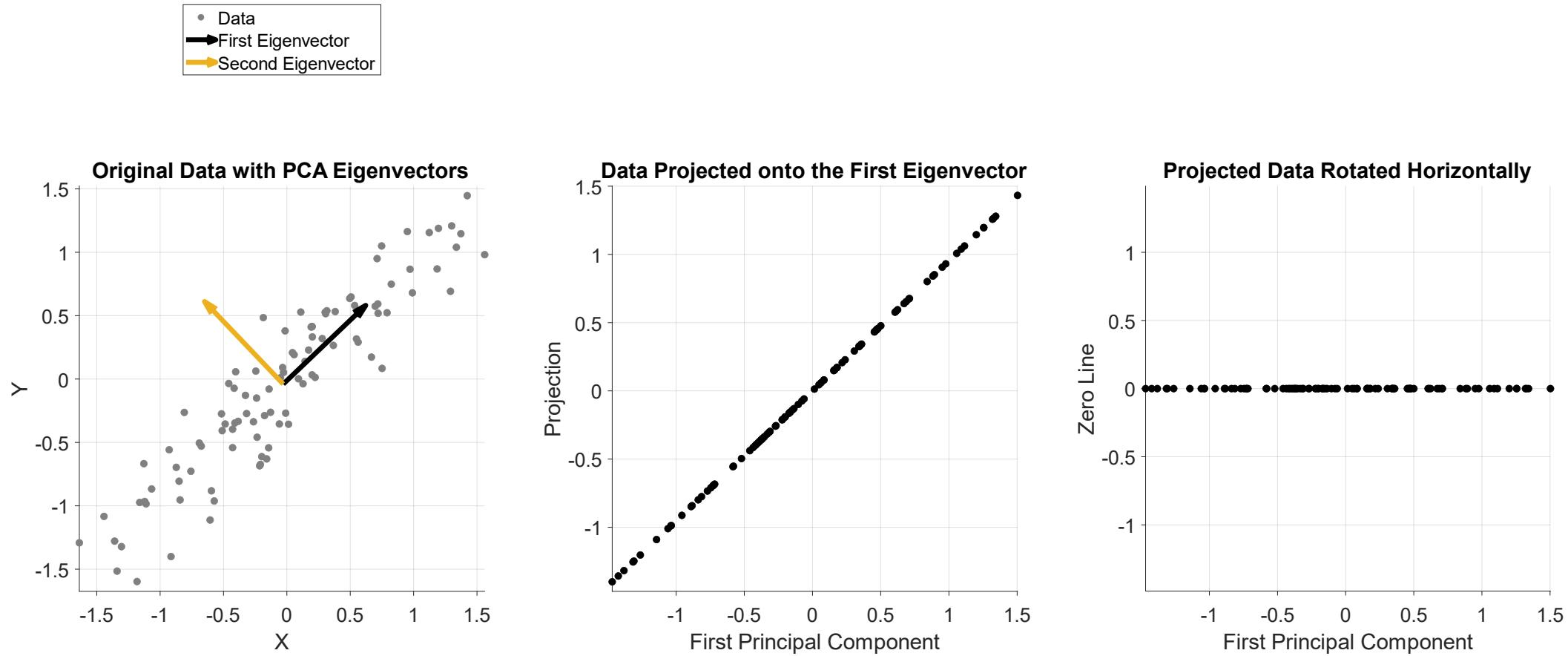
<https://pair-code.github.io/understanding-umap/>

PCA and UMAP performance on MNIST dataset

0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9

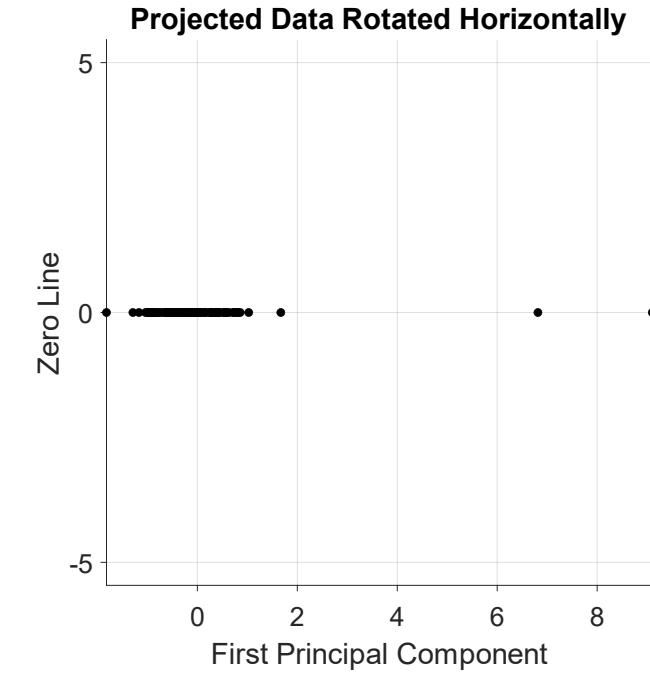
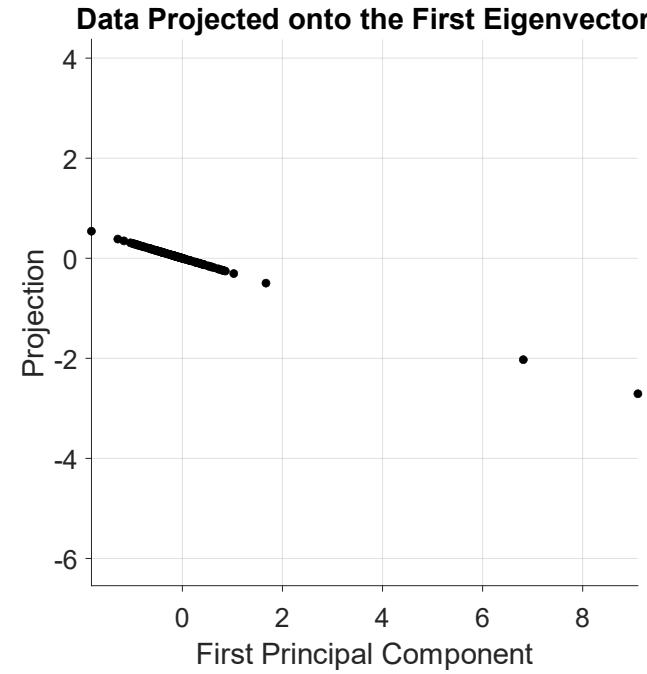
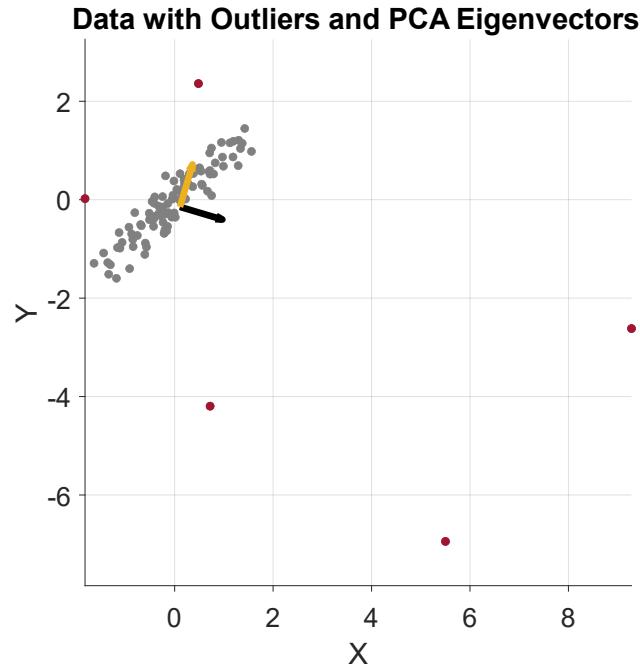


PCA on linearly correlated data



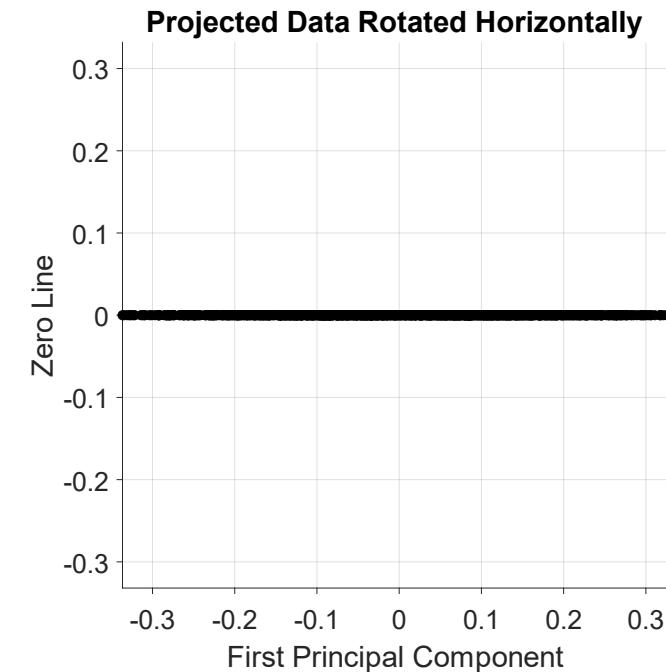
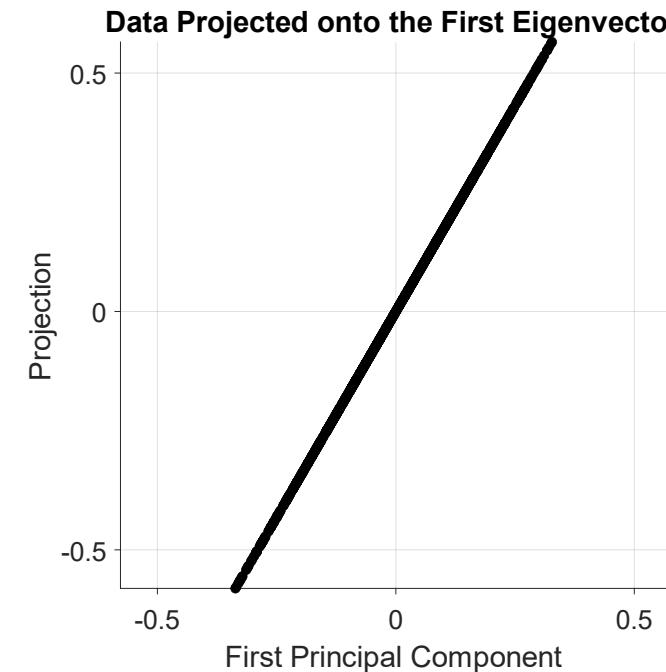
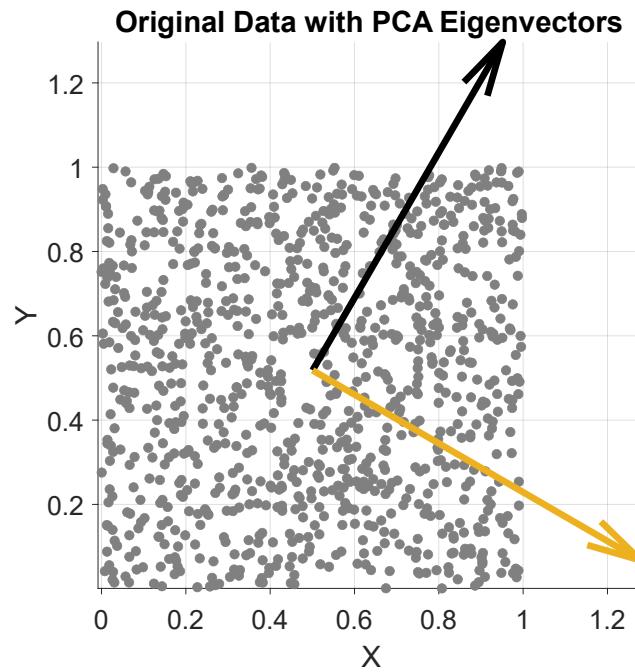
PCA on linearly correlated data with outliers

- Data
- Outliers
- First Eigenvector
- Second Eigenvector



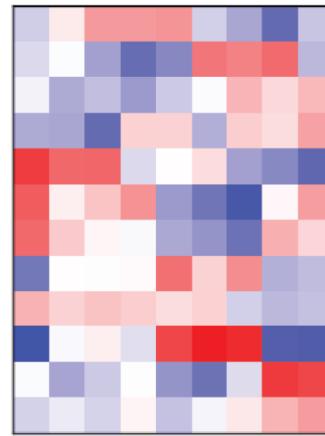
PCA on random noise

- Data
- First Eigenvector
- Second Eigenvector



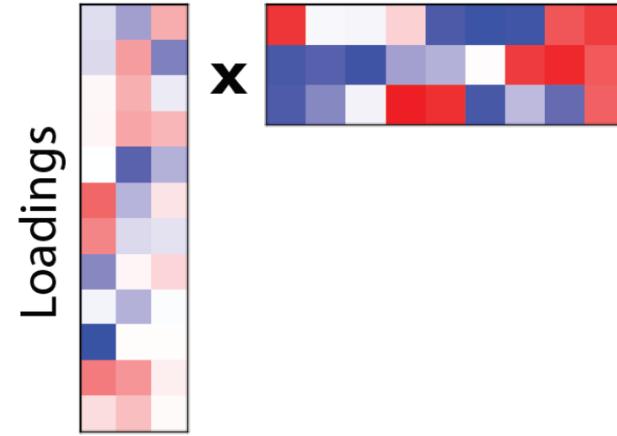


Original Data

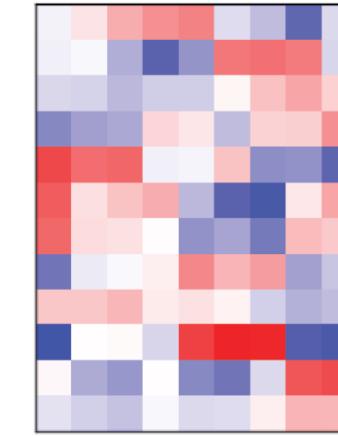


\approx

Components

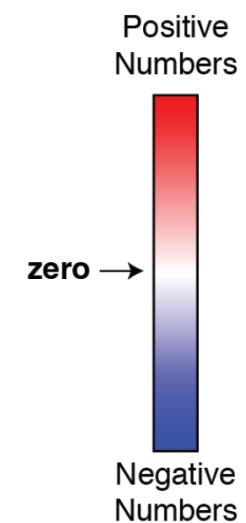
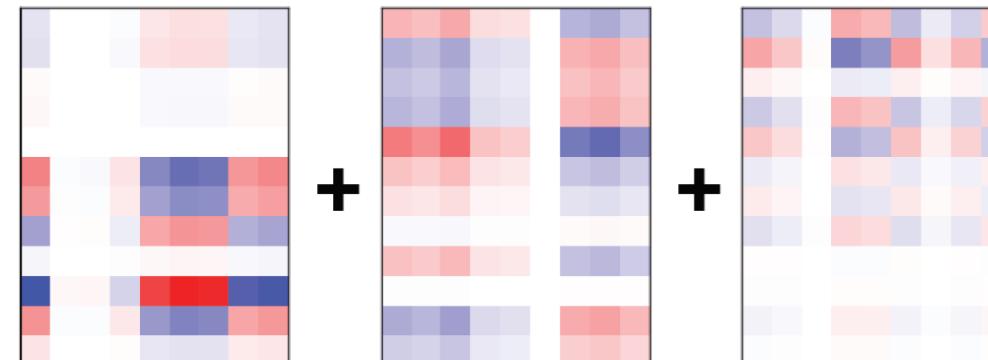


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax



neurosyntax

Session 3

Lecturer: Saman Abbaspoor

Principal Component Analysis (PCA)

Computation

$$\begin{matrix} \text{Data transposed} \\ (\text{mean-subtracted}) \end{matrix} \times \begin{matrix} \text{Data} \\ (\text{mean-subtracted}) \end{matrix} = \begin{matrix} \text{Covariance} \\ \text{matrix} \end{matrix} = \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \\ \text{transposed} \end{matrix} \times \begin{matrix} \text{Eigenvalues} \\ (\lambda) \end{matrix} \times \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \end{matrix}$$

Dimensionality Reduction

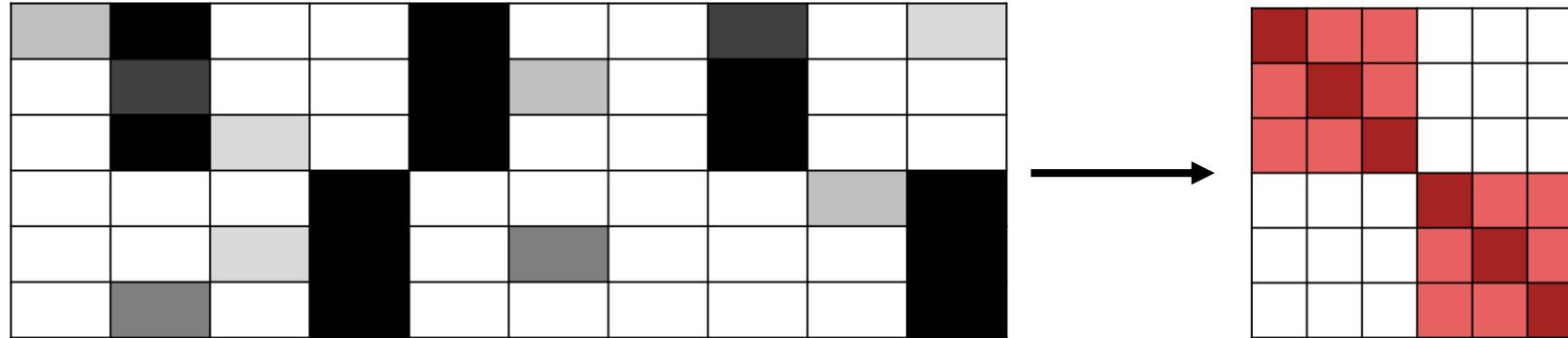
$$\begin{matrix} \text{Data} \\ (\text{mean-subtracted}) \end{matrix} \times \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \\ \text{transposed} \end{matrix} = \begin{matrix} \text{Scores} \end{matrix}$$

Principal Component Analysis (PCA)

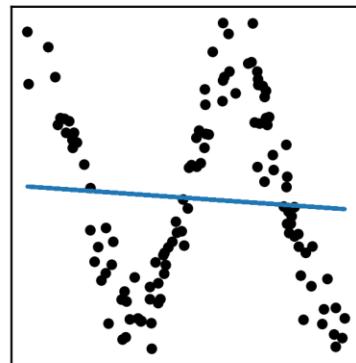
PCA is one of the best-known technique in multivariate analysis, tackling data compression and statistical pattern recognition in a dataset. **The idea is to extract features, making a domain transformation of the input in a way that a reduced number of features retaining most of the explanatory capacity can be selected.** The PCA algorithm is the optimum linear transformation in the mean square error sense, resulting in a set of linearly uncorrelated variables, the so-called principal components (PCs), representing the directions that maximize the rate of variance decrease.

Traditional PCA has limitations when dealing with high order dependencies, since it's a linear method, is independent of the data source, having no tunable parameters, and, when applied to classification problems, assigns each new sample strictly based on what was observed on the training period, not updating statistical properties dynamically.

Information beyond covariance structure?

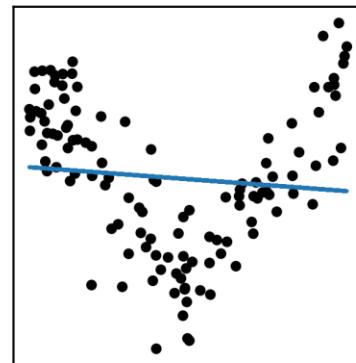


(a) $y = \sin(x) + \varepsilon$



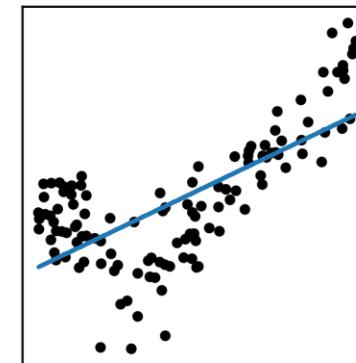
Original: -0.07
Transformed: 0.96
MI: 0.94

(b) $y = |x| + \varepsilon$



Original: -0.10
Transformed: 0.85
MI: 0.85

(c) $y = |x + 0.4| + \varepsilon$



Original: 0.70
Transformed: 0.89
MI: 0.89

PCA Applications



World Happiness Report

Measure Names

- Explained by: GDP per capita
- Explained by: Social support
- Explained by: Healthy life expectancy
- Explained by: Freedom to make life choices
- Explained by: Generosity
- Explained by: Perceptions of corruption
- Dystopia (2.43) + residual

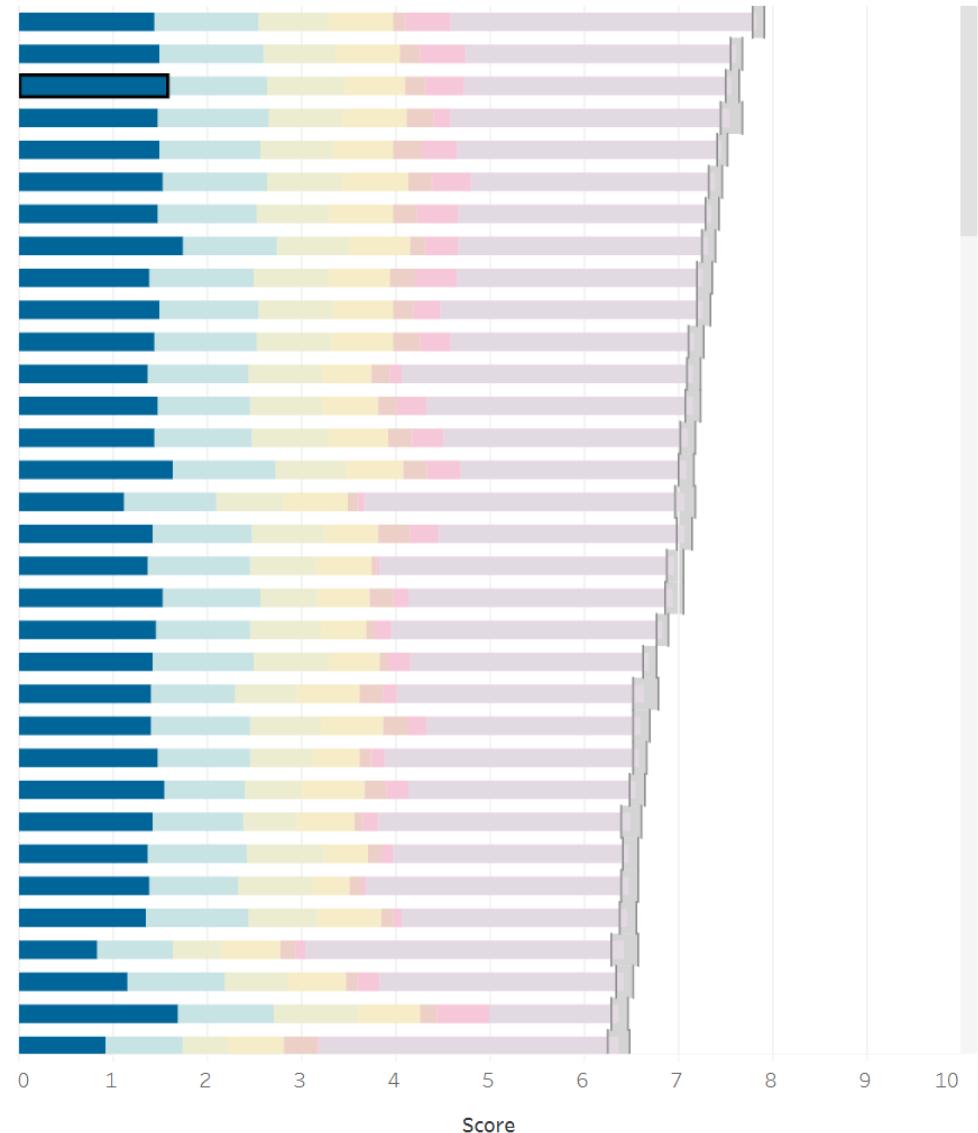
<https://worldhappiness.report/ed/2021/happiness-trust-and-deaths-under-covid-19/>

Official World Happiness Report 2021 Ranking

Figure 2.1 Ranking of Happiness based on a three-year-average 2018-2020.

Country

1. Finland (7.842)
2. Denmark (7.620)
3. Switzerland (7.571)
4. Iceland (7.554)
5. Netherlands (7.464)
6. Norway (7.392)
7. Sweden (7.363)
8. Luxembourg (7.324)*
9. New Zealand (7.277)
10. Austria (7.268)
11. Australia (7.183)
12. Israel (7.157)
13. Germany (7.155)
14. Canada (7.103)
15. Ireland (7.085)
16. Costa Rica (7.069)*
17. United Kingdom (7.064)
18. Czech Republic (6.965)
19. United States (6.951)
20. Belgium (6.834)
21. France (6.690)
22. Bahrain (6.647)
23. Malta (6.602)
24. Taiwan Province of China (6.584)
25. United Arab Emirates (6.561)
26. Saudi Arabia (6.494)
27. Spain (6.491)
28. Italy (6.483)
29. Slovenia (6.461)
30. Guatemala (6.435)*
31. Uruguay (6.431)
32. Singapore (6.377)*
33. Kosovo (6.372)



World Happiness Report

Measure Names

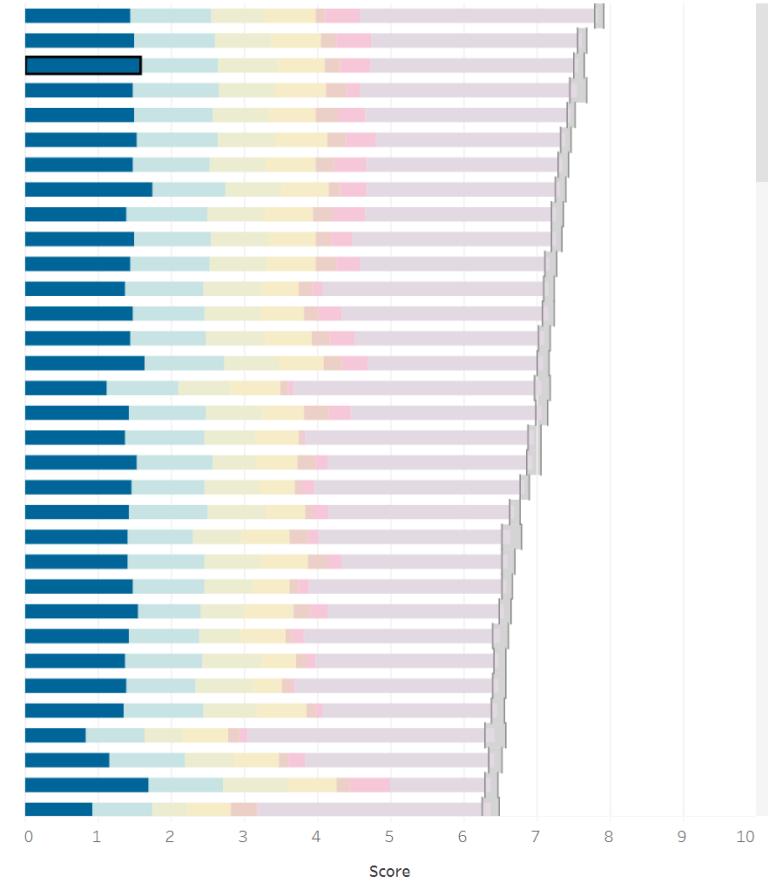
- Explained by: GDP per capita
- Explained by: Social support
- Explained by: Healthy life expectancy
- Explained by: Freedom to make life choices
- Explained by: Generosity
- Explained by: Perceptions of corruption
- Dystopia (2.43) + residual

Official World Happiness Report 2021 Ranking

Figure 2.1 Ranking of Happiness based on a three-year-average 2018-2020.

Country

1. Finland (7.842)
2. Denmark (7.620)
3. Switzerland (7.571)
4. Iceland (7.554)
5. Netherlands (7.464)
6. Norway (7.392)
7. Sweden (7.363)
8. Luxembourg (7.324)*
9. New Zealand (7.277)
10. Austria (7.268)
11. Australia (7.183)
12. Israel (7.157)
13. Germany (7.155)
14. Canada (7.103)
15. Ireland (7.085)
16. Costa Rica (7.069)*
17. United Kingdom (7.064)
18. Czech Republic (6.965)
19. United States (6.951)
20. Belgium (6.834)
21. France (6.690)
22. Bahrain (6.647)
23. Malta (6.602)
24. Taiwan Province of China (6.584)
25. United Arab Emirates (6.561)
26. Saudi Arabia (6.494)
27. Spain (6.491)
28. Italy (6.483)
29. Slovenia (6.461)
30. Guatemala (6.435)*
31. Uruguay (6.431)
32. Singapore (6.377)*
33. Kosovo (6.372)



$$\begin{aligned} \text{PC1} = & 0.538 \textit{Social} + 0.563 \textit{Life} + 0.498 \textit{Choices} \\ & - 0.004 \textit{Generosity} - 0.381 \textit{Corruption} \end{aligned}$$

$$\begin{aligned} \text{PC2} = & -0.266 \textit{Social} - 0.243 \textit{Life} + 0.258 \textit{Choices} \\ & + 0.799 \textit{Generosity} - 0.407 \textit{Corruption} \end{aligned}$$



BRUNO GRISCI · UPDATED 4 YEARS AGO

81

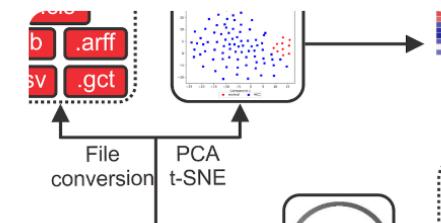
New Notebook

Download (65 MB)



Breast cancer gene expression - CuMiDa

GSE45827 microarray experiment



Data Card Code (13) Discussion (2) Suggestions (0)

About Dataset

Dataset GSE45827 on breast cancer gene expression from CuMiDa

- 6 classes
- 54676 genes
- 151 samples

About

Here we present the **Curated Microarray Database (CuMiDa)**, a repository containing 78 handpicked cancer microarray datasets, extensively curated from 30.000 studies from the Gene Expression Omnibus (GEO), solely for machine learning. The aim of **CuMiDa** is to offer homogeneous and state-of-the-art biological preprocessing of these datasets, together with numerous 3-fold cross validation benchmark results to propel machine learning studies focused on cancer research. The database make available various download options to be employed by other programs, as well for PCA and t-SNE results. **CuMiDa** stands different from existing databases for offering newer datasets, manually and carefully curated, from samples quality, unwanted probes, background correction and normalization, to create a more reliable source of data for computational research.

<http://sbcn.inf.ufrgs.br/cumida>

Usability

9.71

License

Database: Open Database, Cont...

Update frequency

Unspecified

Tags

Health

Biology

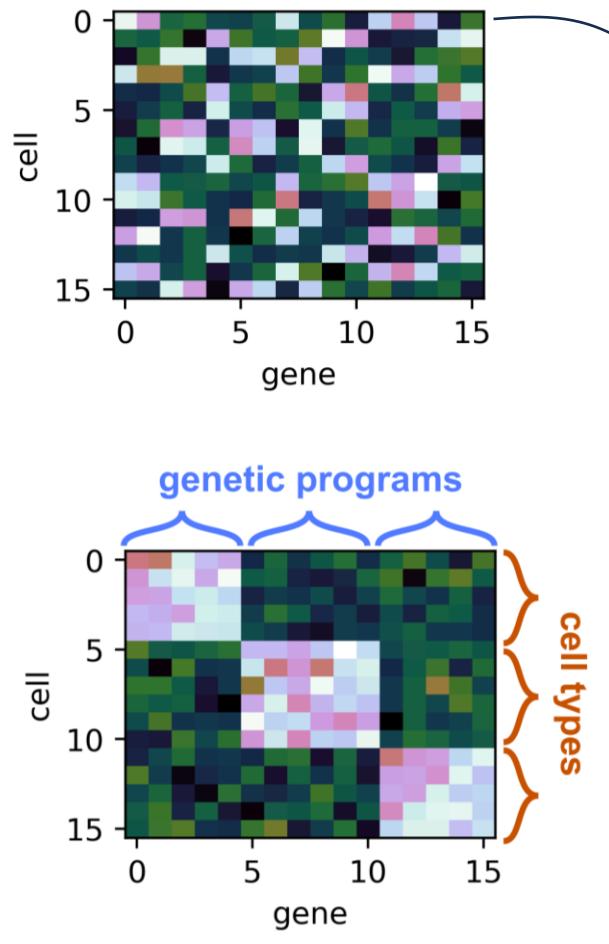
Cancer

Health Conditions

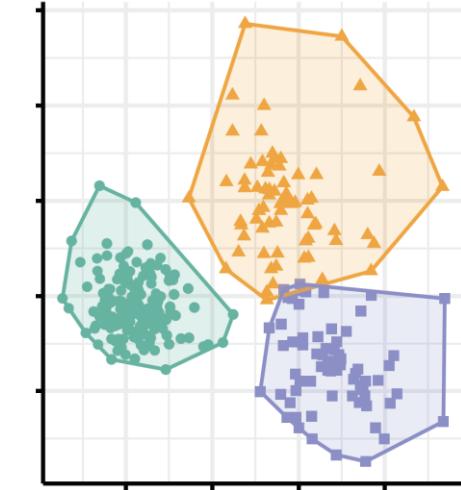
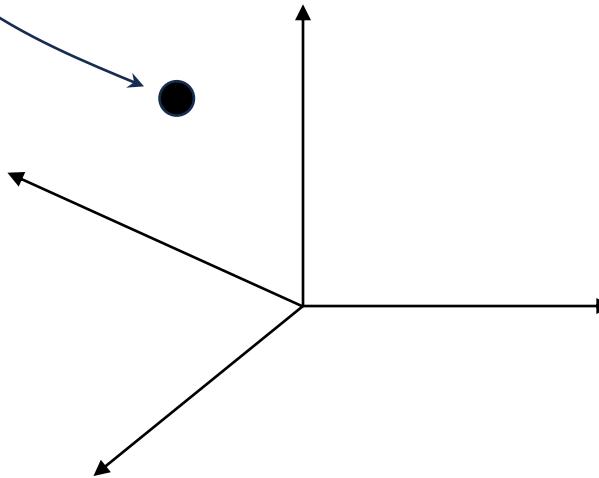
Classification

Diseases

Visualization and Clustering



N-dimensional space





Download

Complete normalized microarray datasets

Complete microarray datasets for the full complement of six brains are available for download. The datasets contain gene expression values normalized across all brains using an improved normalization process implemented in March 2013.

Downloadable files containing all normalized microarray expression values as well as probe and sample metadata necessary for analysis: [H0351.2001](#), [H0351.2002](#), [H0351.1009](#), [H0351.1012](#), [H0351.1015](#), [H0351.1016](#).

An [XML](#) or [CSV](#) link to meta-information and URL to download each of the raw microarray files.

RNA-Sequencing datasets

RNA-Sequencing datasets from two brains are available for download. These datasets contain gene expression values (raw and TPM counts) for a selected set of anatomic structures matched across the two brains, as well as sample and gene metadata necessary for analysis: [H0351.2001](#), [H0351.2002](#).

Archived microarray datasets

Archived data files containing normalized microarray expression values employing a historical normalization method (available prior to March 2013): [H0351.2001](#), [H0351.2002](#), [H0351.1009](#), [H0351.1012](#).

For detailed descriptions of current and historical normalization processes, see the technical white paper, [Microarray Data Normalization](#).



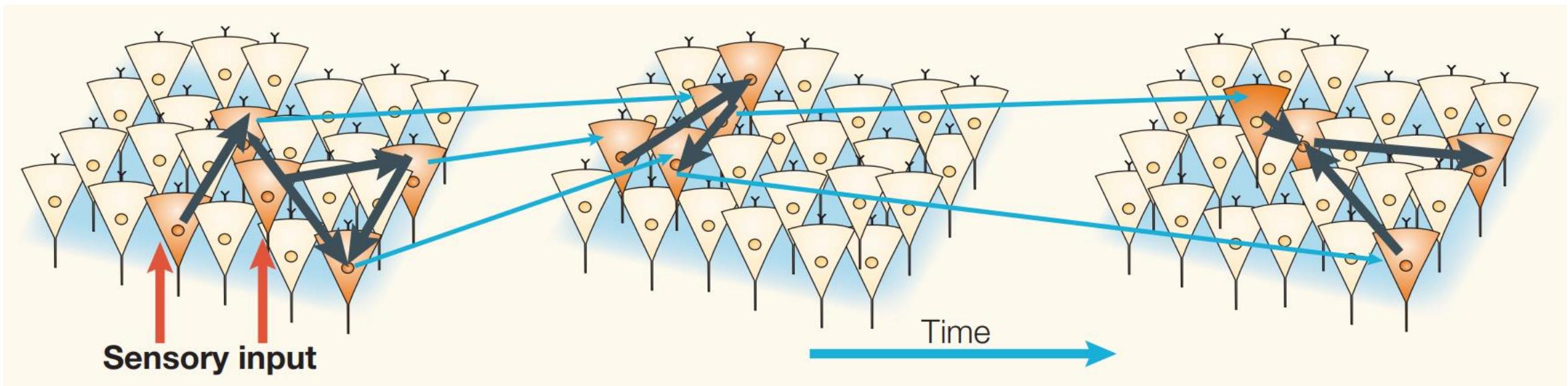
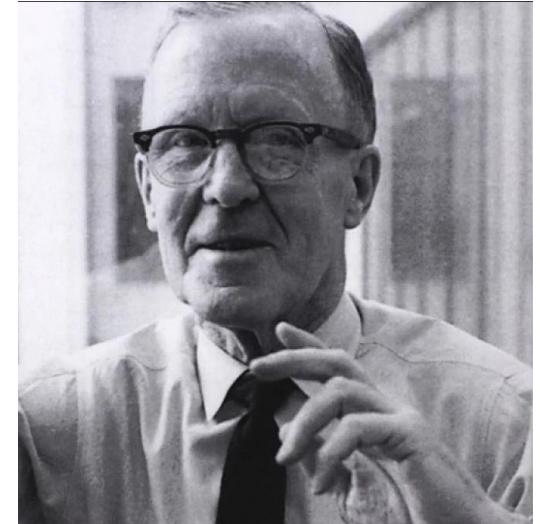
PCA Applications in Neuroscience

(Patterns in Timeseries)

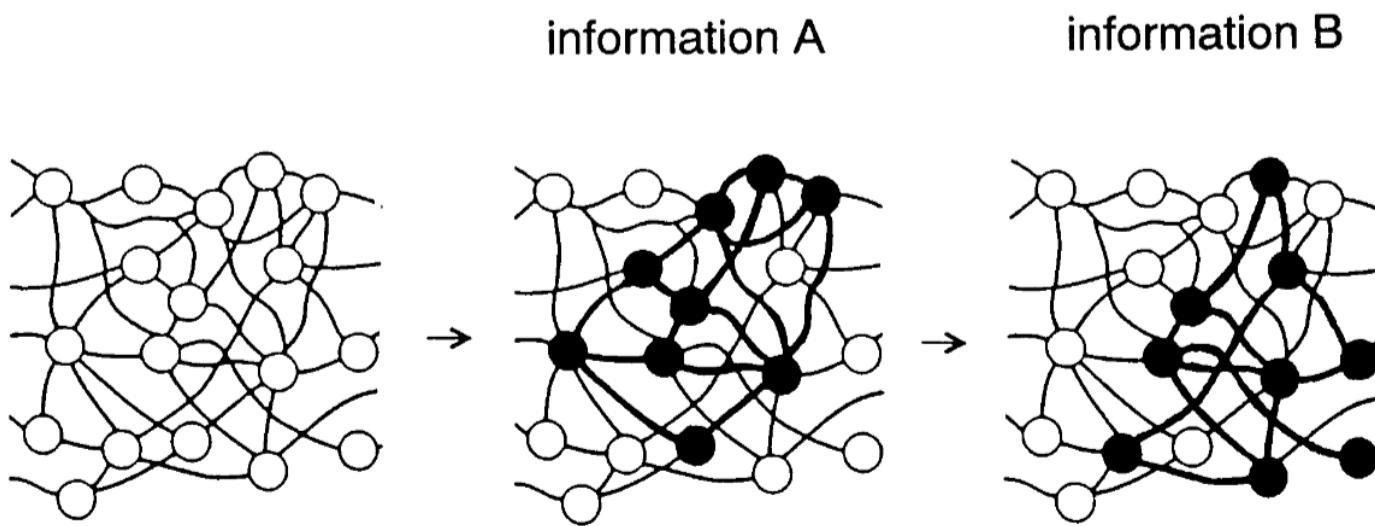
Cell assemblies

Hebb's idea: cell assembly dynamics support cognitive processes

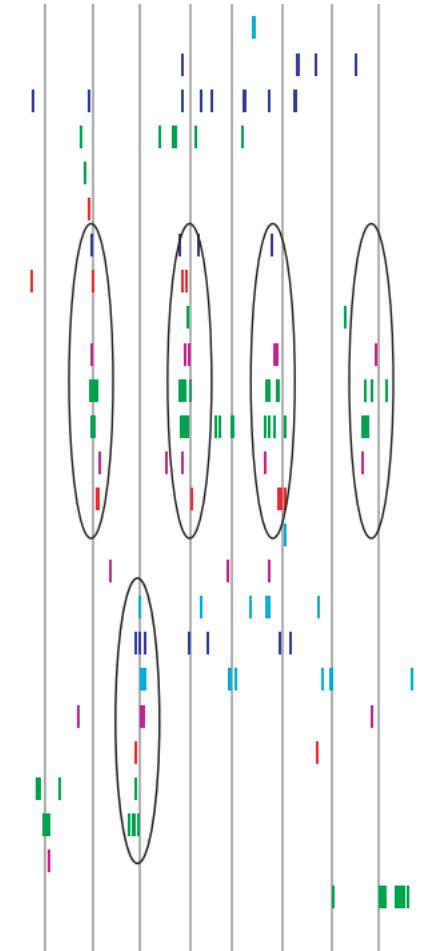
Cell assembly, defined as groups of neurons displaying recurring patterns of coordinated activity



Cell Assembly

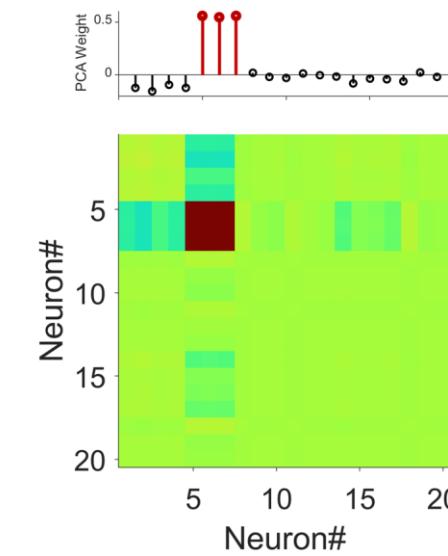
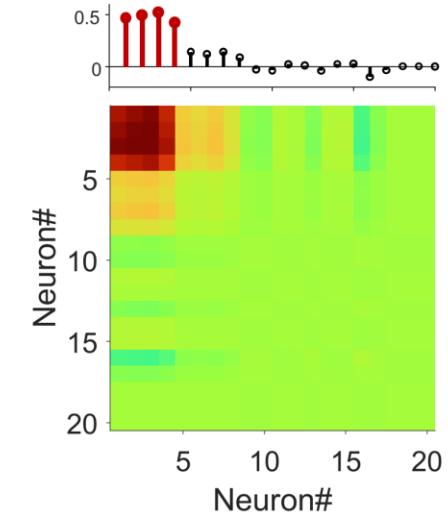
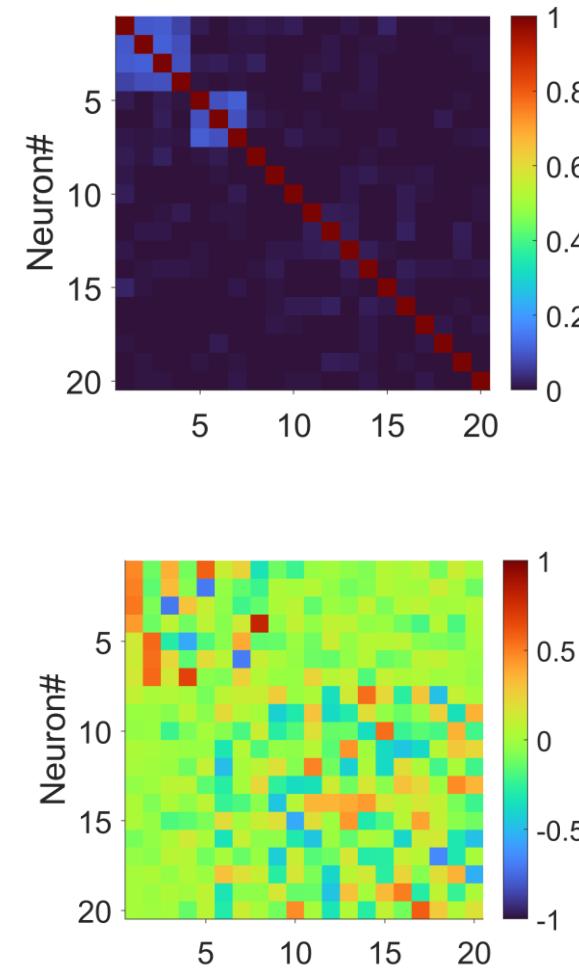
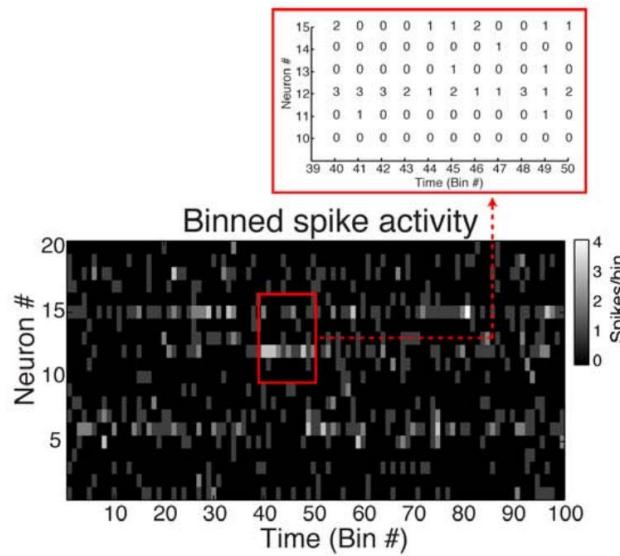
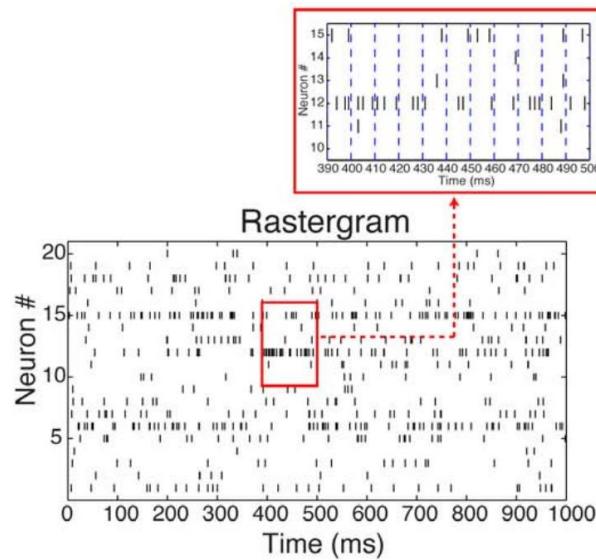


Sakurai, 1998

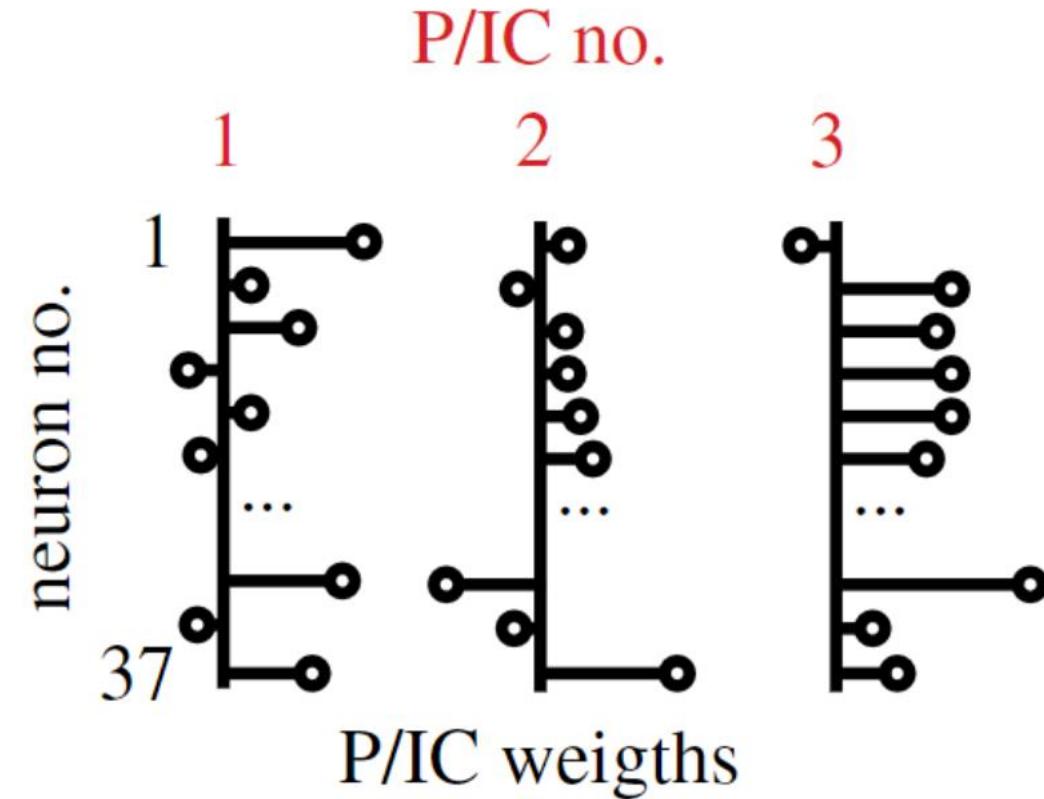
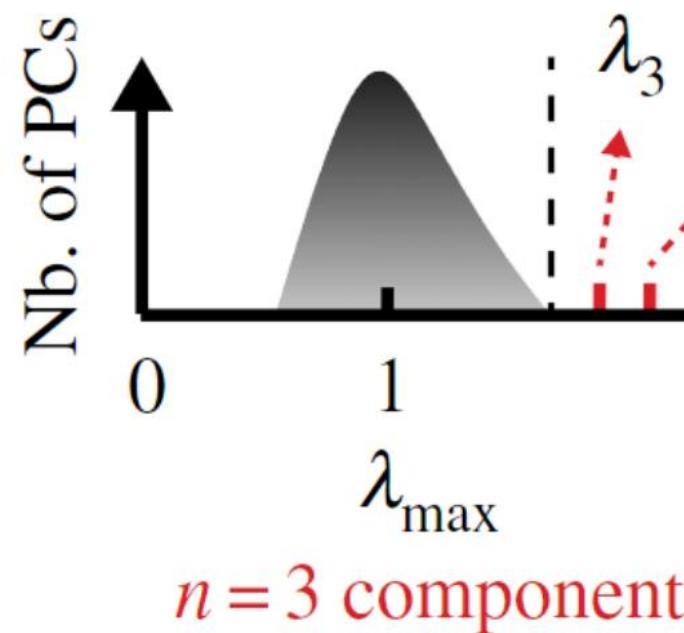


Harris et al., Nature 2003

Detecting cell assemblies in large neuronal populations



Detecting cell assemblies in large neuronal populations

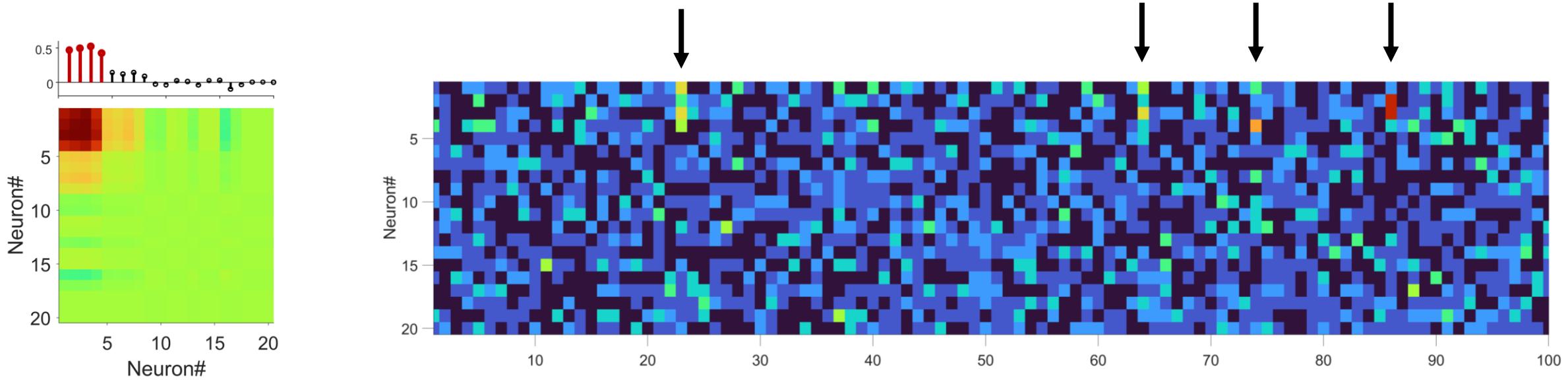


Lopes-dos-Santos et al., Plos One 2011; Lopes-dos-Santos, JNeuro Method 2013

Tingley and Peyrache et al., Phil. Trans. R. Soc. B 2020

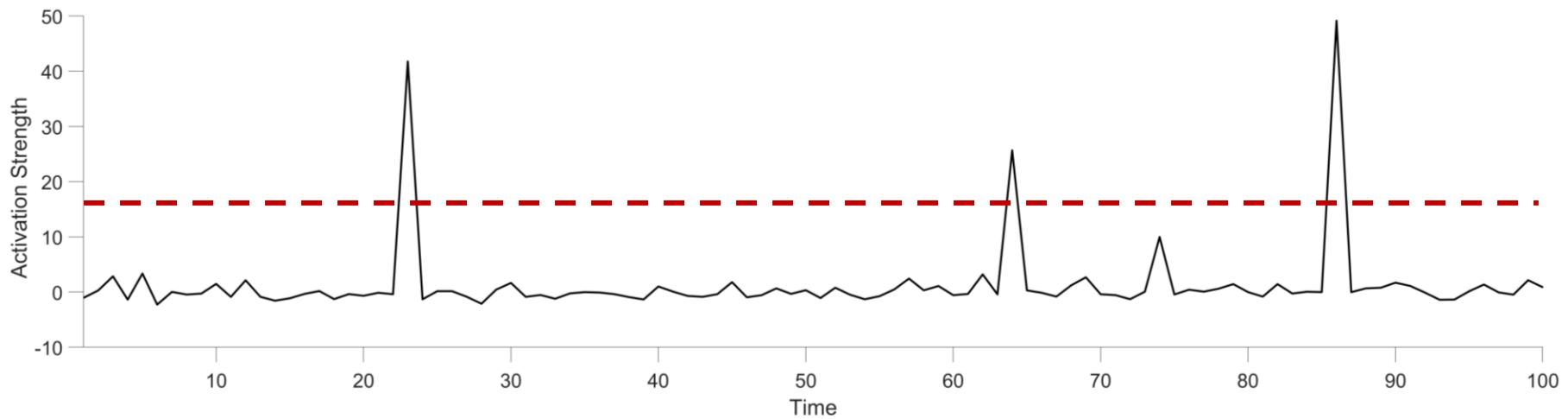
Makdah et al., BioRxiv 2024

Tracking cell assembly activation/reactivation over time



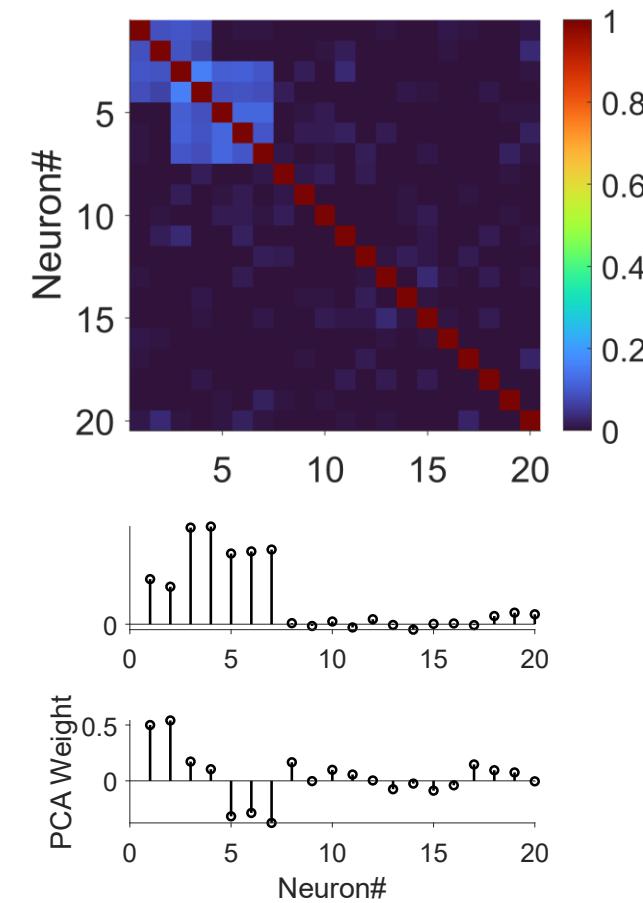
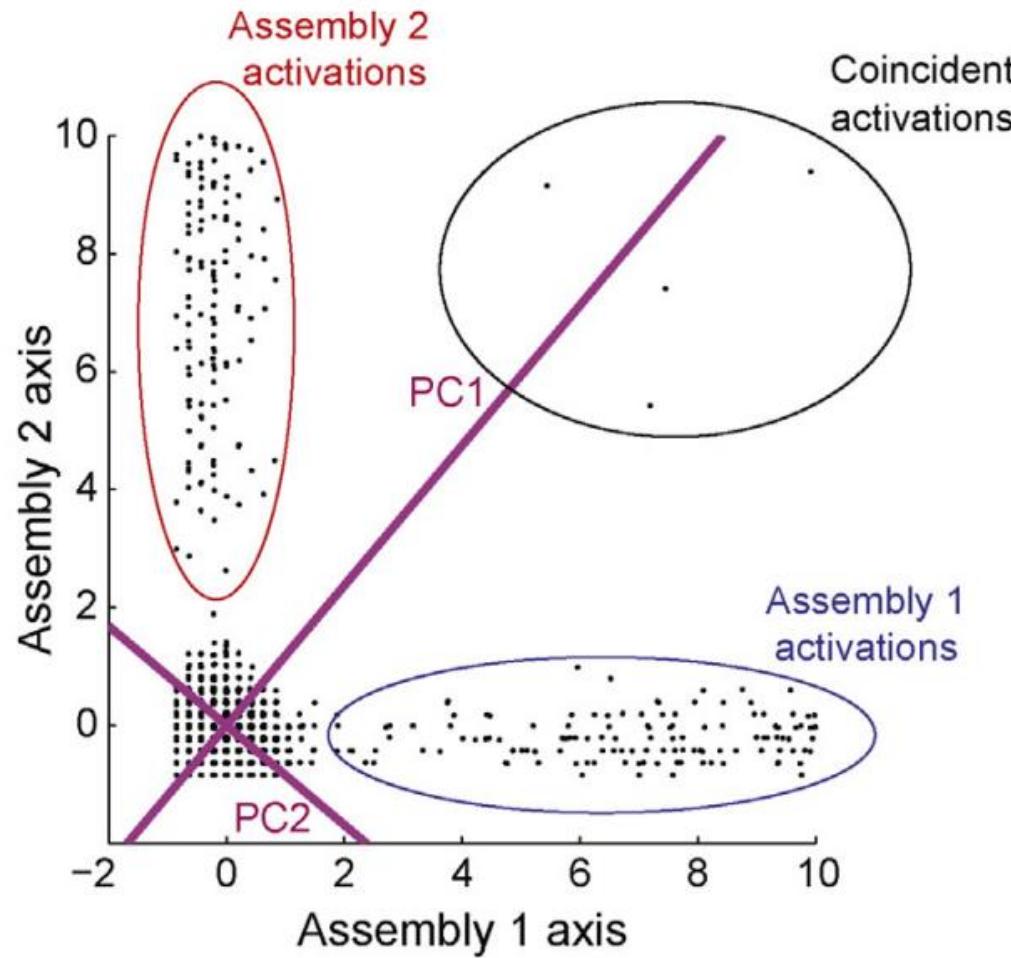
$$R_k(t) = z(t)^T P_k z(t)$$

Activation at each time point is the weighted average of firing of cells in the spike matrix with weights coming from each PCA/ICA component

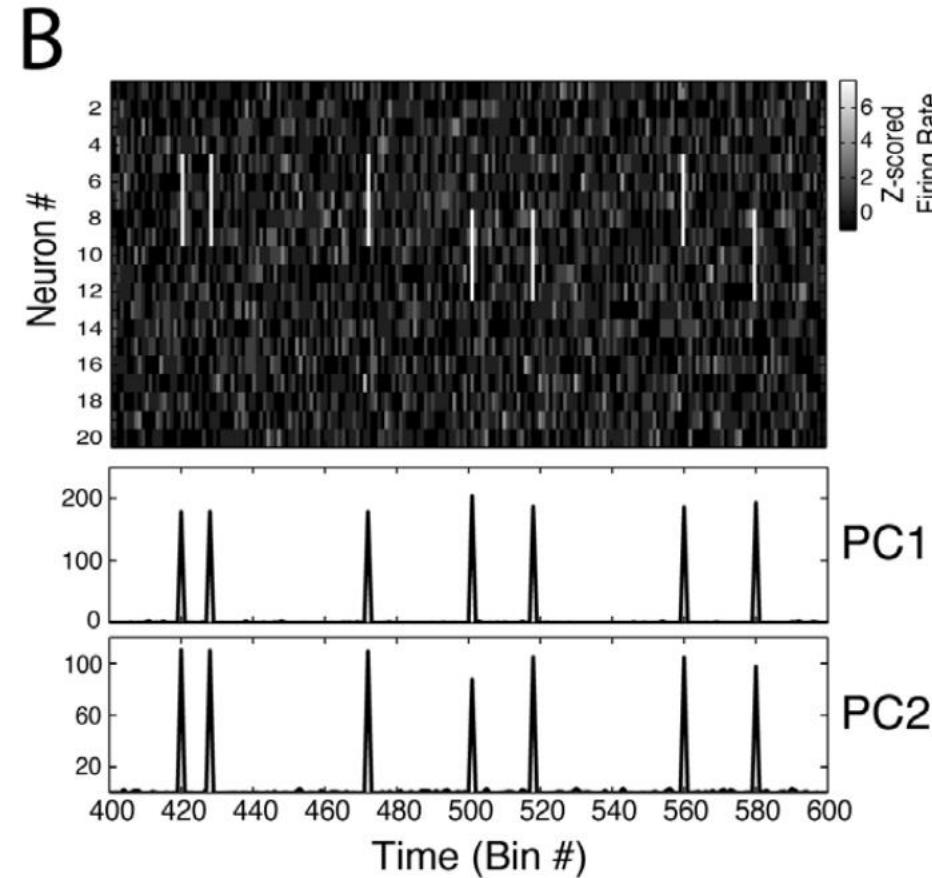
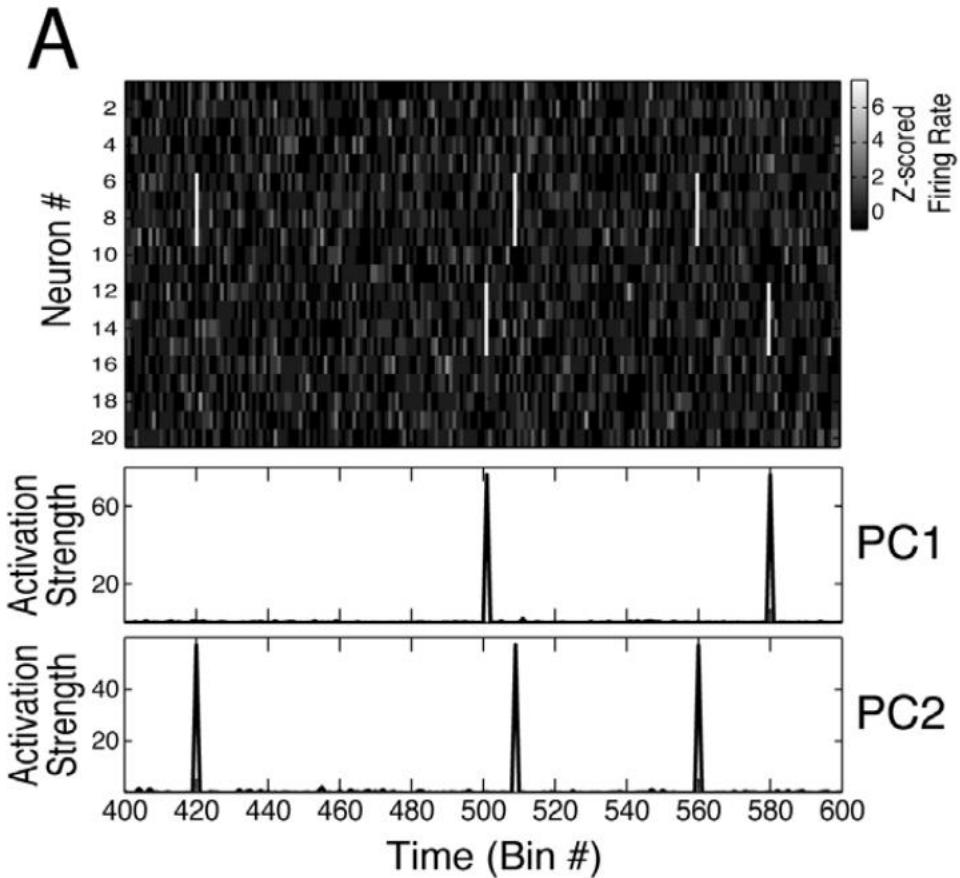




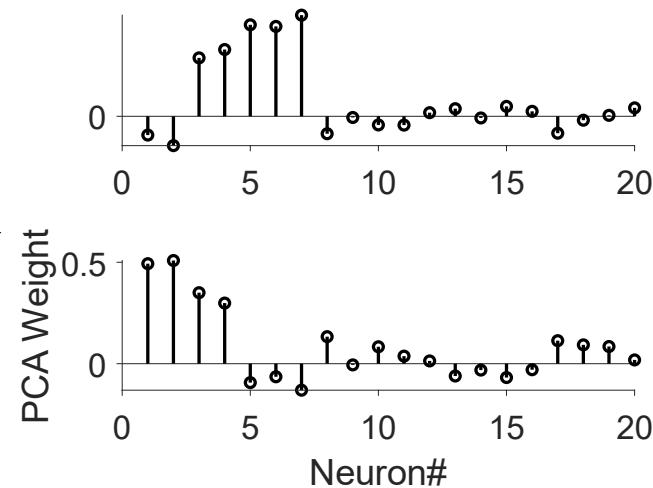
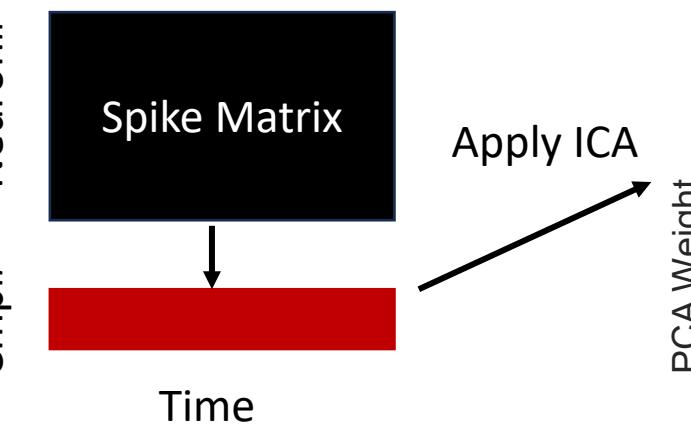
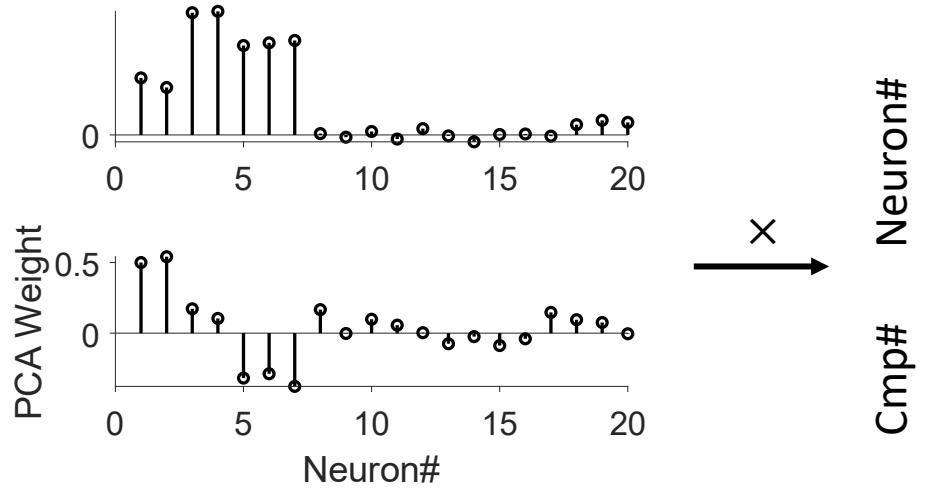
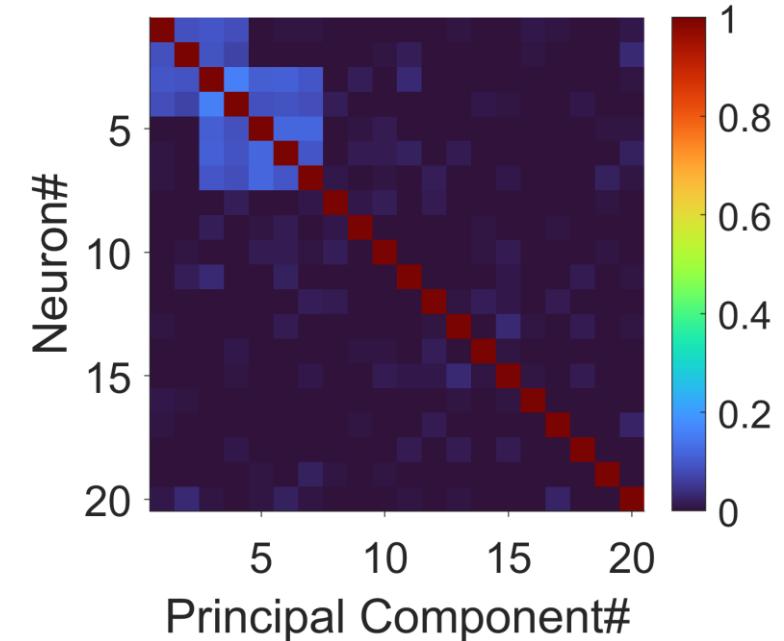
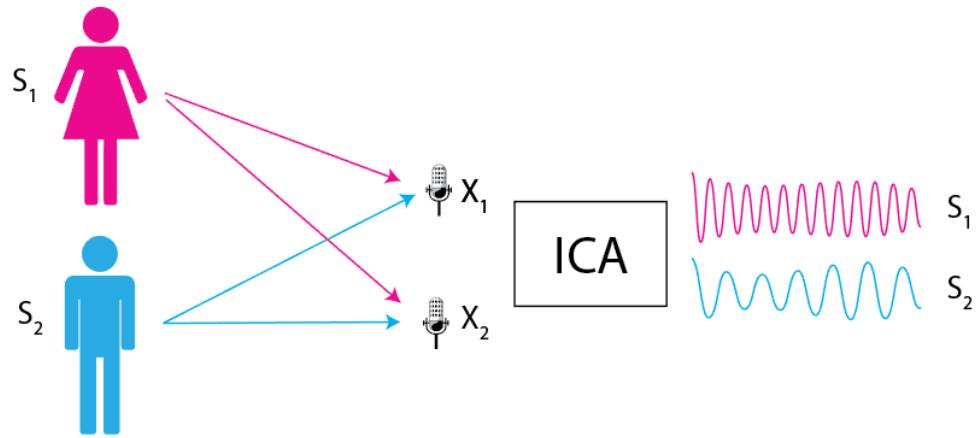
Membership overlap in cell assemblies: violation of orthogonality



Detecting cell assemblies in large neuronal populations

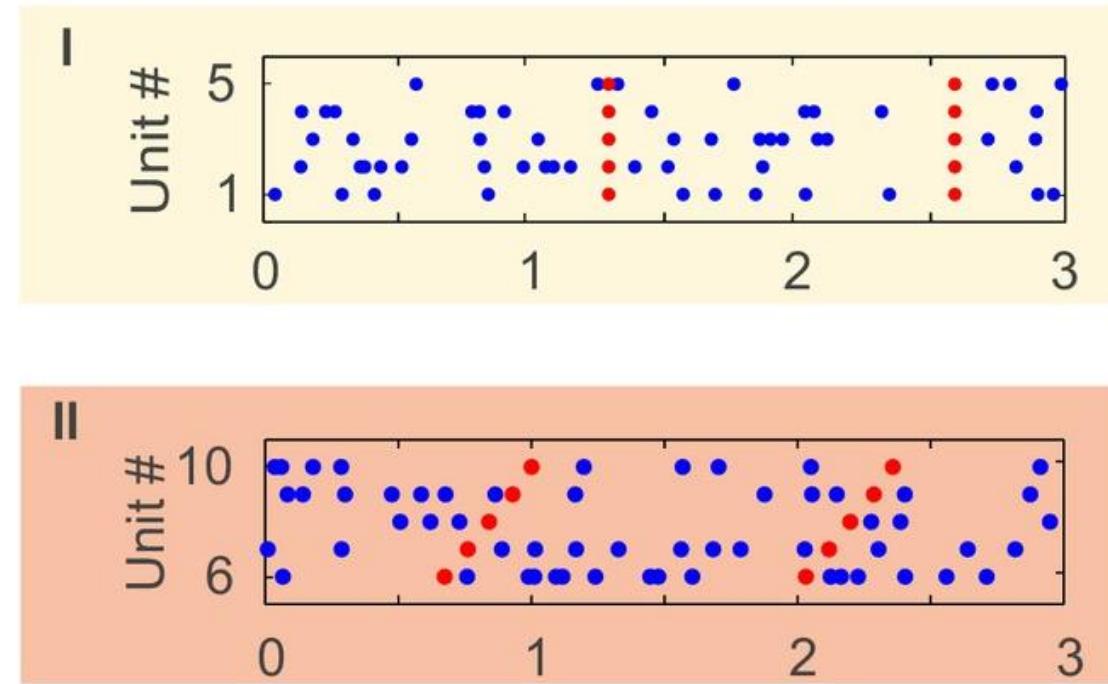


Detecting cell assemblies in large neuronal populations

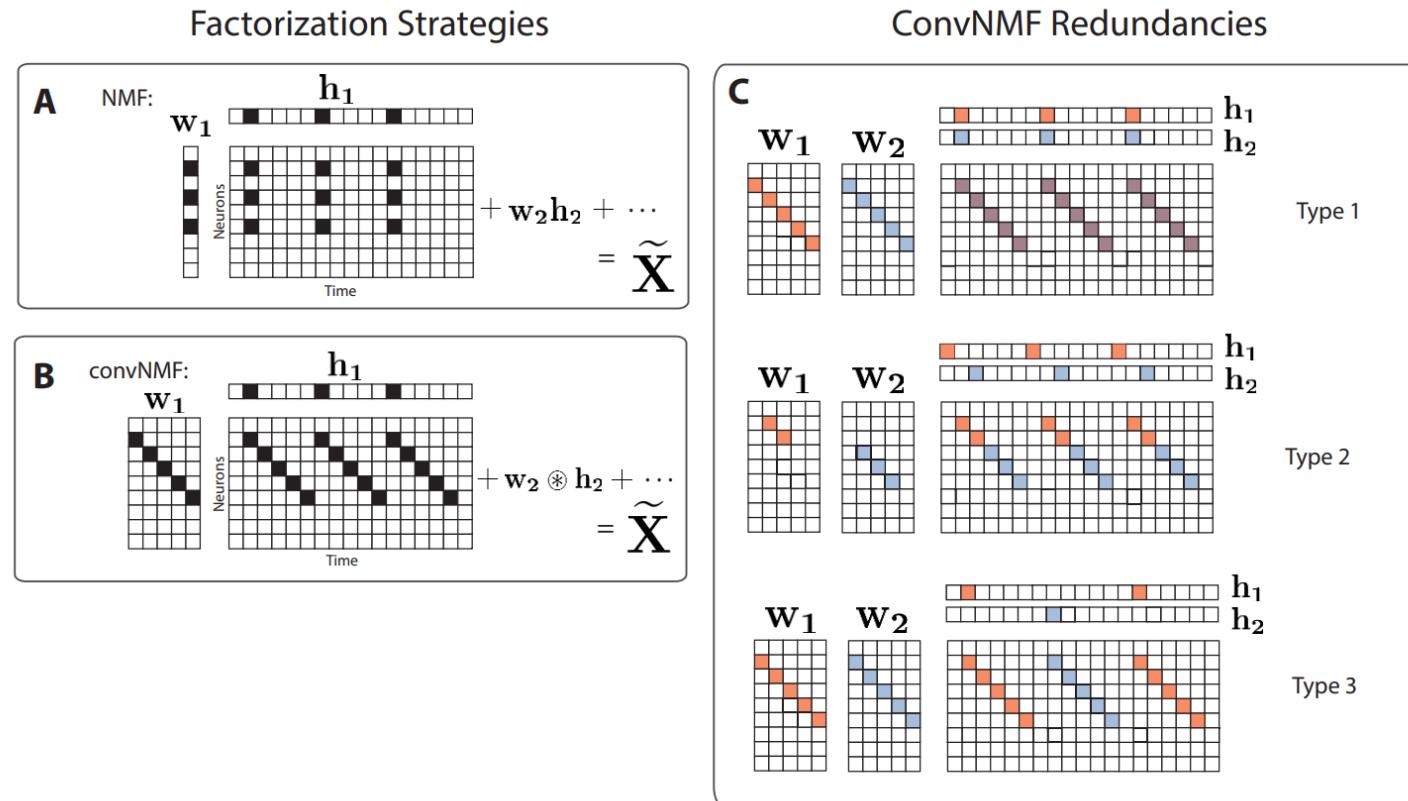




Cell assemblies at multiple time scales with arbitrary lag constellations

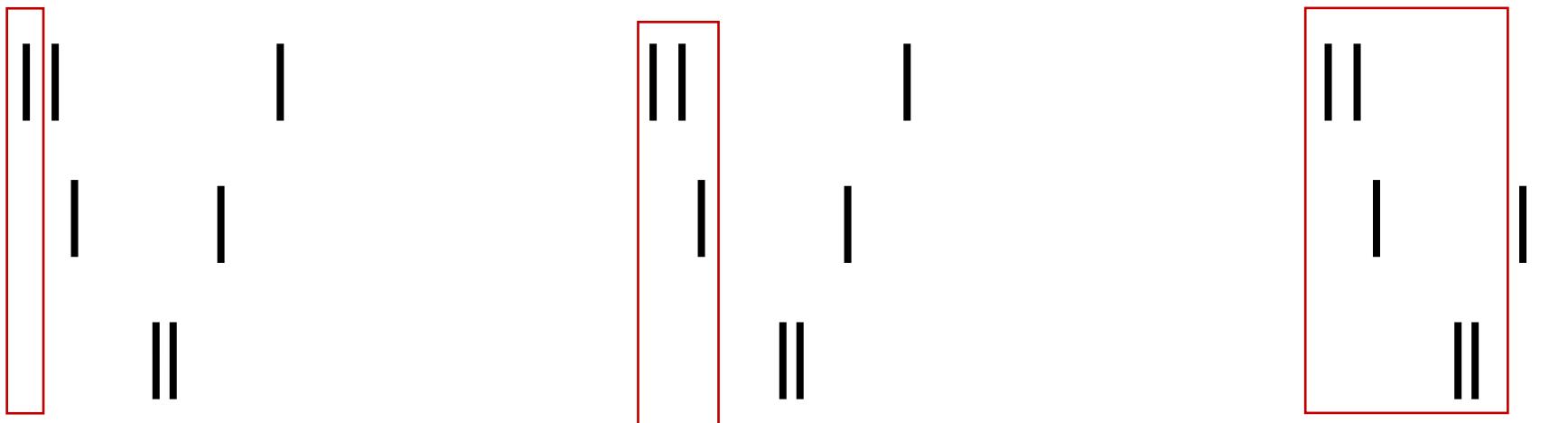


Unsupervised discovery of temporal sequences in high-dimensional datasets, with applications to neuroscience



Temporal smoothing

Time bin



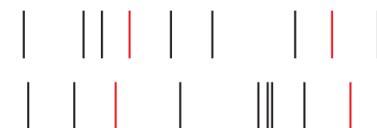
a No jitter



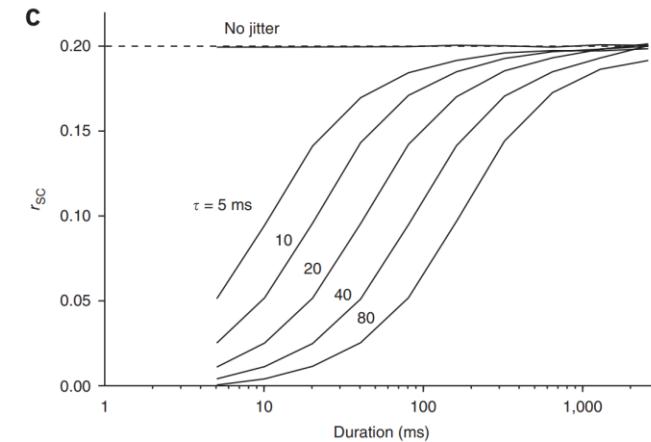
Spike train cross-correlogram



b Jitter

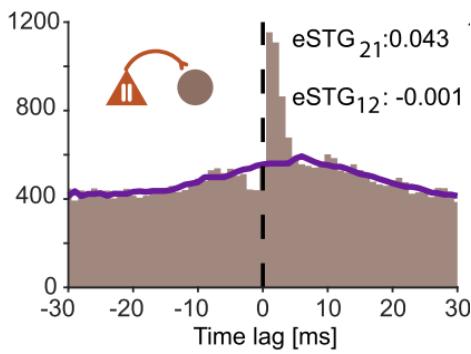


c



Physiologically interpretable and systematic estimation of cell assembly time constant

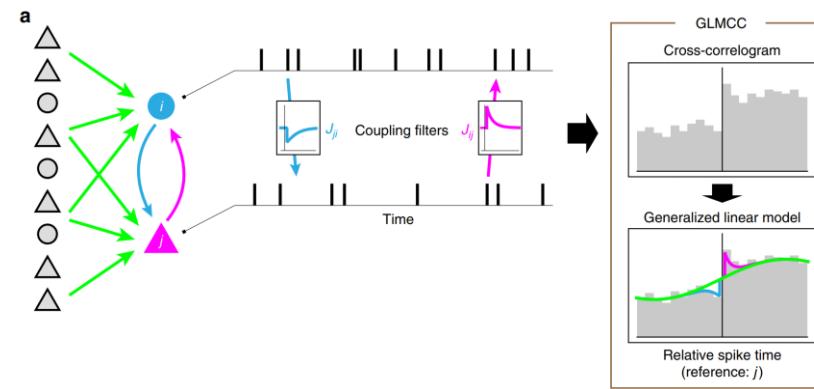
Pairwise cross-correlogram



Spivak et al., Nat Comm Bio, 2022

<https://github.com/EranStarkLab/CCH-deconvolution>

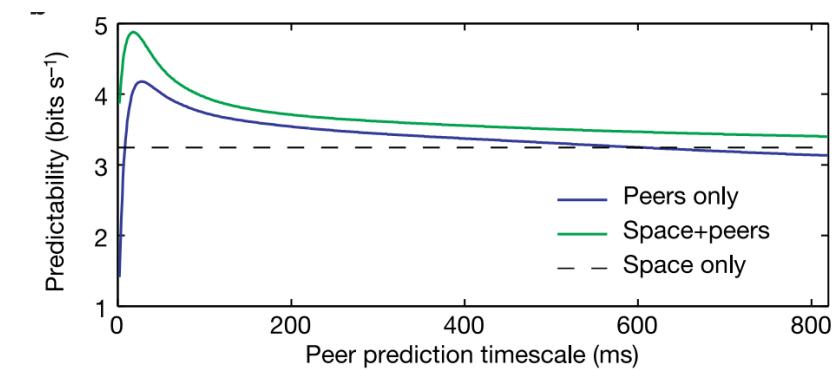
Generalized linear model (GLM)



Kobayashi et al., Nat Comm, 2019

<https://github.com/NII-Kobayashi/GLMCC>

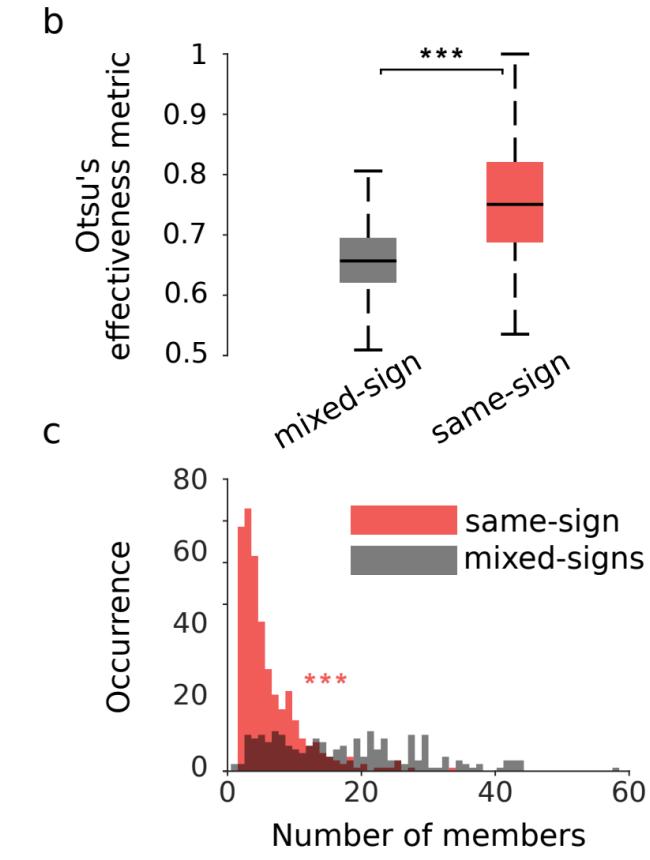
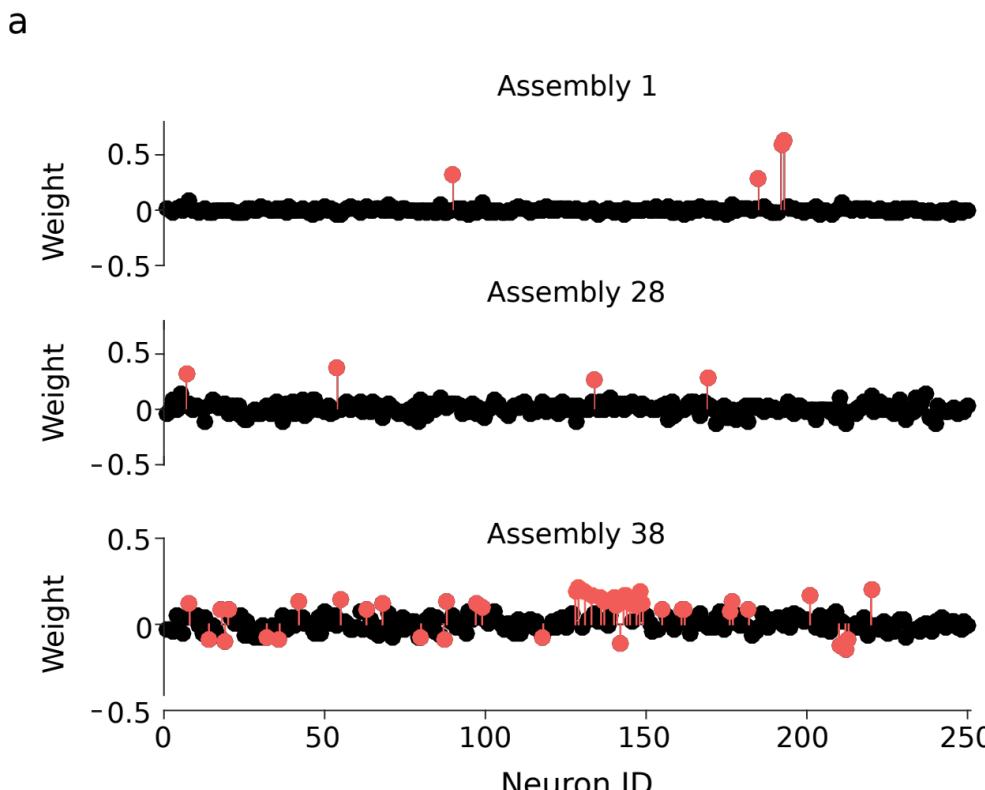
Peer Prediction



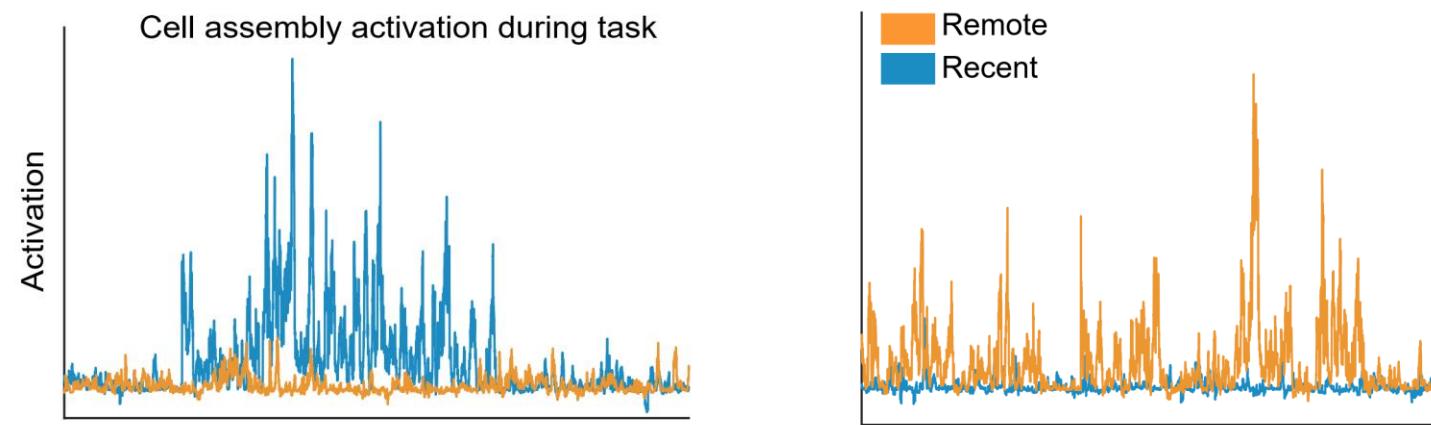
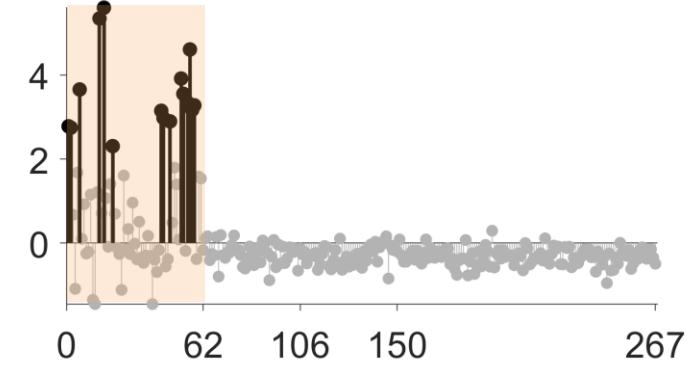
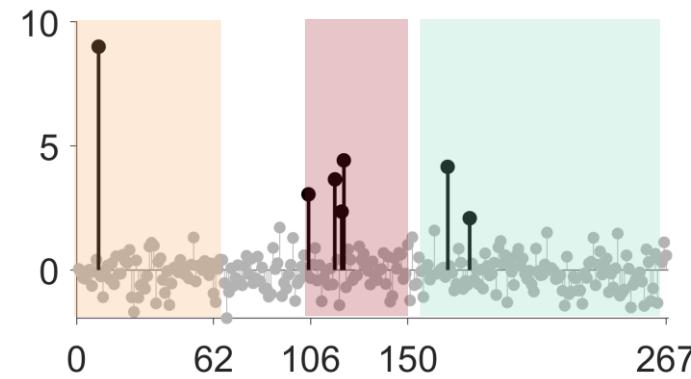
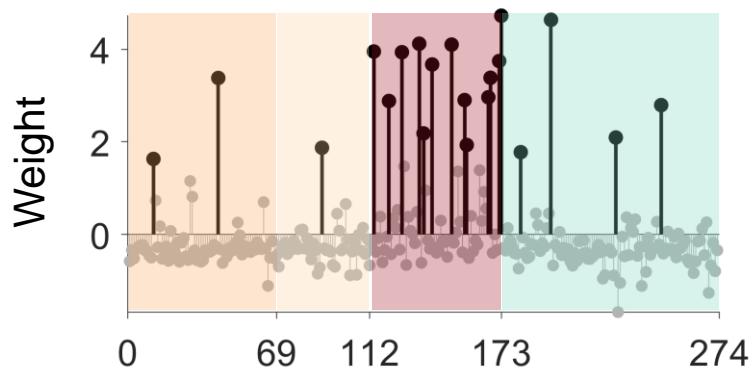
Harris et al., Nature, 2003

bz_peerPrediction.m
<https://github.com/buzsakilab/buzcode>

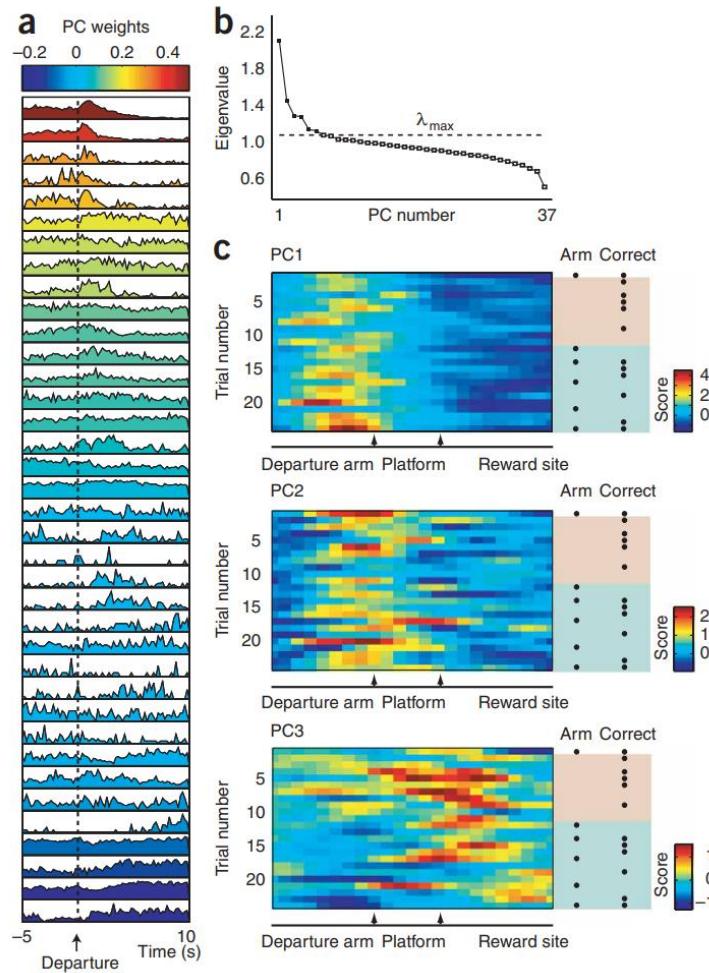
PCA for assemblies containing patterns of neural interactions with inhibition



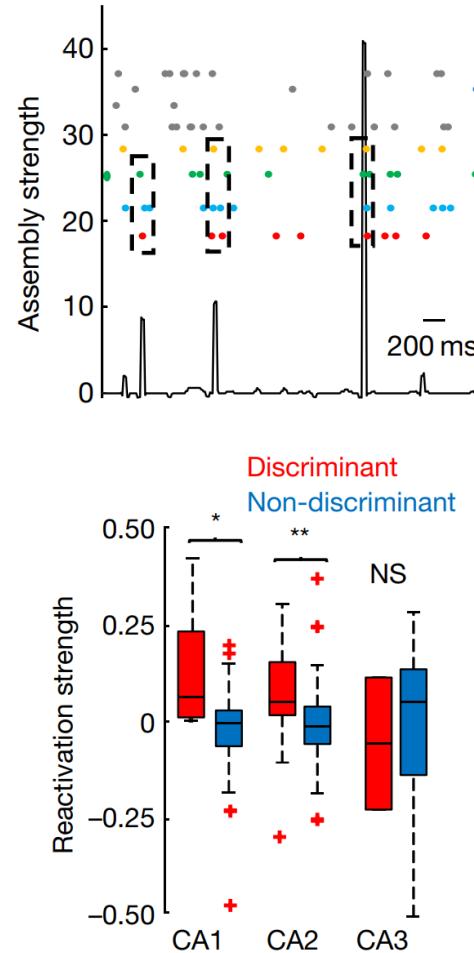
Cell assembly analysis



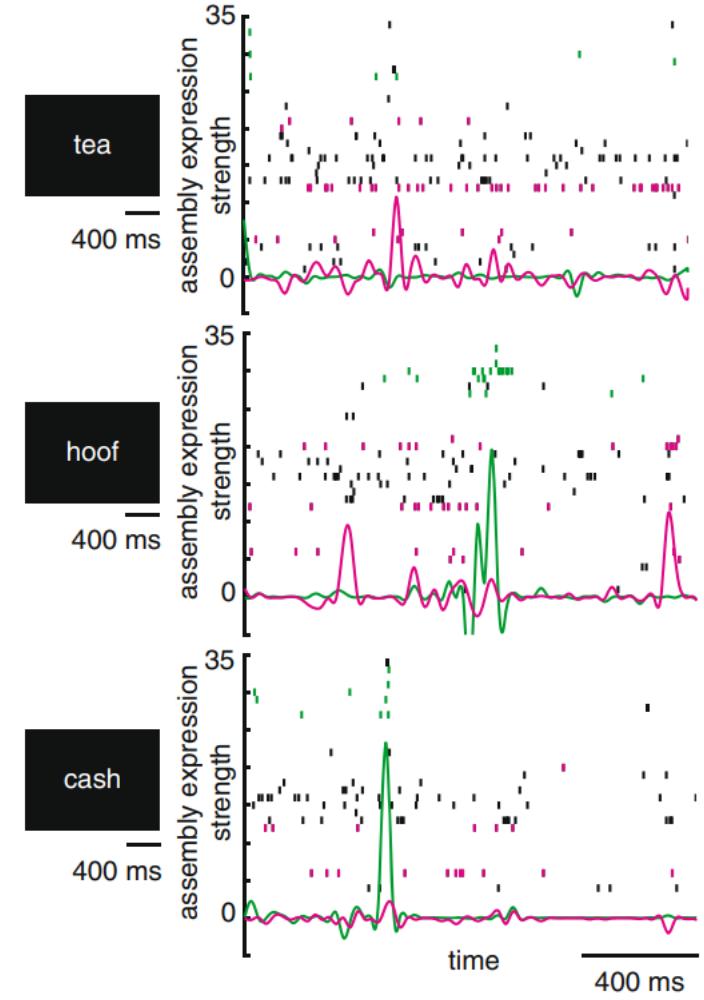
Examples in the literature



Peyrache et al., Nat Neuro 2009

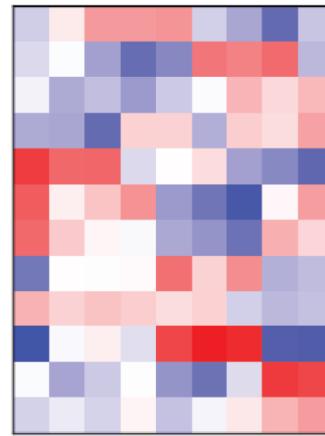


Oliva et al., Nature 2020



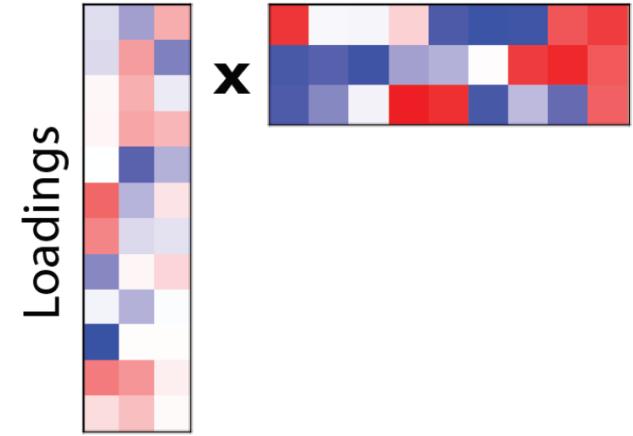
Umbach et al., Nat Comm 2022

Original Data

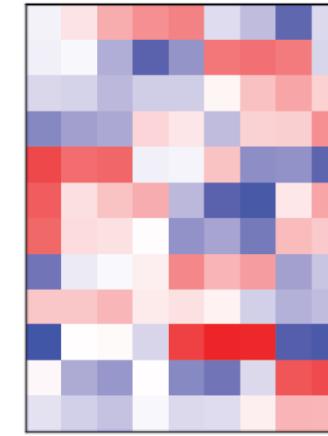


\approx

Components

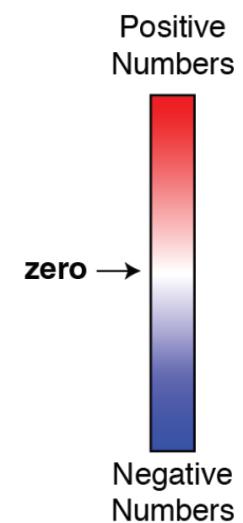
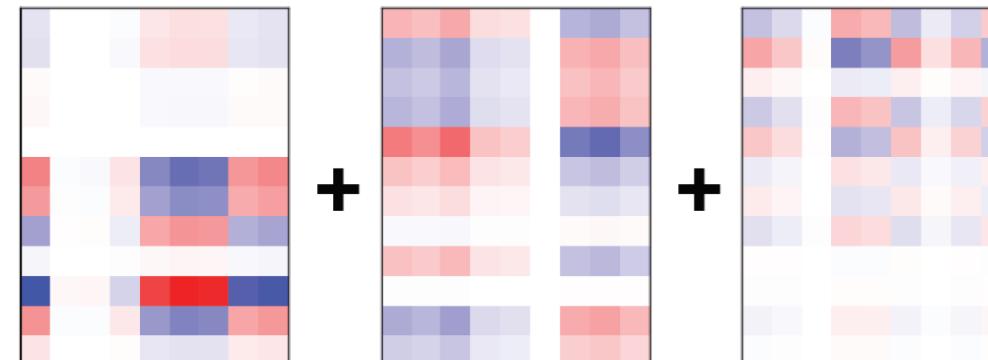


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax



neurosyntax

Session 4

Lecturer: Saman Abbaspoor

Principal Component Analysis (PCA)

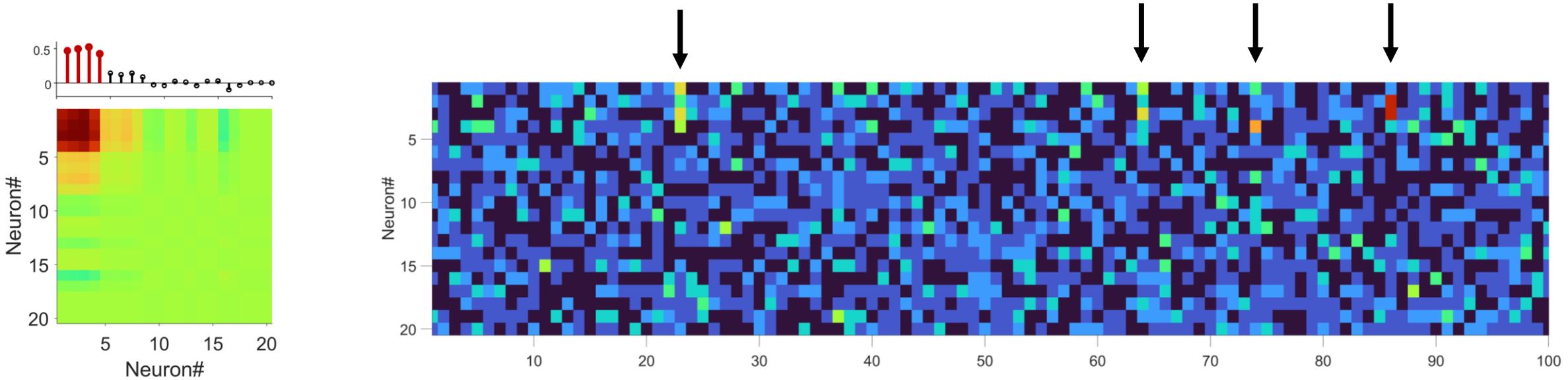
Computation

$$\begin{matrix} \text{Data transposed} \\ (\text{mean-subtracted}) \end{matrix} \times \begin{matrix} \text{Data} \\ (\text{mean-subtracted}) \end{matrix} = \begin{matrix} \text{Covariance} \\ \text{matrix} \end{matrix} = \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \\ \text{transposed} \end{matrix} \times \begin{matrix} \text{Eigenvalues} \\ (\lambda) \end{matrix} \times \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \end{matrix}$$

Dimensionality Reduction

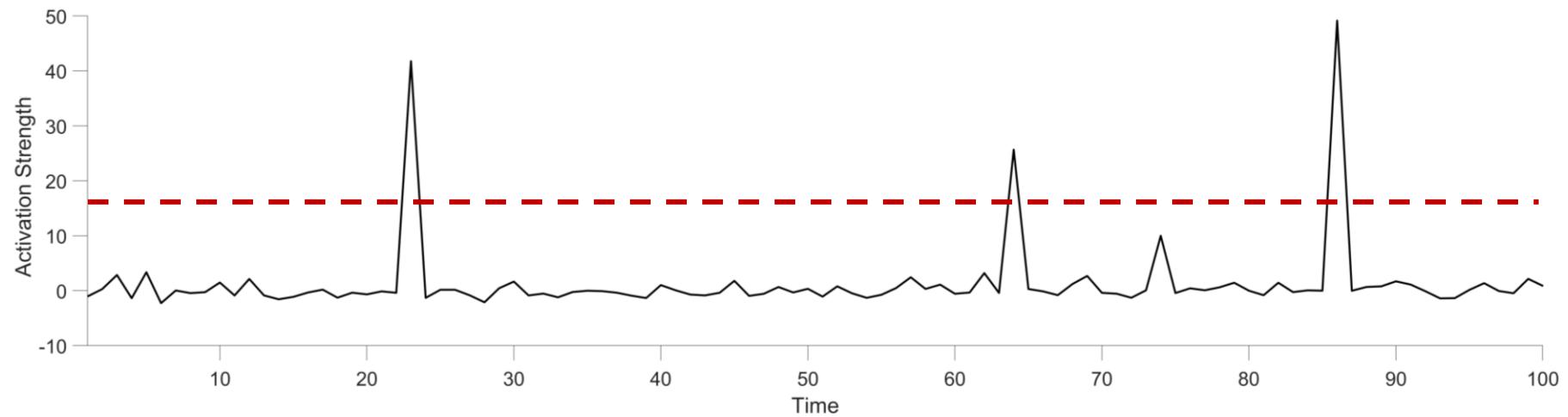
$$\begin{matrix} \text{Data} \\ (\text{mean-subtracted}) \end{matrix} \times \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \\ \text{transposed} \end{matrix} = \begin{matrix} \text{Scores} \end{matrix}$$

Interpreting loadings and scores



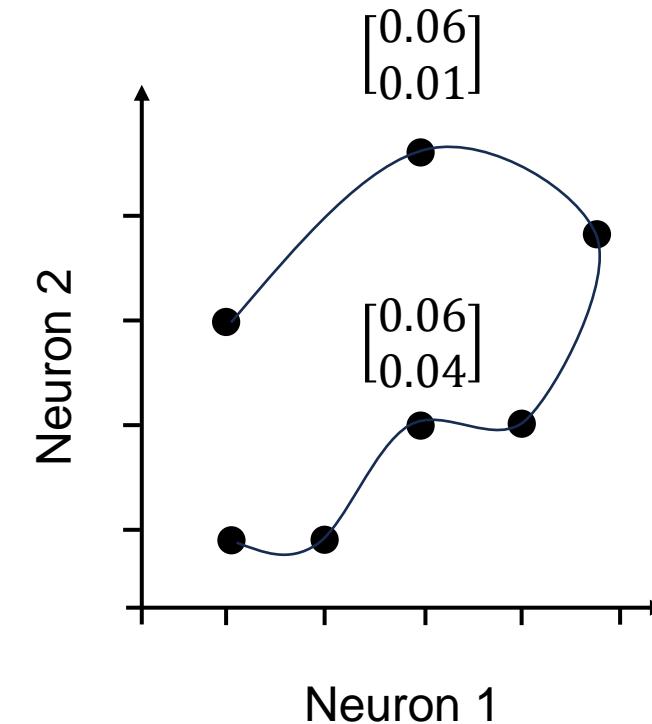
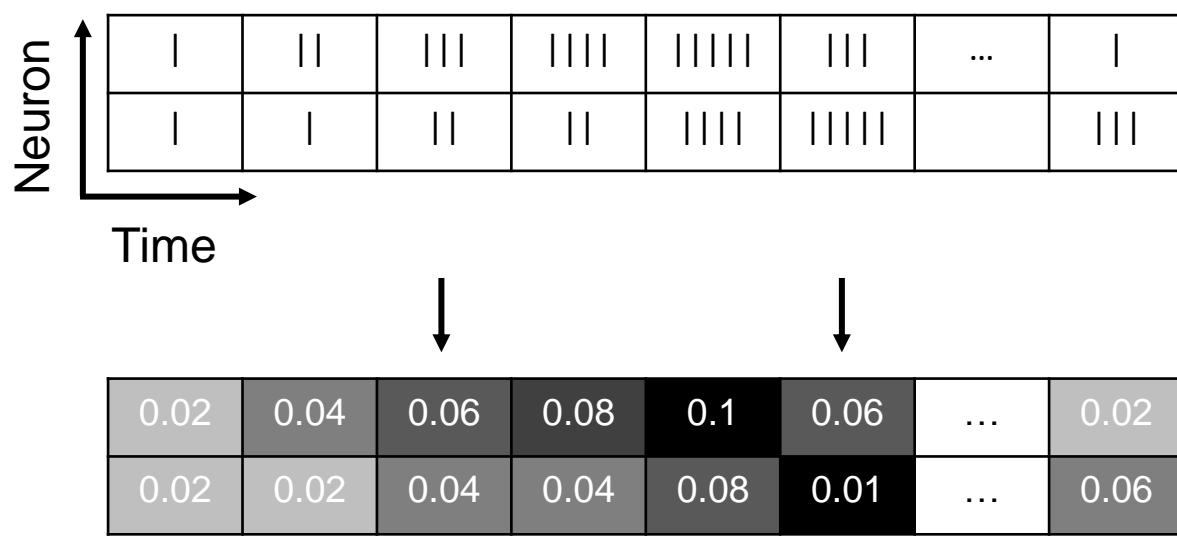
$$R_k(t) = z(t)^T P_k z(t)$$

Activation at each time point is the weighted average of firing of cells in the spike matrix with weights coming from each PCA/ICA component

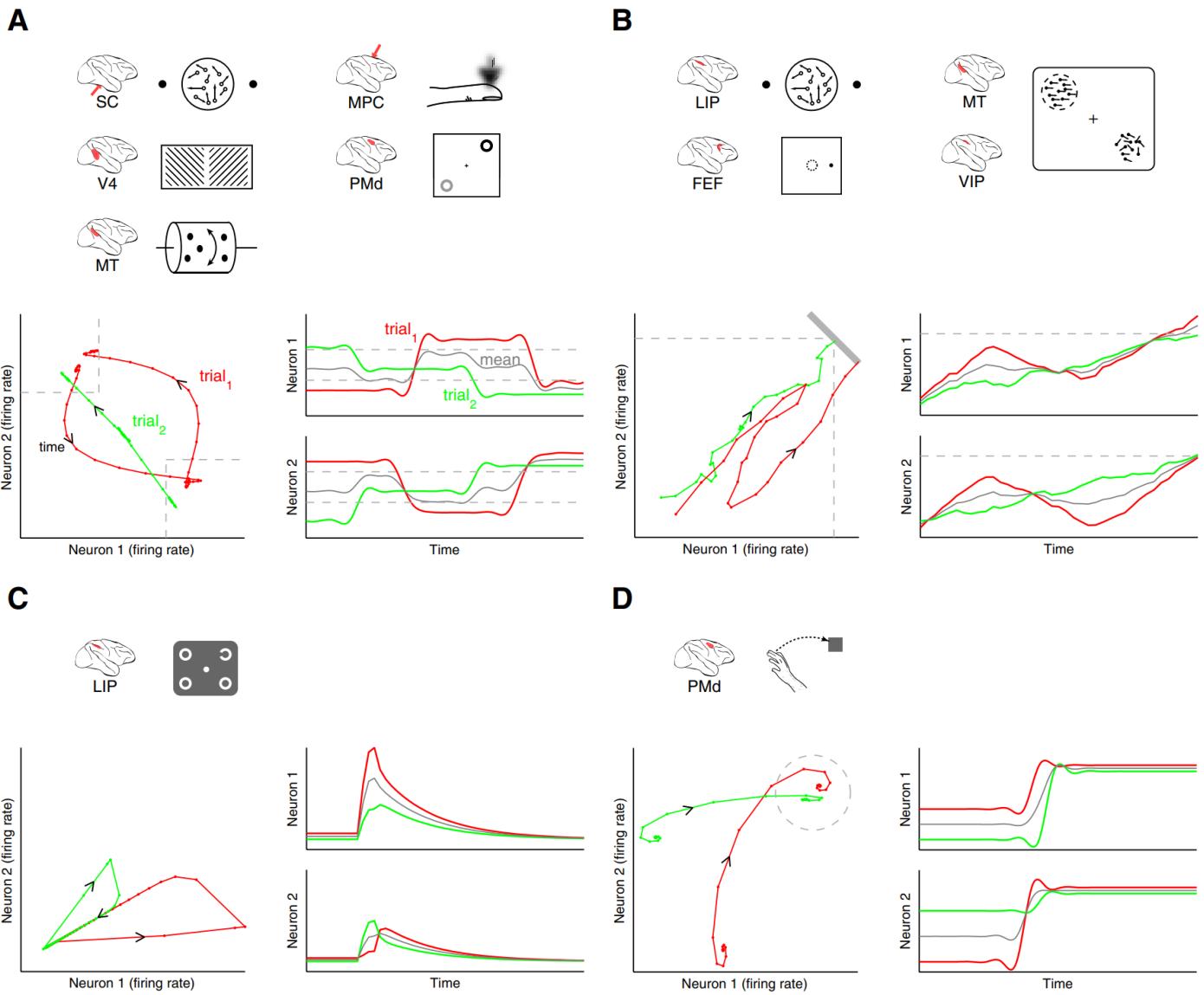


Neural Manifold

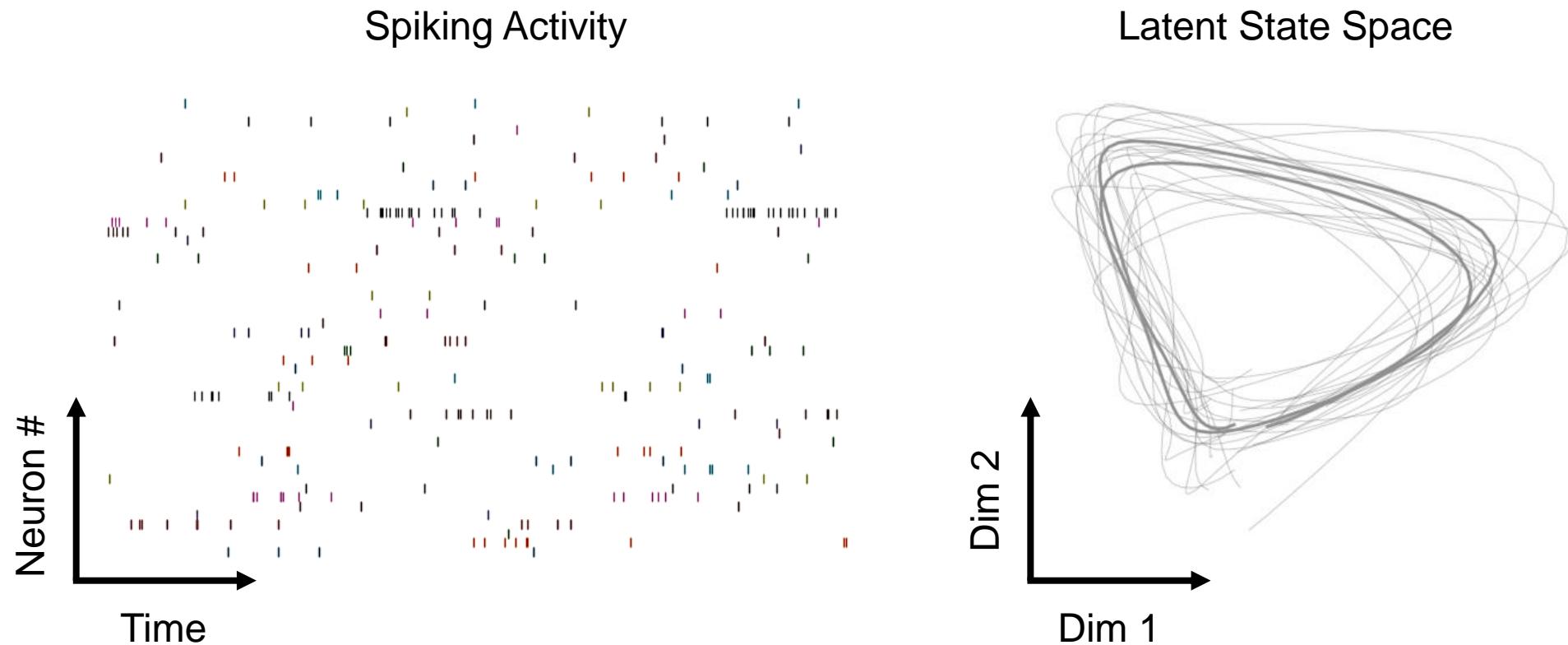
Neural trajectory of parallel spike trains



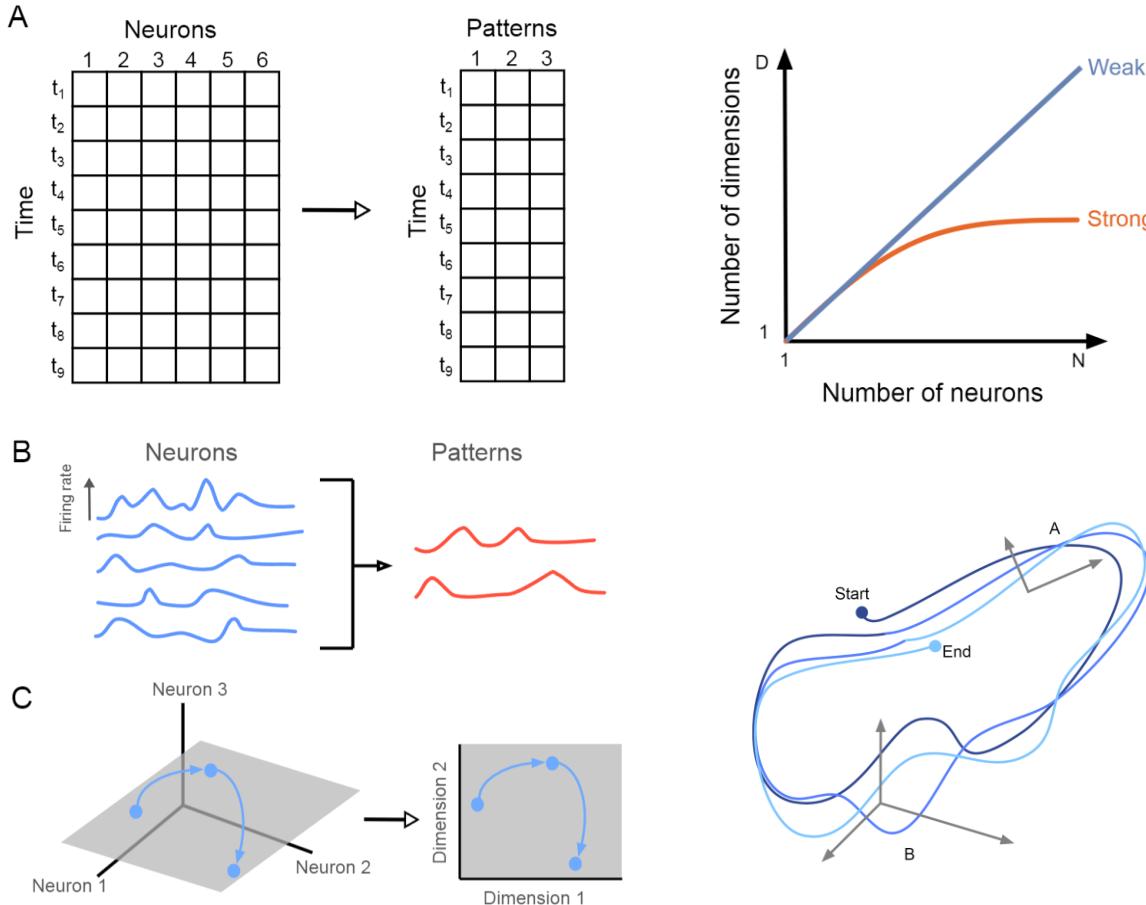
Neural trajectory of parallel spike trains



Dimensionality reduction for large-scale neural recordings

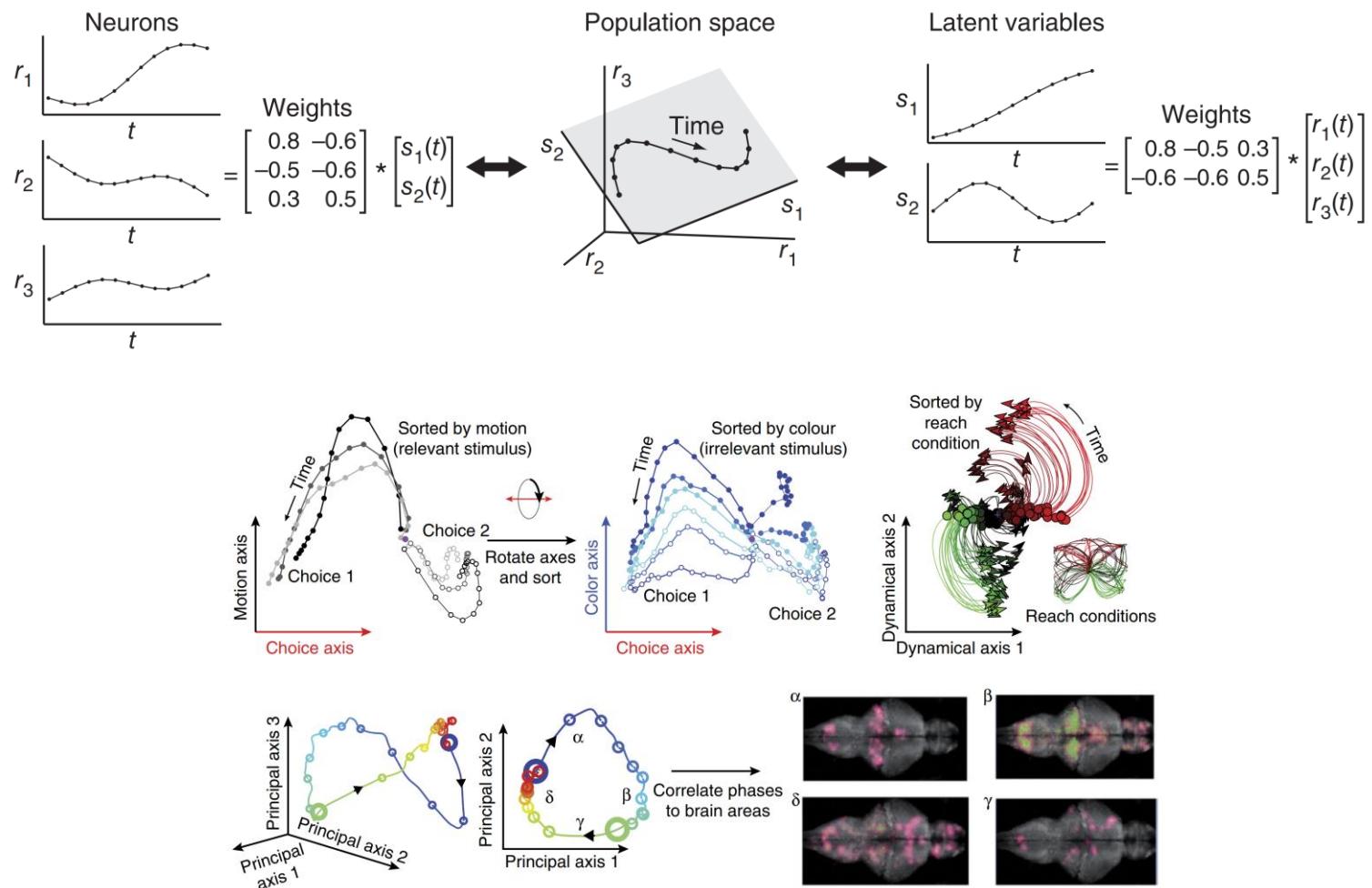
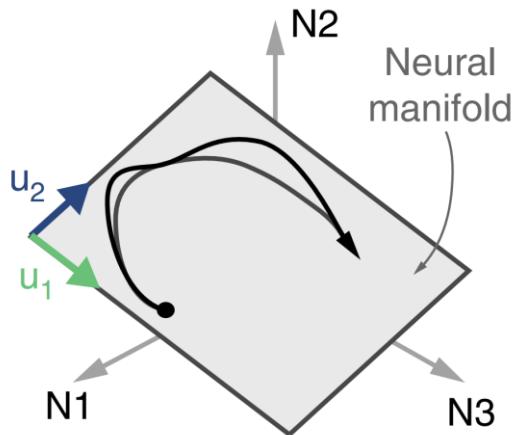
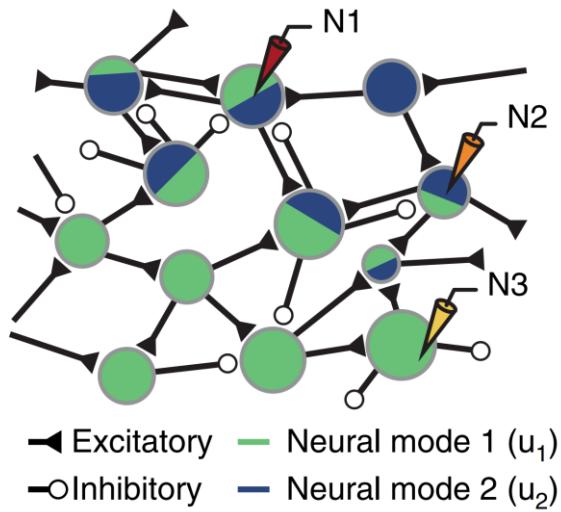


Strong and weak principles of neural dimension reduction



- The weak principle is that dimension reduction is a convenient tool for making sense of complex neural data.
- The strong principle is that dimension reduction shows us the true latent signal(s) encoded by a population of neurons and so moves us closer to how neural circuits actually operate and compute.

Constraints on dynamics in high-dimensional spaces

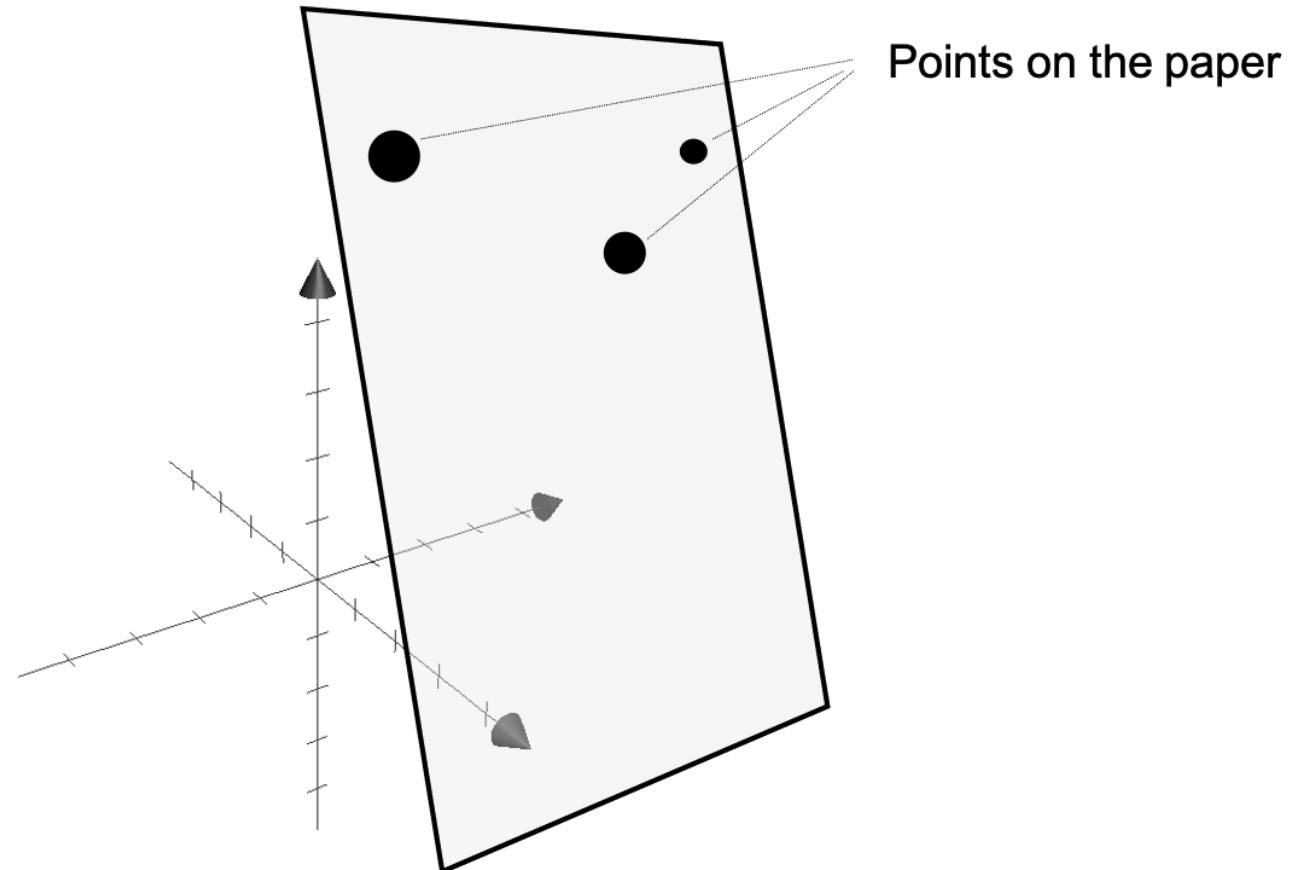


Manifold hypothesis

The manifold hypothesis is that real-world high dimensional data lie on low-dimensional manifolds embedded in the high-dimensional space. The main idea here is that even though our real-world data is high-dimensional, there is actually some lower-dimensional representation.

The **intrinsic dimensionality** of a space is the number of *required* pieces of information that we need to describe each object in the space. This may differ from the number of pieces of information that we *are* using, which we call the **extrinsic dimensionality** of the space.

A piece of paper in 3D space



TESTING THE MANIFOLD HYPOTHESIS

CHARLES FEFFERMAN, SANJOY MITTER, AND HARIHARAN NARAYANAN

ABSTRACT. The hypothesis that high dimensional data tend to lie in the vicinity of a low dimensional manifold is the basis of manifold learning. The goal of this paper is to develop an algorithm (with accompanying complexity guarantees) for testing the existence of a manifold that fits a probability distribution supported in a separable Hilbert space, only using i.i.d samples from that distribution. More precisely, our setting is the following. Suppose that data are drawn independently at random from a probability distribution \mathcal{P} supported on the unit ball of a separable Hilbert space \mathcal{H} . Let $\mathcal{G}(d, V, \tau)$ be the set of submanifolds of the unit ball of \mathcal{H} whose volume is at most V and reach (which is the supremum of all r such that any point at a distance less than r has a unique nearest point on the manifold) is at least τ . Let $\mathcal{L}(\mathcal{M}, \mathcal{P})$ denote mean-squared distance of a random point from the probability distribution \mathcal{P} to \mathcal{M} . We obtain an algorithm that tests the manifold hypothesis in the following sense.

The algorithm takes i.i.d random samples from \mathcal{P} as input, and determines which of the following two is true (at least one must be):

- (1) There exists $\mathcal{M} \in \mathcal{G}(d, CV, \frac{\tau}{C})$ such that $\mathcal{L}(\mathcal{M}, \mathcal{P}) \leq C\epsilon$.
- (2) There exists no $\mathcal{M} \in \mathcal{G}(d, V/C, C\tau)$ such that $\mathcal{L}(\mathcal{M}, \mathcal{P}) \leq \frac{\epsilon}{C}$.

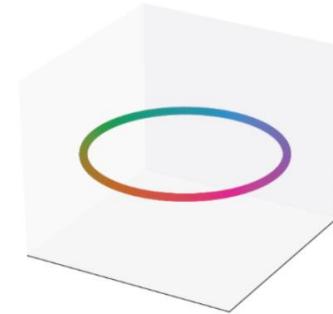
The answer is correct with probability at least $1 - \delta$.

Examining intrinsic and embedding dimensionality

The intrinsic dimension is the minimal number of continuous variables needed to parametrize the manifold. In contrast, the embedding dimension is a measure of the number of dimensions explored by the manifold within the ambient Euclidean space.

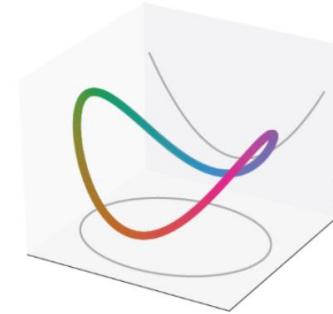
Intrinsic Dimension

1



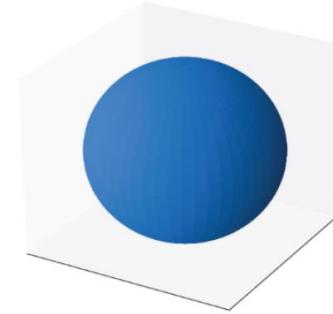
Embedding Dimension

2



>2

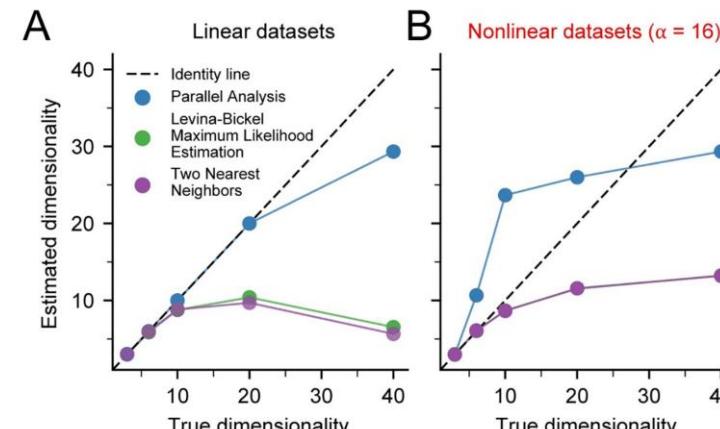
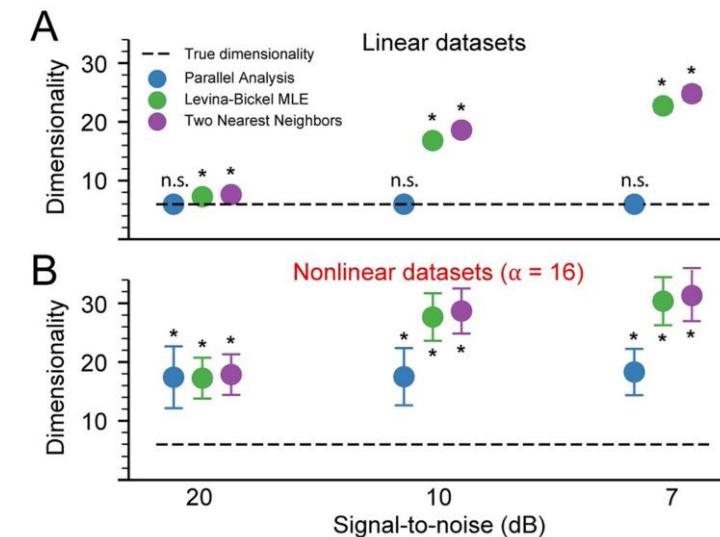
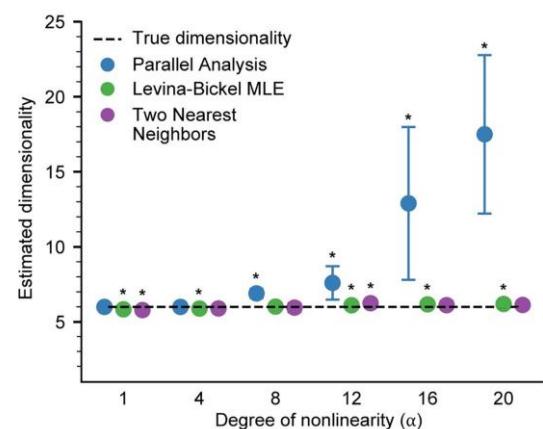
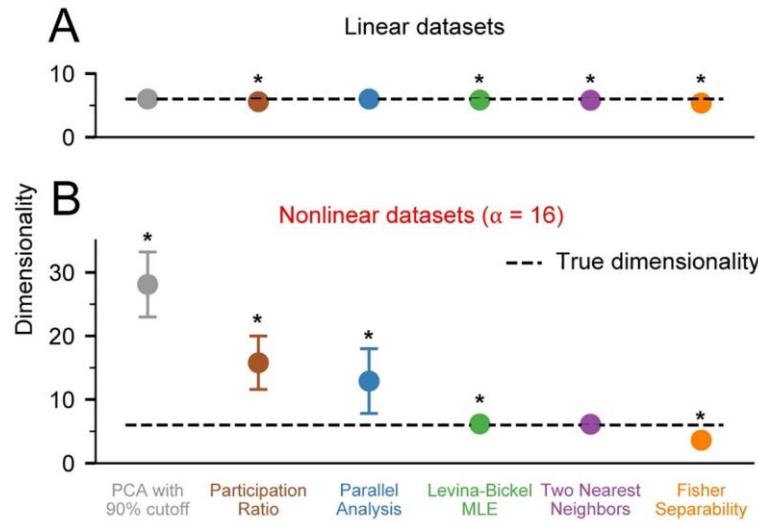
1



3

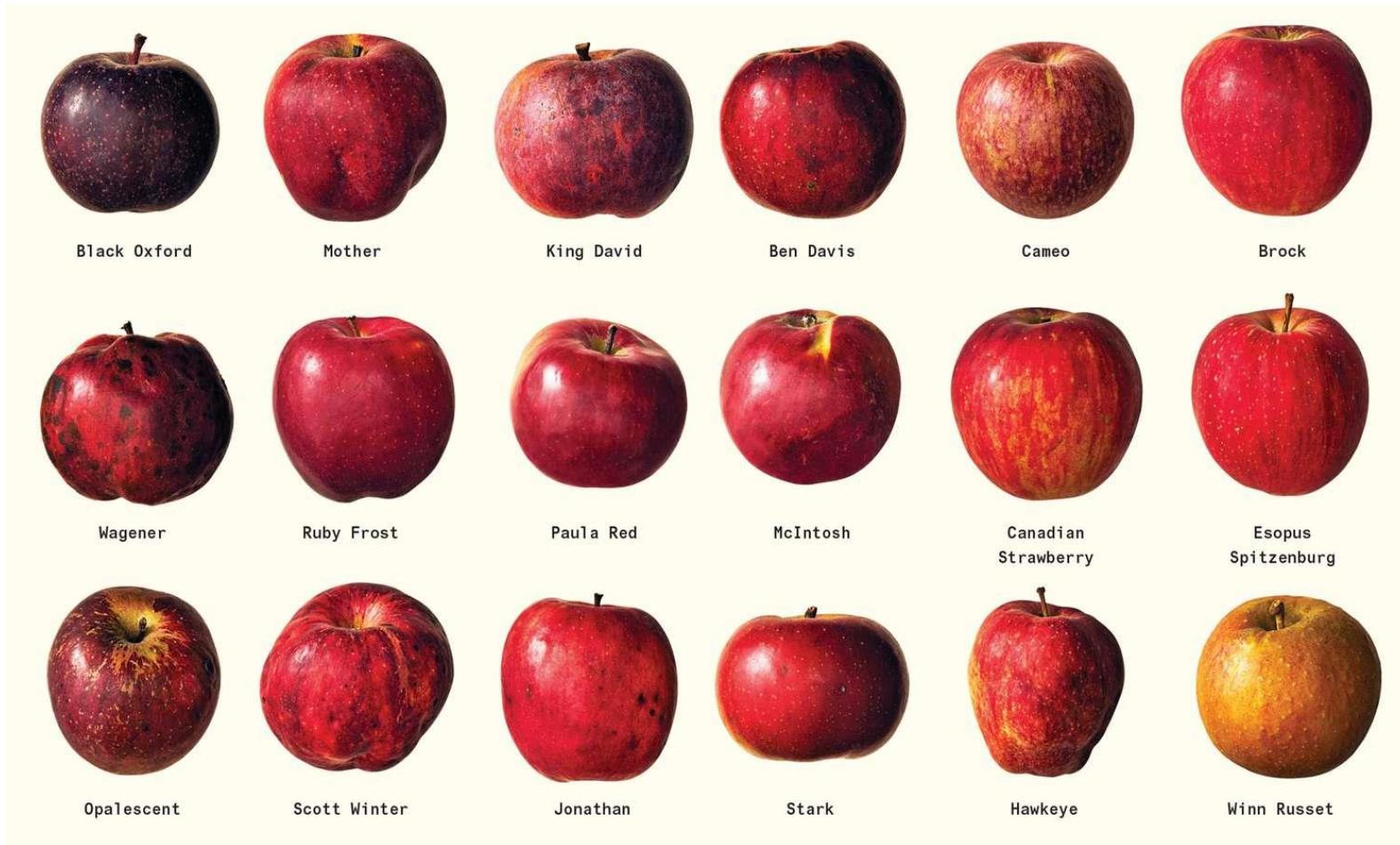
2

Intrinsic Dimension Estimation



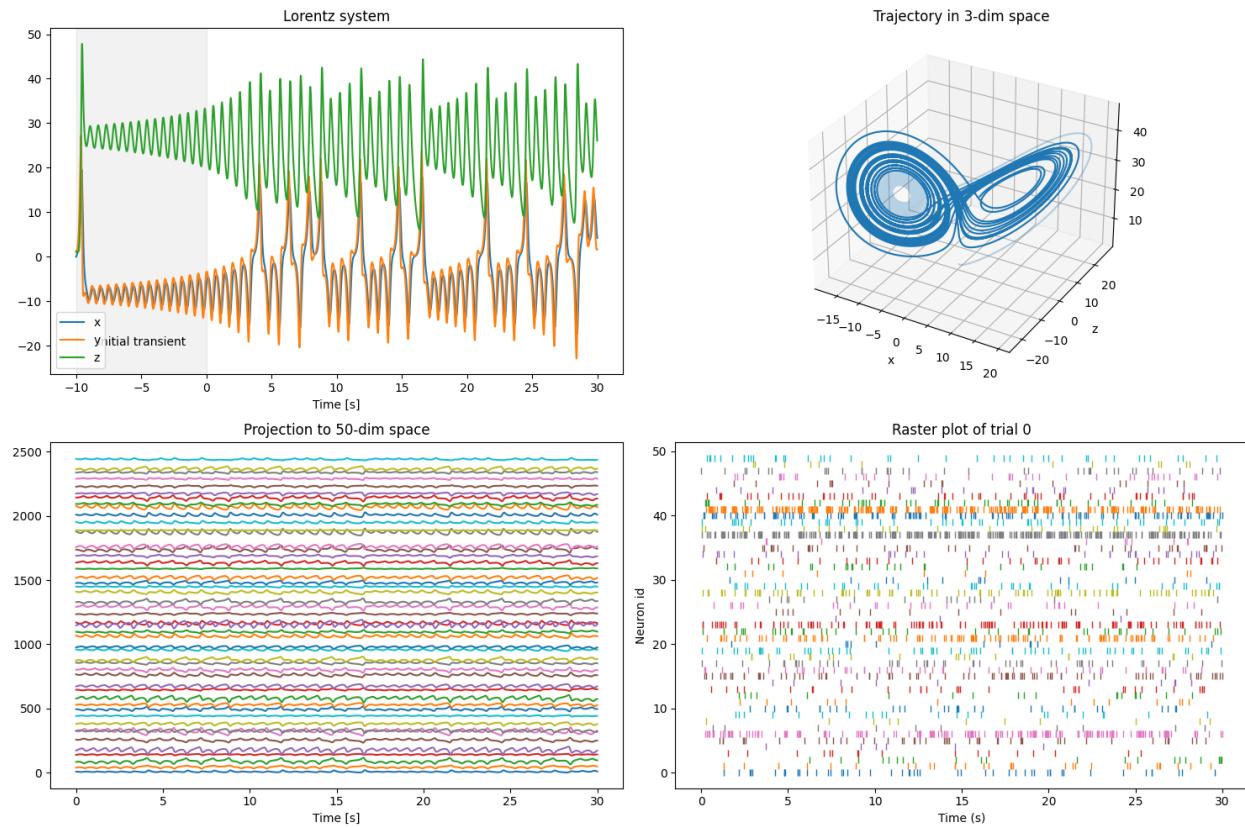
Recognizing latent structures in the environment

In statistics, latent variables are variables that can only be inferred indirectly through a mathematical model from other observable variables that can be directly observed or measured.



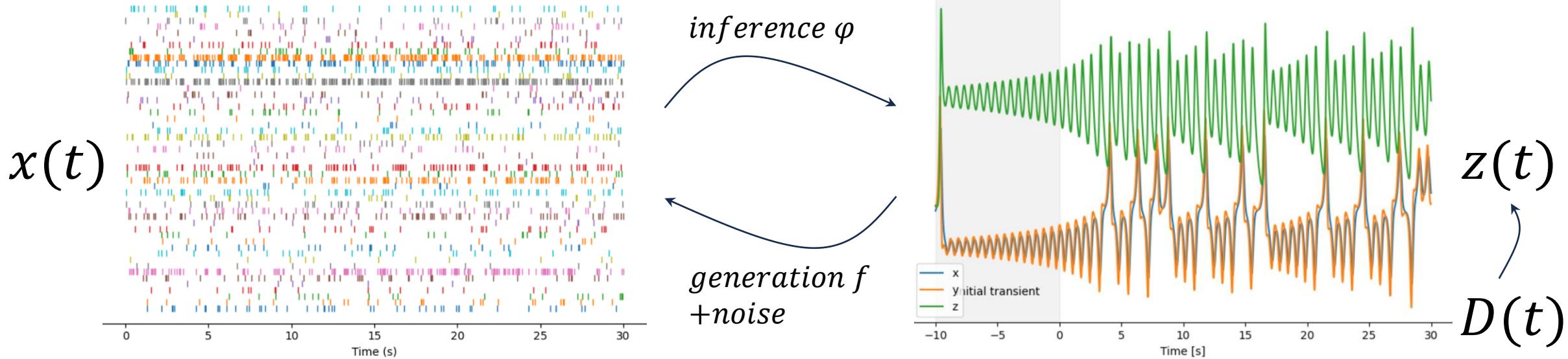
Assumption: Latent state generative model of neural population

Task: Inferring Latent Dynamics Underlying Neural Population Activity



Assumption: Latent state generative model of neural population

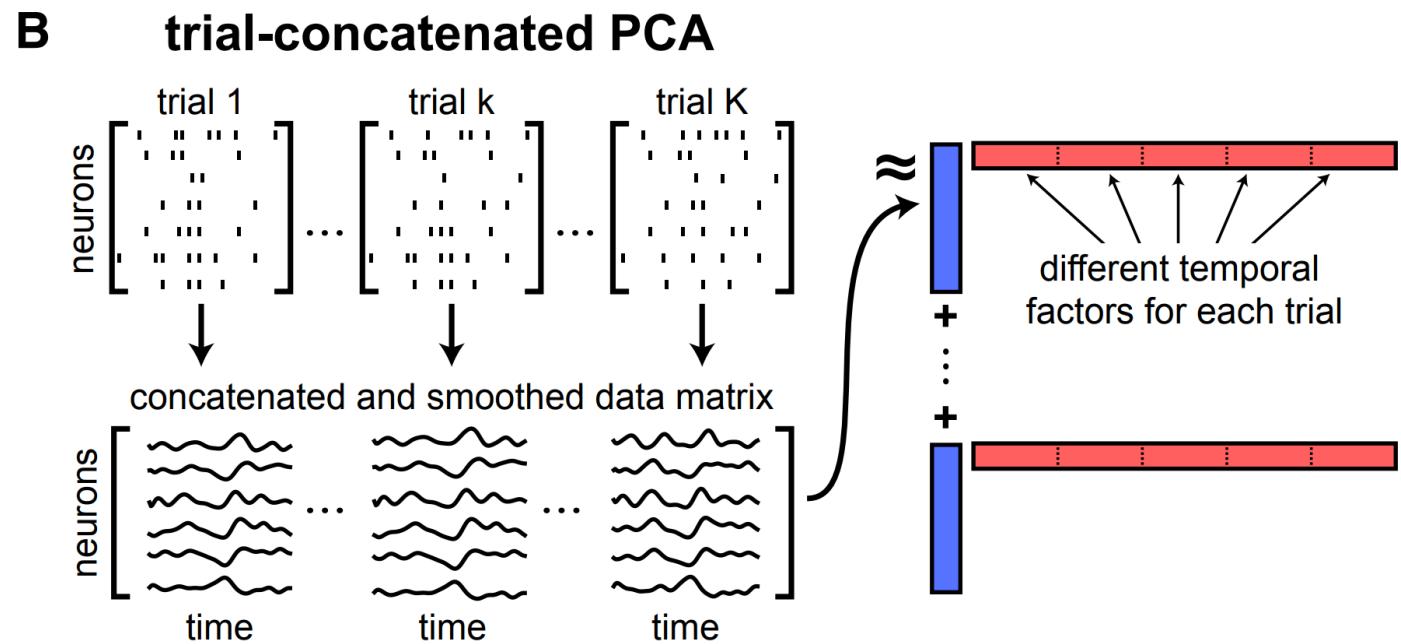
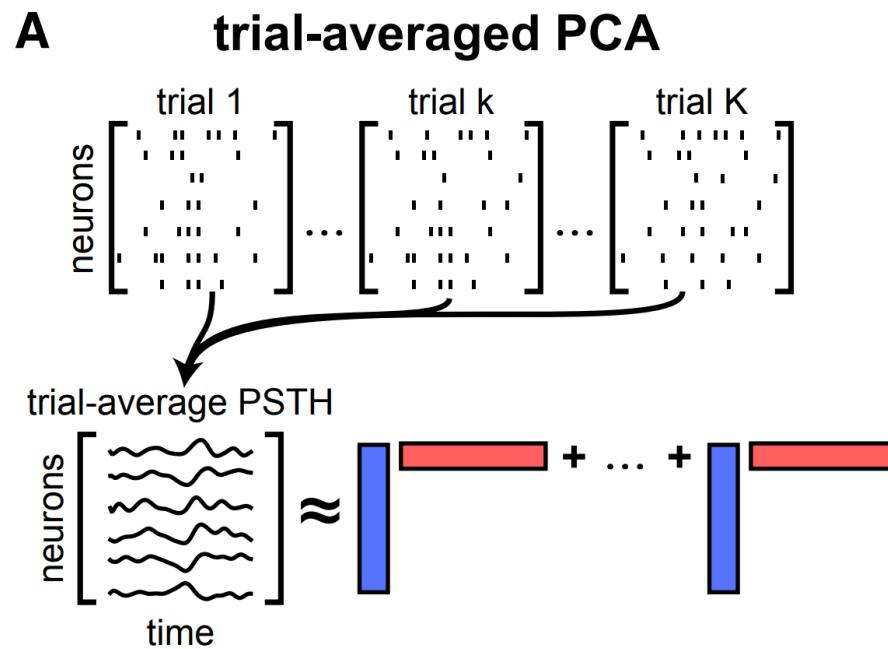
Task: Inferring Latent Dynamics Underlying Neural Population Activity



The strong principle is that dimension reduction shows us the true underlying signal embodied by the neural circuit, the so-called "latent signal." It is a theory that the brain really is low-dimensional compared to the number of neurons. The joint activity of a population of neurons is a (noisy) realization of this low-dimensional system—a realization using many more elements (neurons) than dimensions in the system.

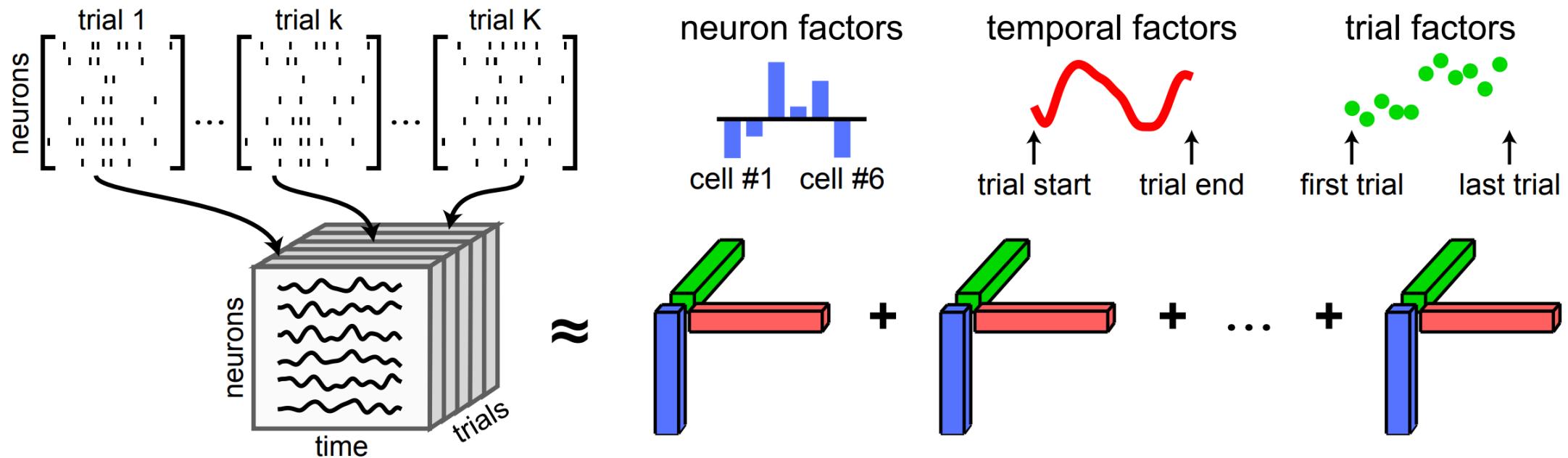


Matrix Representation of Trial-Structured Neural Data



Tensor Representation of Trial-Structured Neural Data

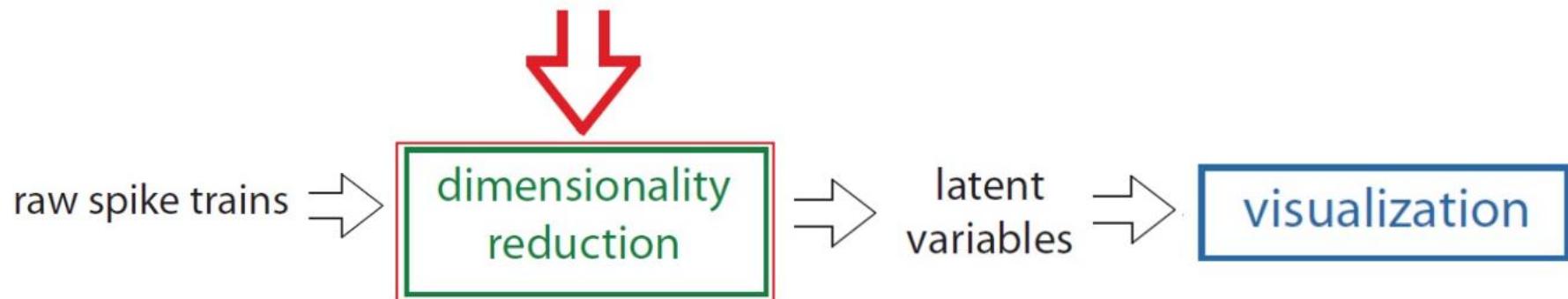
Tensor Components Analysis (TCA)



Unsupervised Discovery of Demixed, LowDimensional Neural Dynamics across Multiple
Timescales through Tensor Component Analysis, Williams, 2018



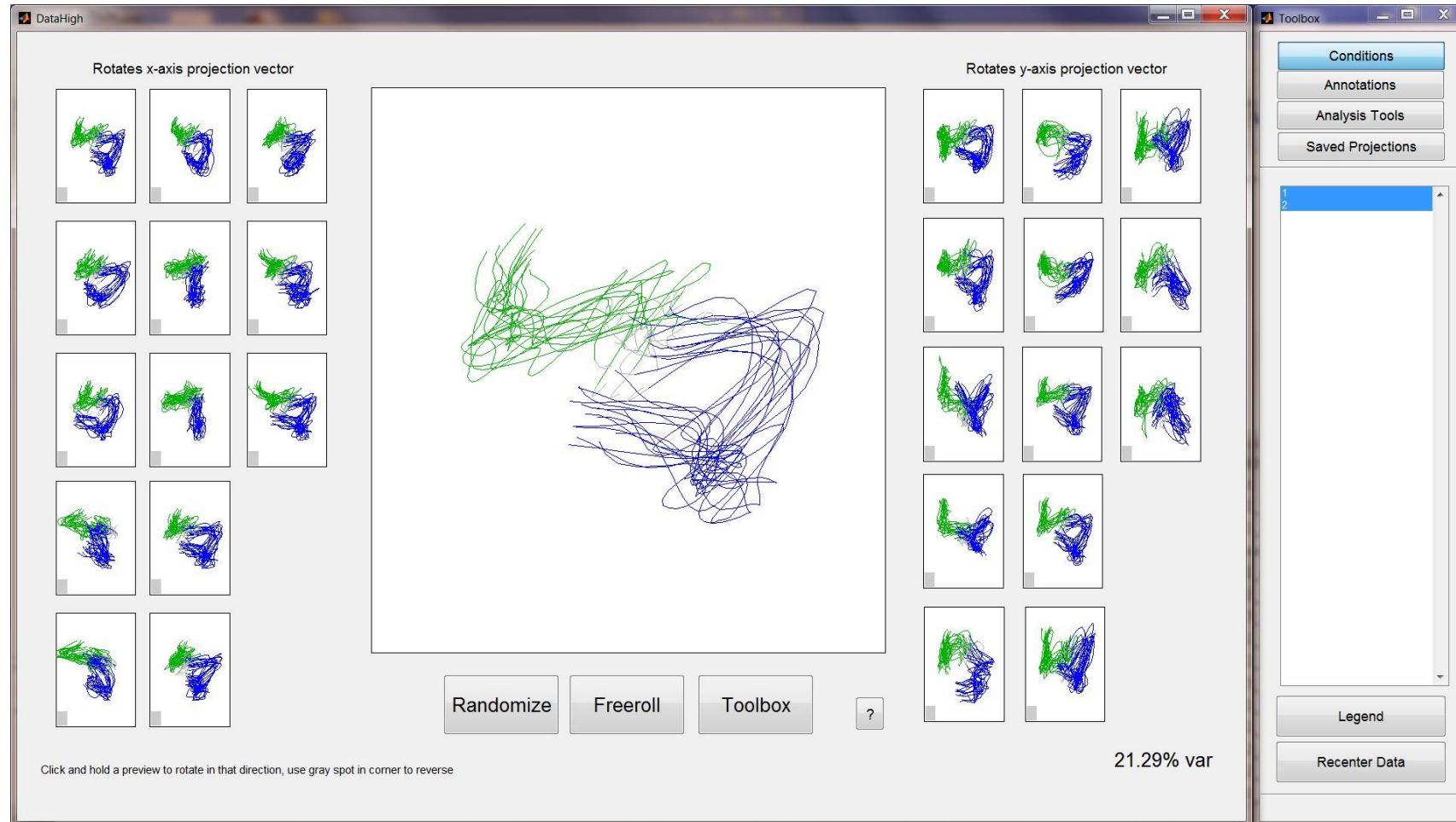
DataHigh



<https://users.ece.cmu.edu/~byronyu/software/DataHigh/datahigh.html>

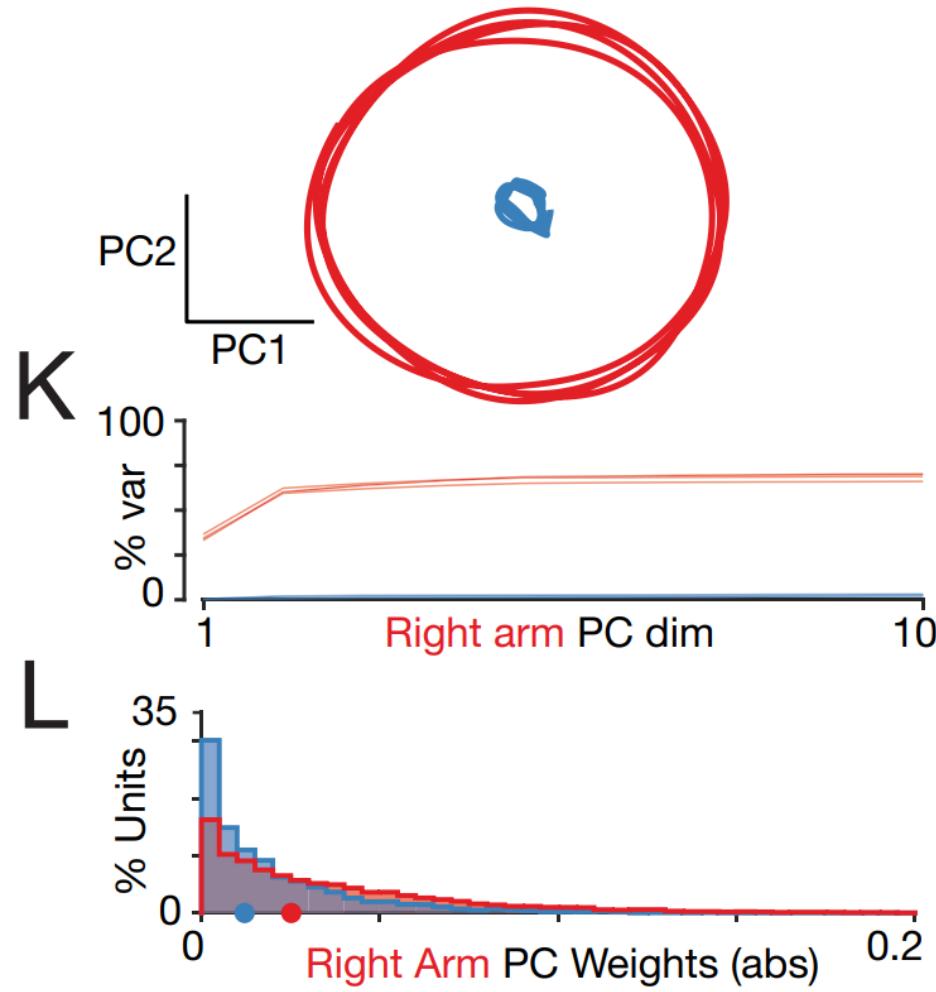
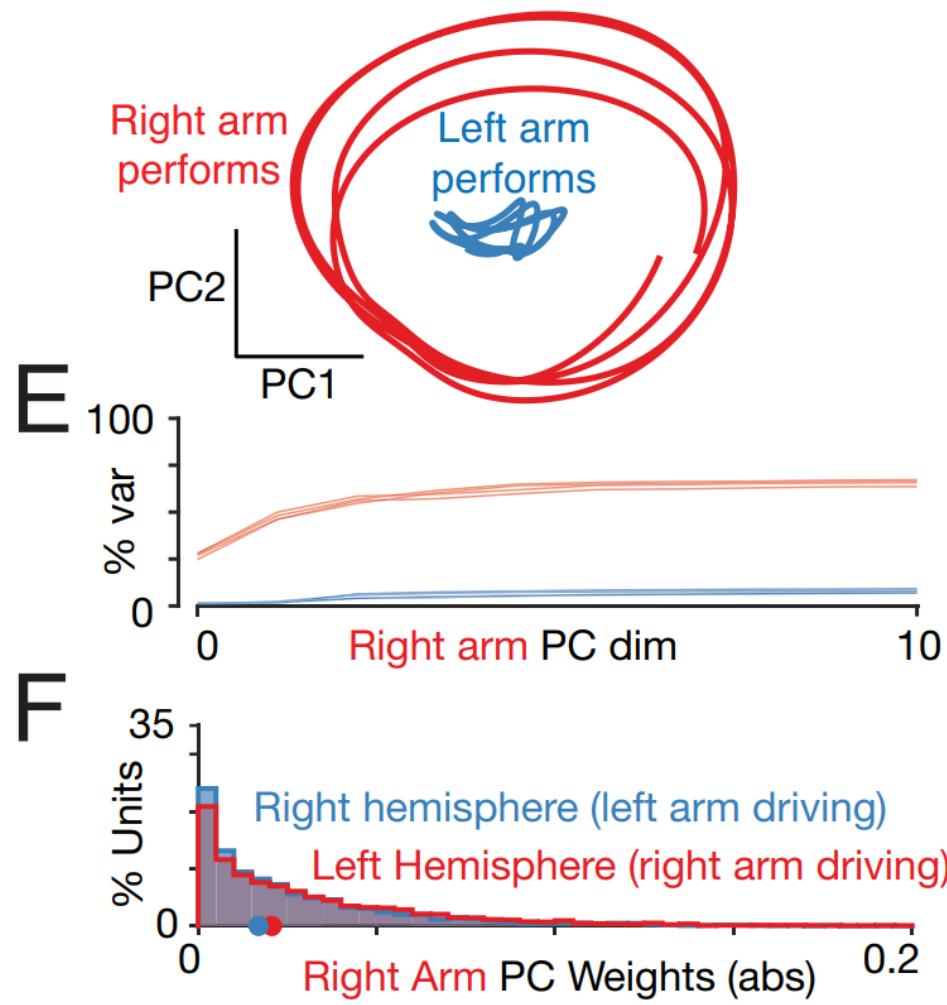
Data High

DataHigh is a Matlab-based graphical user interface to visualize and interact with high-dimensional neural population activity.



<https://users.ece.cmu.edu/~byronyu/software/DataHigh/datahigh.html>





PCA Assumptions

While PCA itself does not make assumptions about the data, the most popular interpretation of PCA does.

This classic interpretation posits that patterns in the loadings or scores indicate some latent underlying patterns in the data. This interpretation relies on four assumptions and may be misleading if those assumptions are violated.

First, the underlying patterns must exist and be independent of one another.

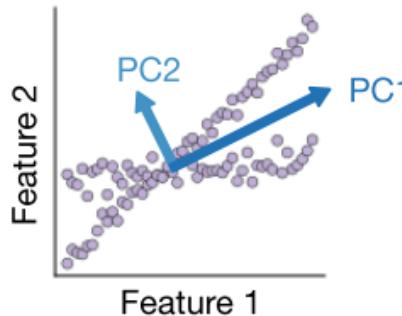
Second, they must combine linearly to form the observed data.

Third, the data must contain exclusively these patterns and additive, uncorrelated noise.

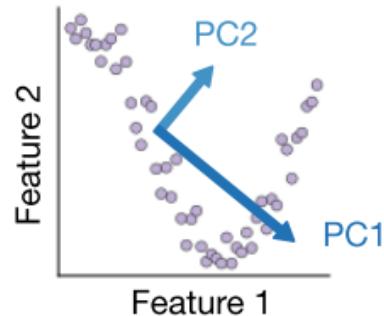
Fourth, observations must be independent. When any of these does not hold, the above interpretation of PCA may not hold.

Biases in PCA

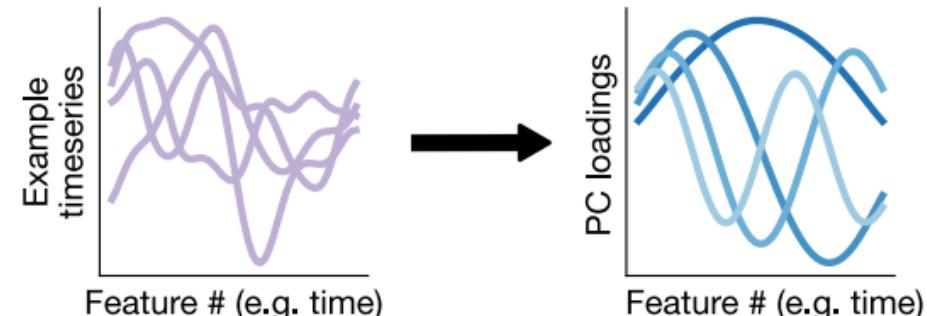
d Non-orthogonal factors



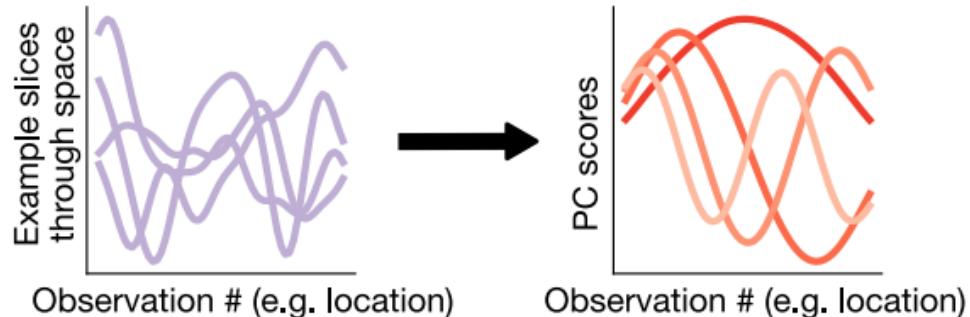
e Non-linearity



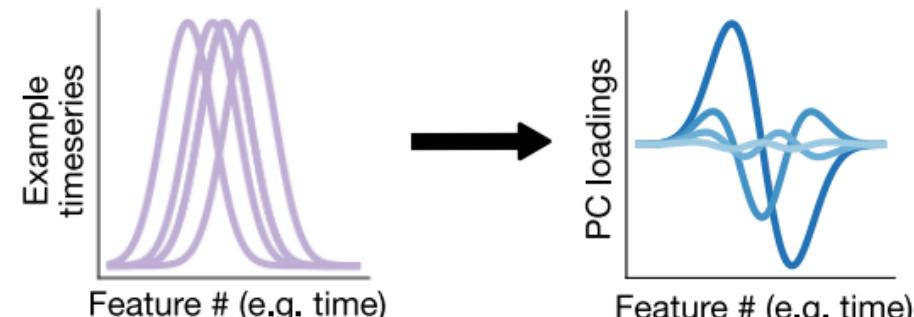
f Feature smoothness (e.g. temporal autocorrelation)



g Observation smoothness (e.g. spatial autocorrelation)

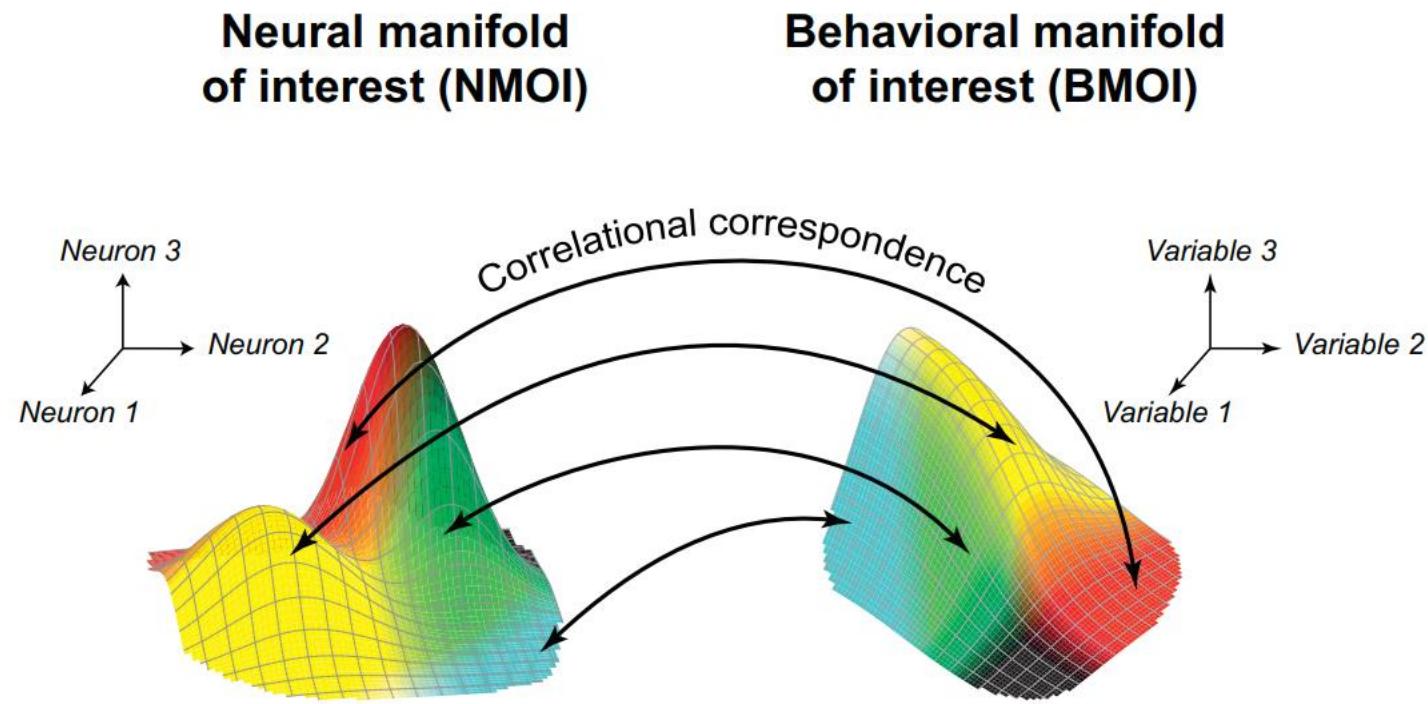


h Shifted or misaligned features

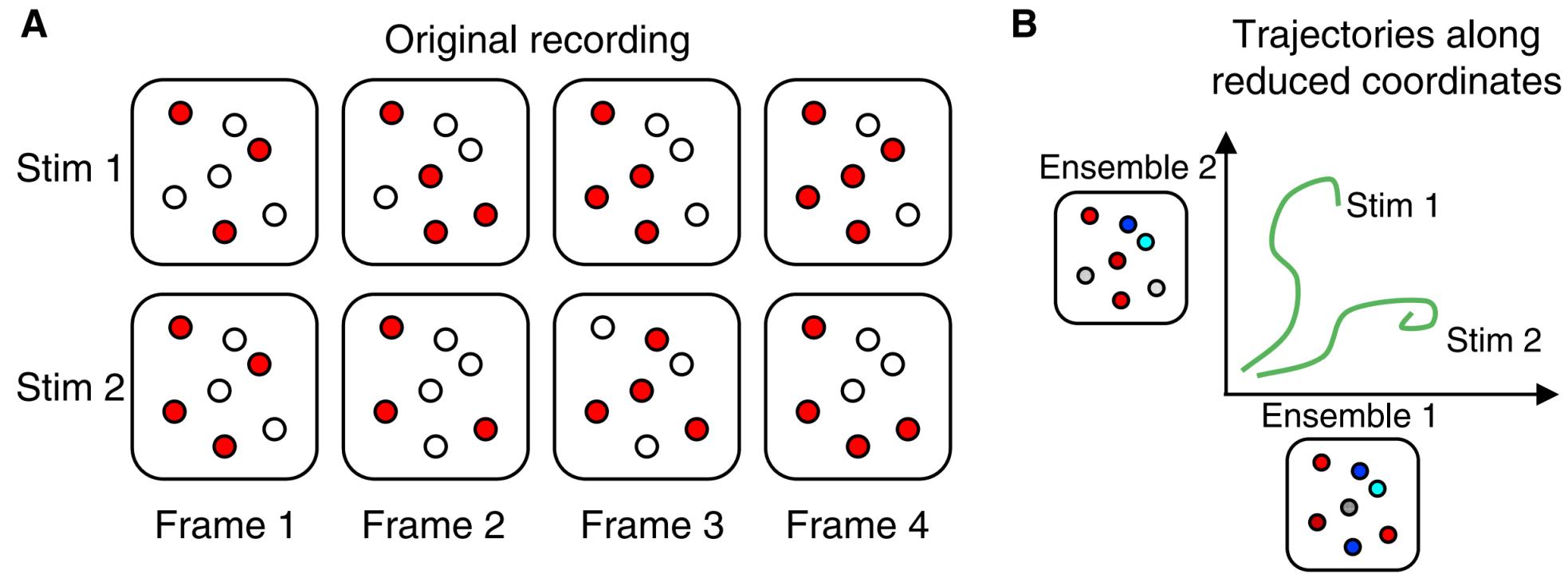


Component Analysis, Manifold Learning, State-space models, and Applications

Correlational Correspondence between Intrinsic Neural Manifold and the Manifold of the Measured Behavioral Variables



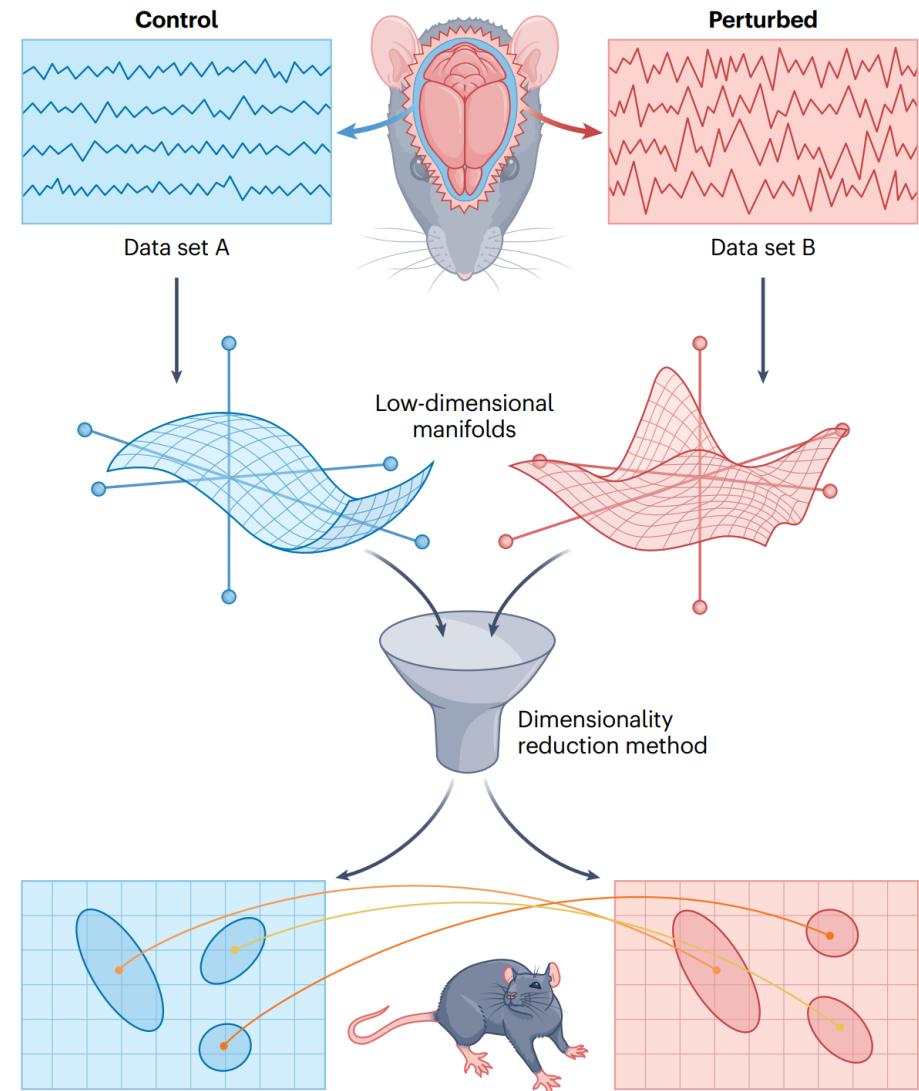
Signal components and their behavioral correlates



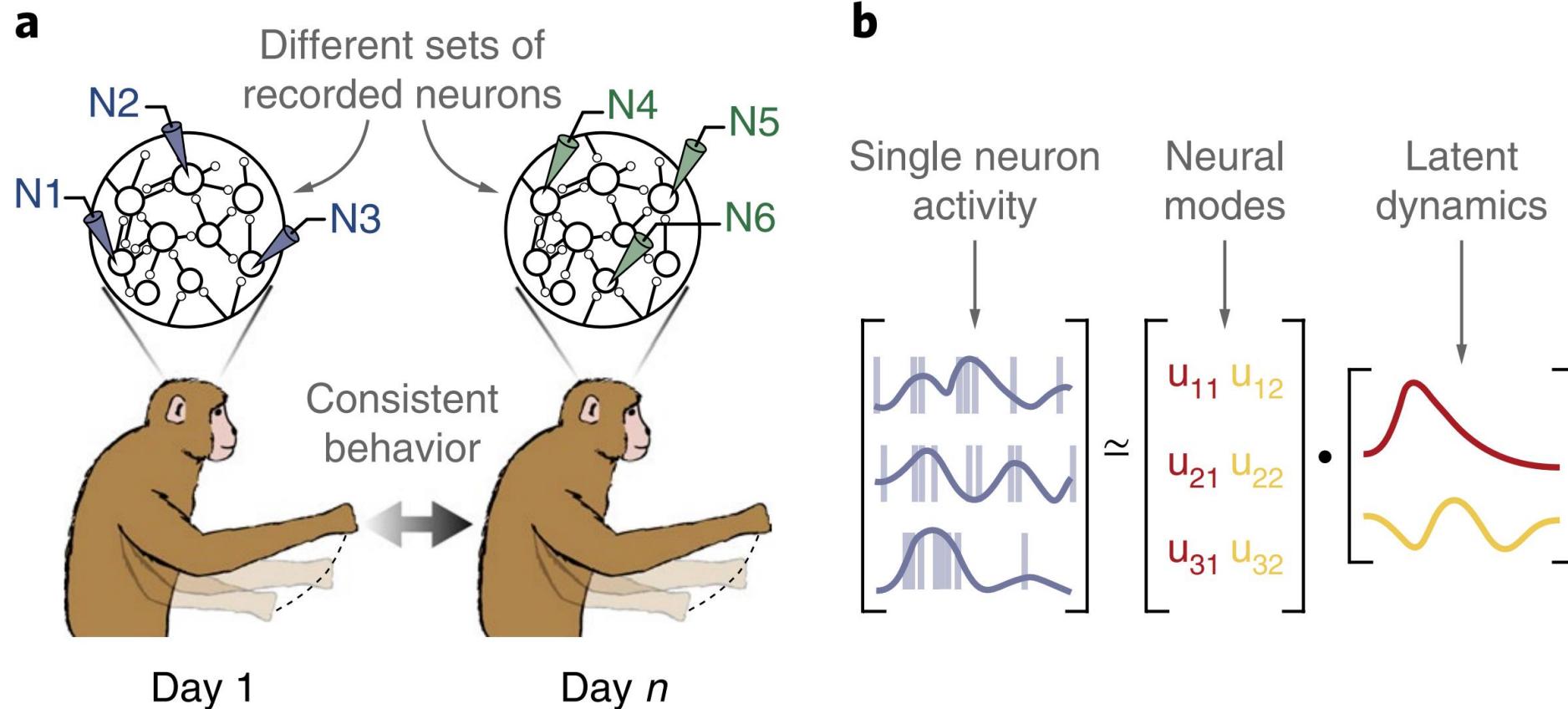
Current Biology

Comparisons of neural recordings across time, across subsets of neurons and across individuals requires the alignment of low-dimensional latent representations

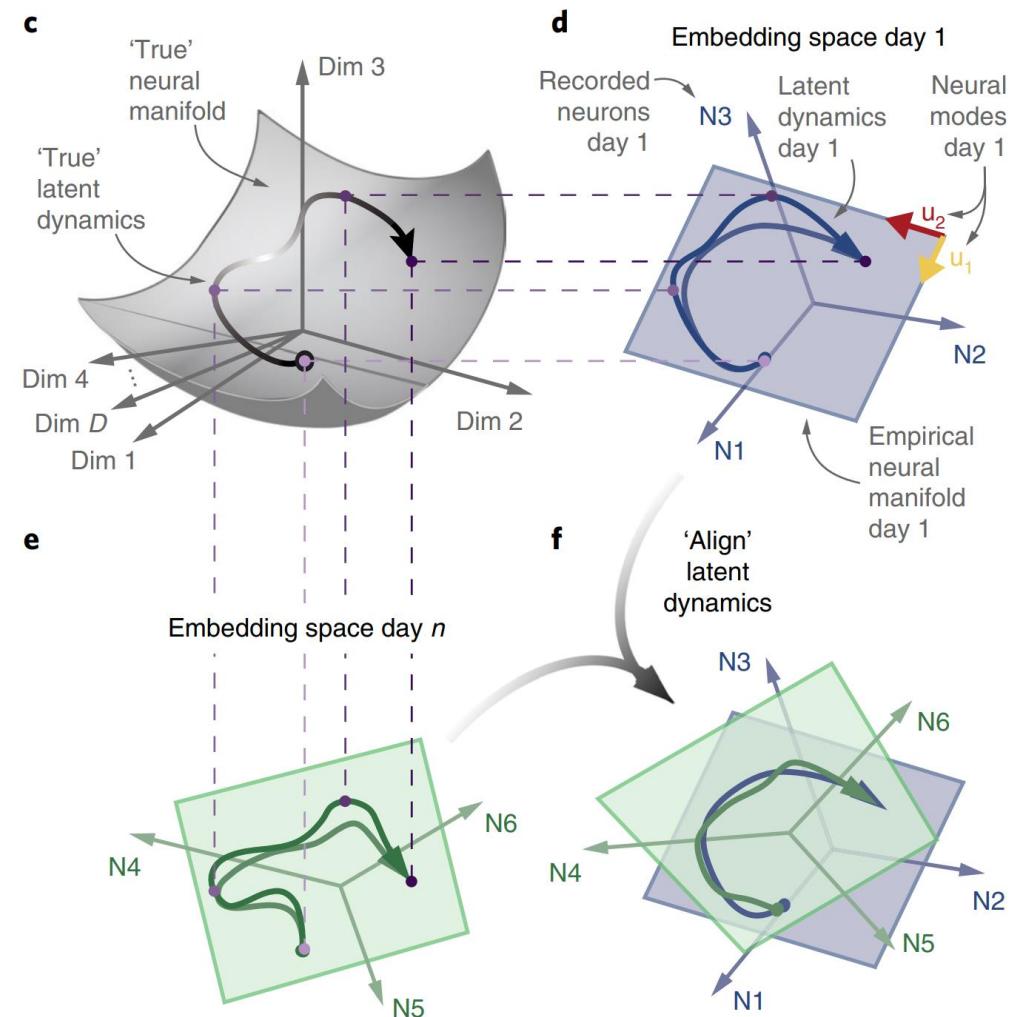
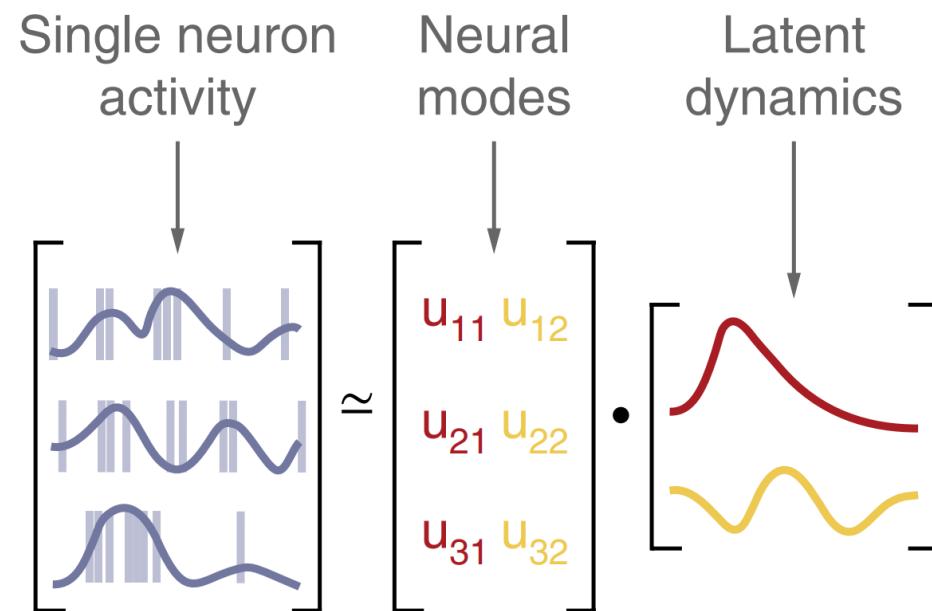
Dabagia et al., nature biomedical engineering 2023



Signal components and their behavioral correlates

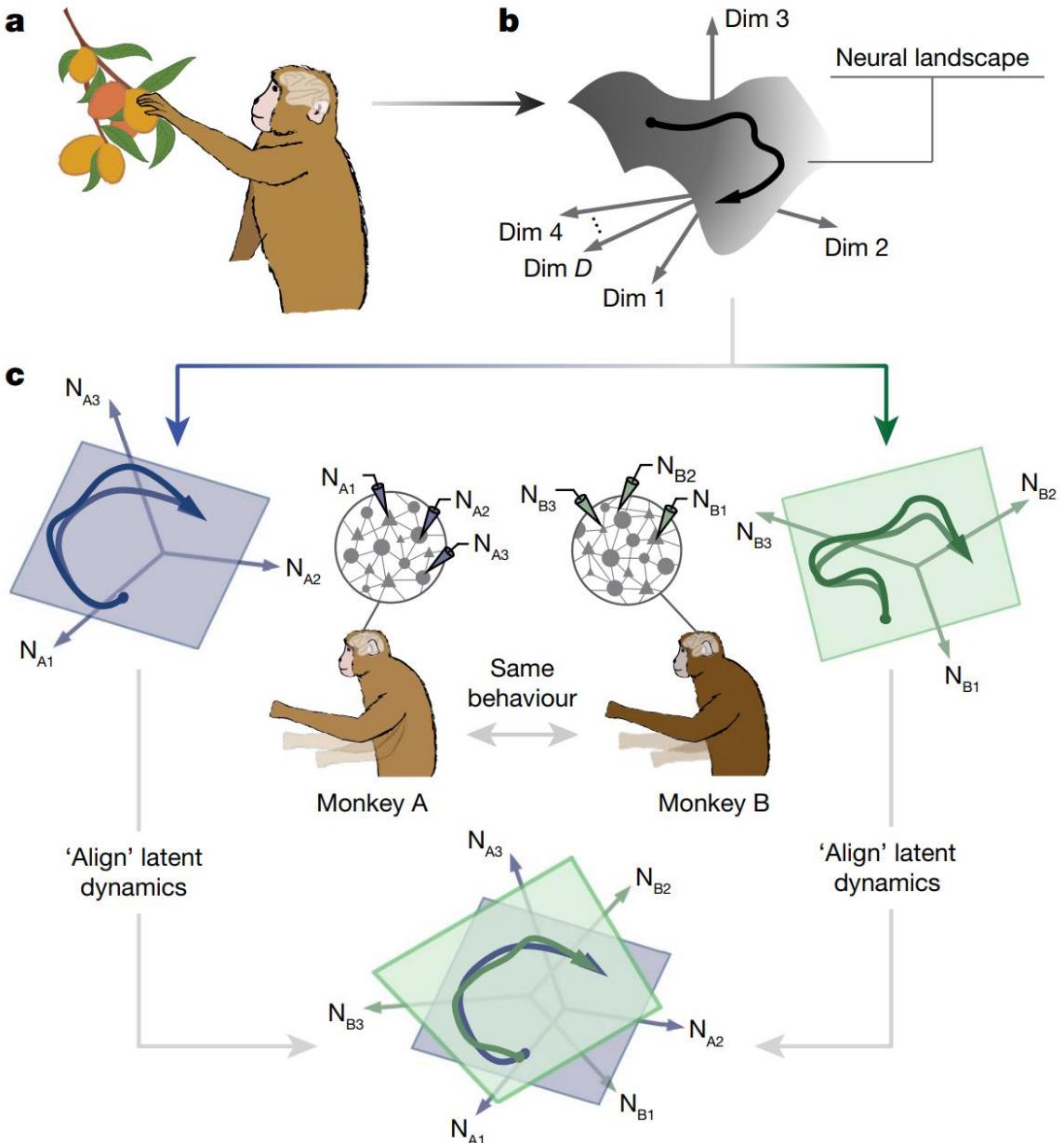


Signal components and their behavioral correlates

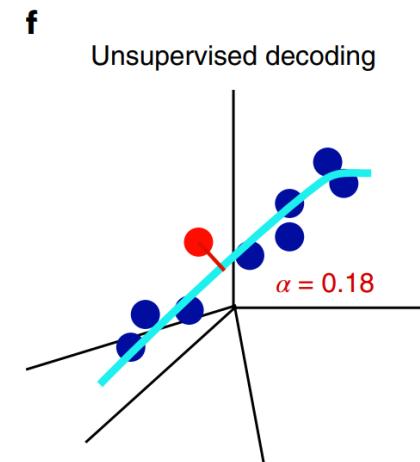
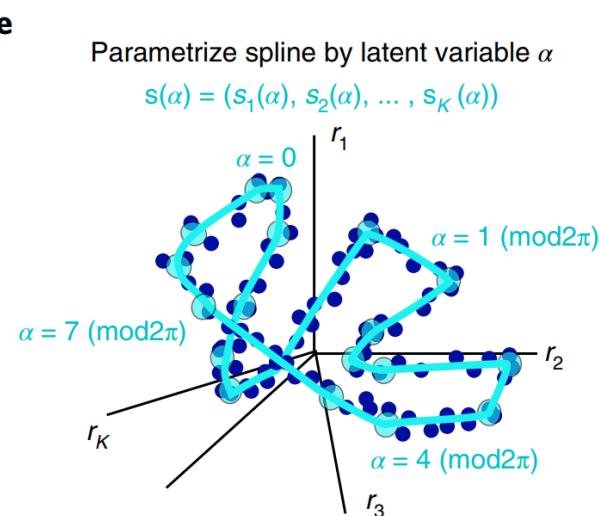
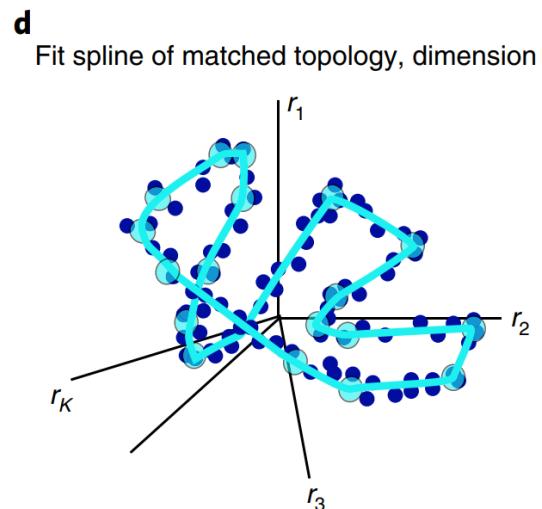
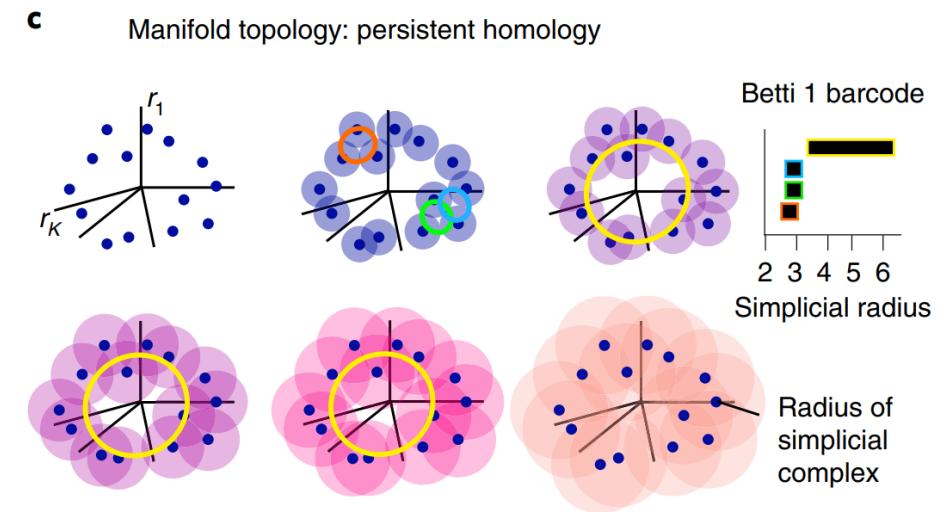
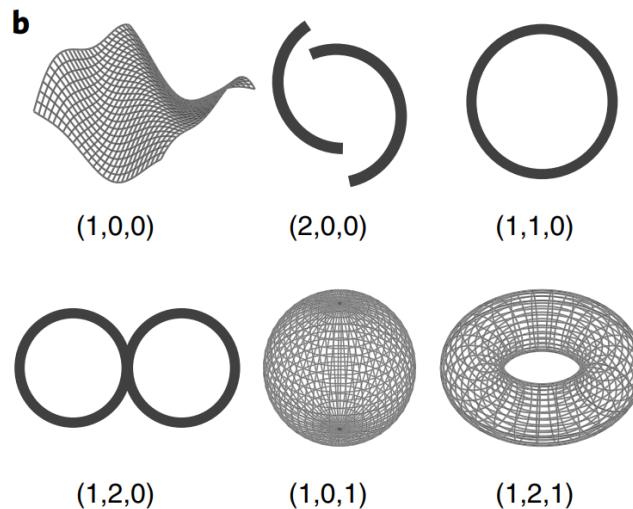
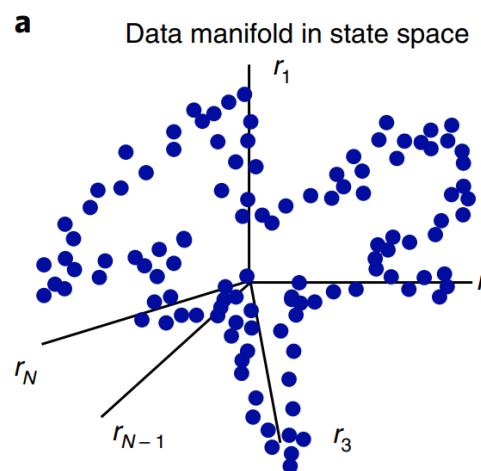


Preserved neural dynamics across animals performing similar behaviour

Safaie et al., nature 2023

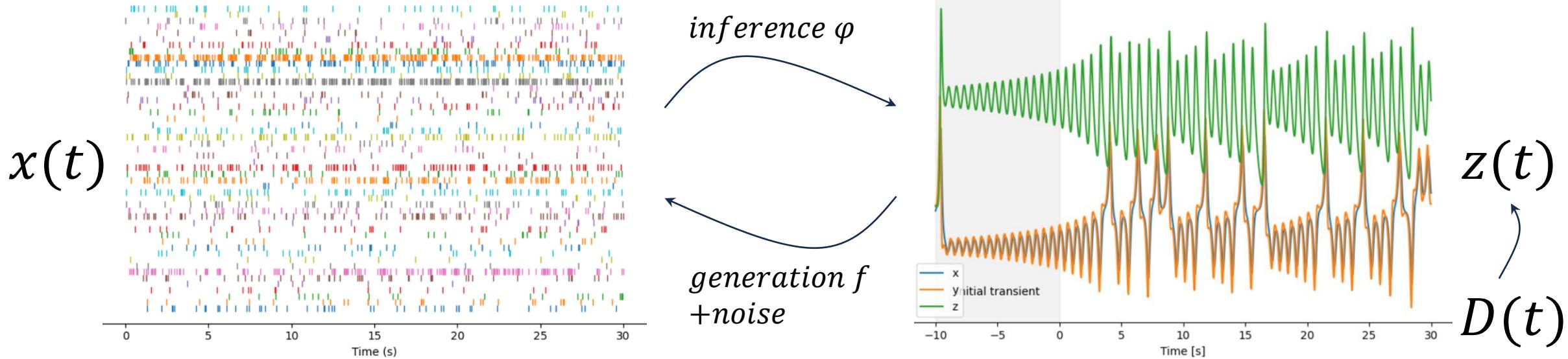


Manifold learning and characterization



Assumption: Latent state generative model of neural population

Task: Inferring Latent Dynamics Underlying Neural Population Activity



The strong principle is that dimension reduction shows us the true underlying signal embodied by the neural circuit, the so-called "latent signal." It is a theory that the brain really is low-dimensional compared to the number of neurons. The joint activity of a population of neurons is a (noisy) realization of this low-dimensional system—a realization using many more elements (neurons) than dimensions in the system.

Assumption: Latent state generative model of neural population

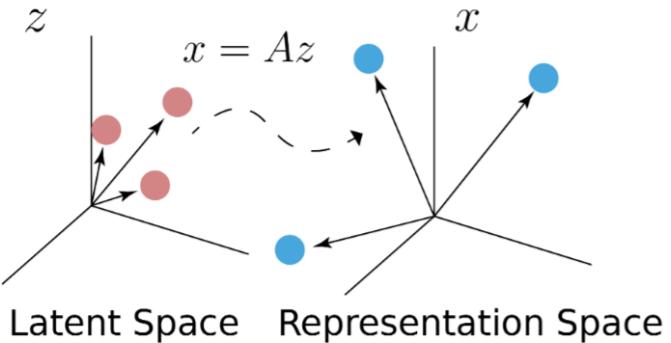
Task: Inferring Latent Dynamics Underlying Neural Population Activity

Table 1 Overview of dimensionality reduction methods commonly applied to neural population responses

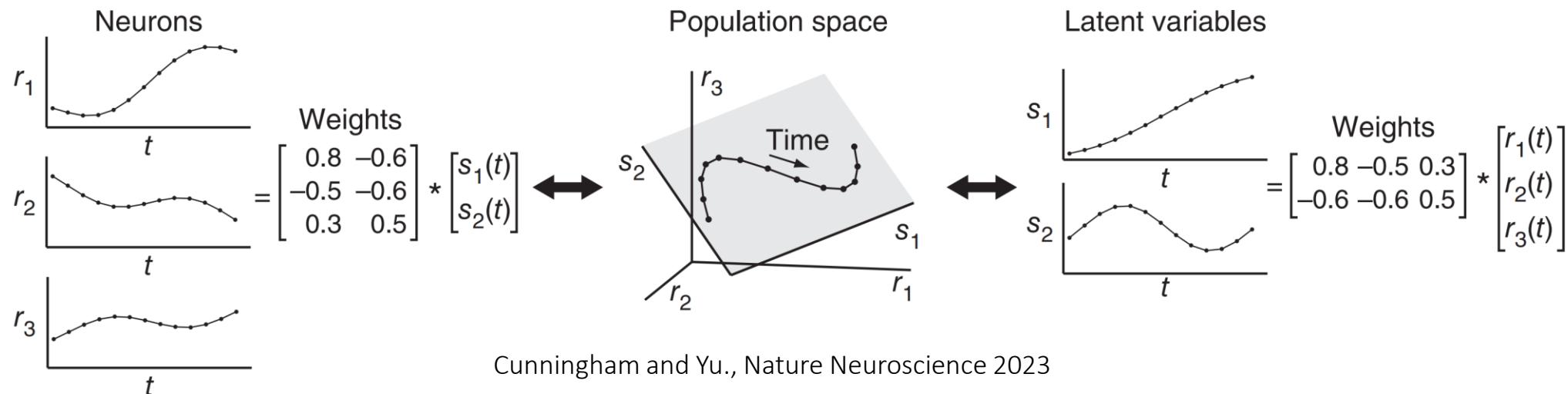
Method	Analysis objective	Temporal smoothing	Explicit noise model	Representative uses in neuroscience (refs.)
PCA	Covariance	No	No	6,29,50
FA	Covariance	No	Yes	10,43,100
LDS/GPFA	Dynamics	Yes	Yes	12,71,72
NLDS	Dynamics	Yes	Yes	77,78
LDA	Classification	No	No	9,19,56
Demixed	Regression	No	Yes/No	4,16
Isomap/LLE	Manifold discovery	No	No	14,44,58

Assumption: Latent state generative model of neural population

Task: Inferring Latent Dynamics Underlying Neural Population Activity

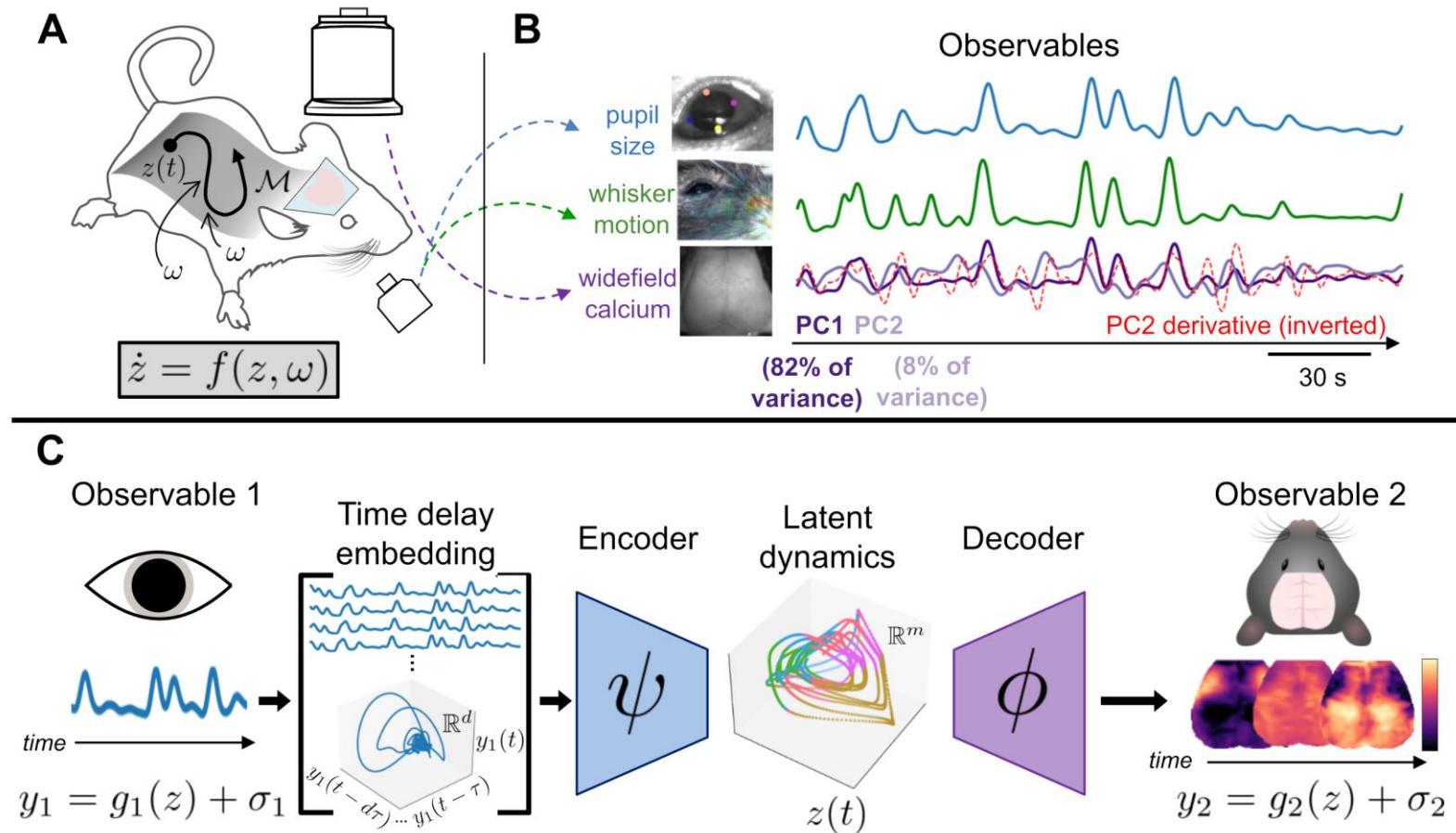


In PCA, the mapping function f is an instantaneous linear function and the noise is Gaussian (in the probabilistic extension of PCA). The dynamics are not explicitly modeled. The inference function is the matrix inverse of the mapping function



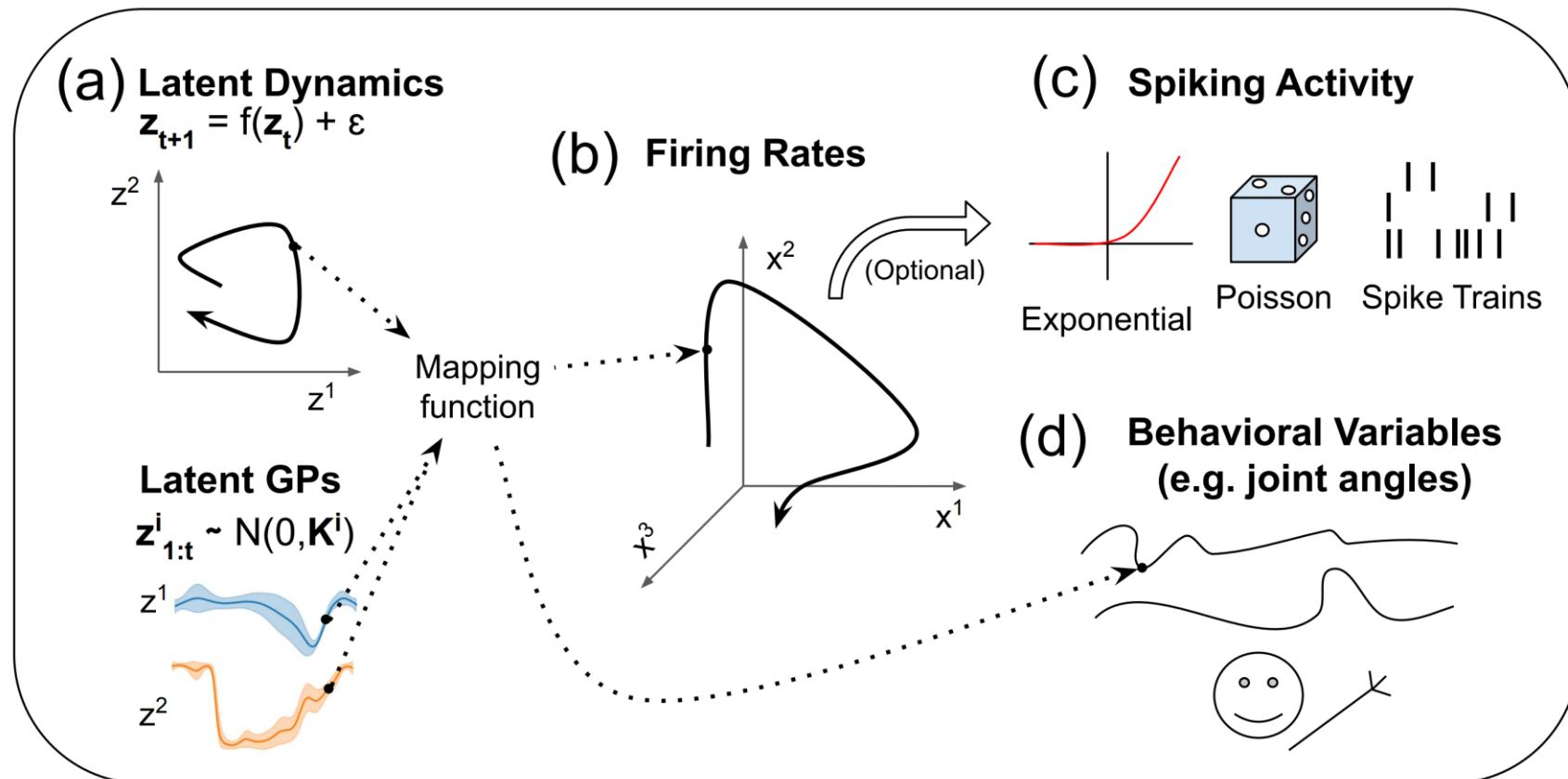
Assumption: Latent state generative model of neural population

Task: Inferring Latent Dynamics Underlying Neural Population Activity



Assumption: Latent state generative model of neural population

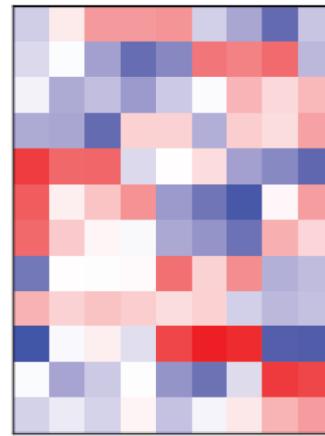
Task: Inferring Latent Dynamics Underlying Neural Population Activity



References

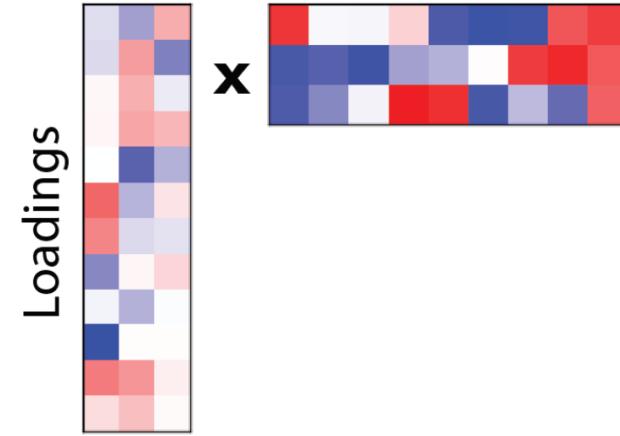
- Dimensionality reduction for large-scale neural recordings, Cunningham and Yu., Nature Neuroscience 2014
- Neural population geometry and optimal coding of tasks with shared latent structure Albert J. Wakhloo et al., arXiv 2024
- Strong and weak principles of neural dimension reduction, Mark D. Humphries. arXiv 2023
- Building population models for large-scale neural recordings: opportunities and pitfalls, Cole Hurwitz., arXiv 2021
- Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification, Sani, et al. Nature Neuroscience 2019
- Large-scale neural recordings call for new insights to link brain and behavior, Urai, et al., Nature Neuroscience, 2022
- Neural population geometry: An approach for understanding biological and artificial neural networks, Chung and Abbott. Current opinion in neurobiology, 2021
- Dimensionality reduction in neuroscience. Pang et al., Current Biology, 2016
- On simplicity and complexity in the brave new world of large-scale neuroscience. Gao P, Ganguli S., Curr Opin Neurobiol 2015
- Navigating the Neural Space in Search of the Neural Code, Jazayeri and Afraz, Neuron 2017
- Computation Through Neural Population Dynamics. Vyas et al. Annu Rev Neurosci 2020
- A theory of multineuronal dimensionality, dynamics, and measurement. Gao et al. bioRxiv 2017
- Is population activity more than the sum of its parts? Pillow and Aoi, Nature Neuroscience 2017
- Structure in neural population recordings: an expected byproduct of simpler phenomena? Elsayed and Cunningham, Nature Neuroscience 2017
- The role of untuned neurons in sensory information coding, Zylberberg, bioRxiv 2018
- Linking Connectivity, Dynamics, and Computations in Low-Rank Recurrent Neural Networks. Mastrogiuseppe and Ostojic. Neuron 2018
- Compressed Sensing, Sparsity, and Dimensionality in Neuronal Information Processing and Data Analysis, Ganguli and Sompolinsky, Annu. Rev. Neurosci. 2012

Original Data

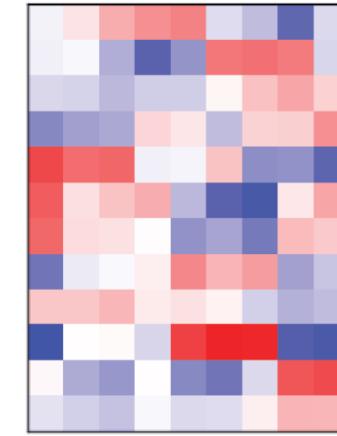


\approx

Components

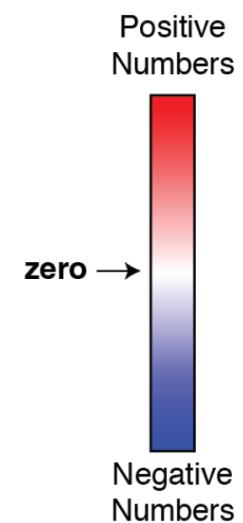
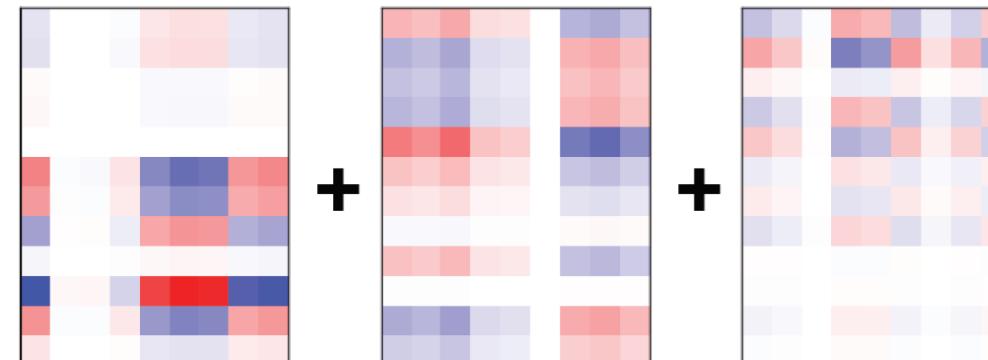


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax



neurosyntax

Session 5

Lecturer: Saman Abbaspoor

Singular Value Decomposition SVD

Principal Component Analysis (PCA)

Computation

$$\begin{matrix} \text{Data transposed} \\ (\text{mean-subtracted}) \end{matrix} \times \begin{matrix} \text{Data} \\ (\text{mean-subtracted}) \end{matrix} = \begin{matrix} \text{Covariance} \\ \text{matrix} \end{matrix} = \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \\ \text{transposed} \end{matrix} \times \begin{matrix} \text{Eigenvalues} \\ (\lambda) \end{matrix} \times \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \end{matrix}$$

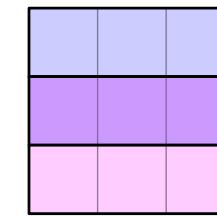
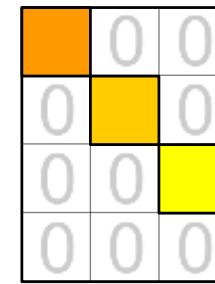
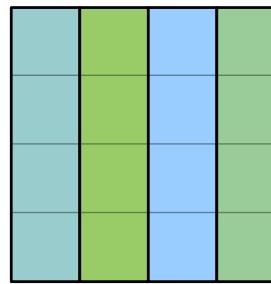
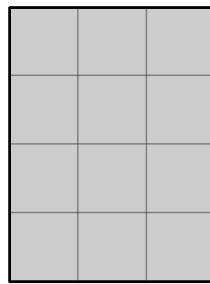
Dimensionality Reduction

$$\begin{matrix} \text{Data} \\ (\text{mean-subtracted}) \end{matrix} \times \begin{matrix} \text{Eigenvectors} \\ (\text{loadings}) \\ \text{transposed} \end{matrix} = \begin{matrix} \text{Scores} \end{matrix}$$

Singular Value Decomposition

No restriction on Symmetry, Dimension, Rank

$$M = U\Sigma V^*$$



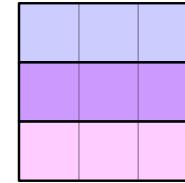
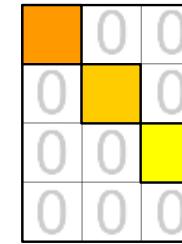
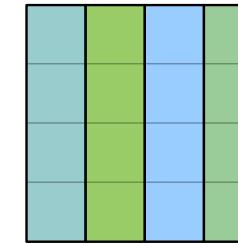
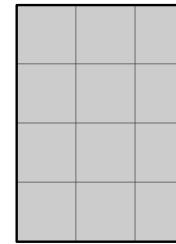
$$\begin{matrix} \mathbf{M} &= & \mathbf{U} & \Sigma & \mathbf{V}^* \\ m \times n & & m \times m & m \times n & n \times n \end{matrix}$$

Singular Value Decomposition

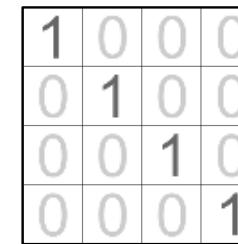
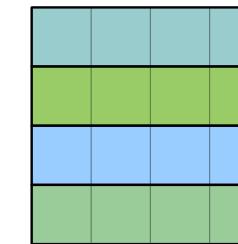
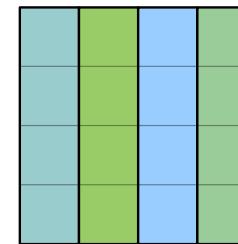
No restriction on Symmetry, Dimension, Rank

$$M = U\Sigma V^*$$

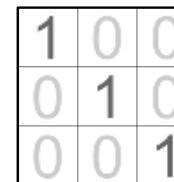
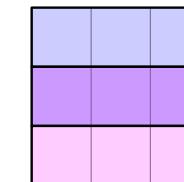
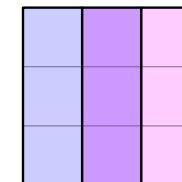
Orthogonal Rectangular Orthogonal
 Diagonal



$$\begin{matrix} M \\ m \times n \end{matrix} = \begin{matrix} U \\ m \times m \end{matrix} \begin{matrix} \Sigma \\ m \times n \end{matrix} \begin{matrix} V^* \\ n \times n \end{matrix}$$



$$\begin{matrix} U \\ m \times m \end{matrix} = \begin{matrix} I_m \end{matrix}$$



$$\begin{matrix} V \\ n \times n \end{matrix} = \begin{matrix} I_n \end{matrix}$$

$$A \begin{bmatrix} M \times N \end{bmatrix}$$

$$\begin{bmatrix} M \times M \end{bmatrix}$$

$$AA^T = S_L$$

$$V_1 \quad V_2 \quad \dots \quad V_m$$

$$\begin{bmatrix} N \times N \end{bmatrix}$$

$$A^TA = S_R$$

$$V_1 \quad V_2 \quad \dots \quad V_n$$

$$\begin{array}{cccc} \lambda_1 & \lambda_2 & & \lambda_m \\ | & | & & | \\ V_1 & V_2 & \cdots & V_m \\ | & | & & | \end{array}$$

S_L and S_R are PSD (positive semi-definite) matrices

Eigenvalues corresponding to each eigenvector is nonnegative

$$\begin{array}{cccc} \lambda_{1*} & \lambda_{2*} & & \lambda_{n*} \\ | & | & & | \\ V_1 & V_2 & \cdots & V_n \\ | & | & & | \end{array}$$

After sorting eigenvalues in descending orders,

$$\lambda_1 = \lambda_{1*} > \lambda_2 = \lambda_{2*} > \lambda_n = \lambda_{n*} > 0$$

$$(\lambda > \lambda_n = \lambda_{n*}) = 0$$

Singular Value Decomposition

No restriction on Symmetry, Dimension, Rank

tall, skinny A:

$$\begin{matrix} A \\ \hline \end{matrix} = \begin{matrix} U & \text{irrelevant} & S & V^\top \\ \hline U & & S & V^\top \end{matrix}$$

$$A \quad U \quad S \quad V^\top$$

short, fat A:

$$\begin{matrix} A \\ \hline \end{matrix} = \begin{matrix} U & S & \text{zeros} & V^\top \\ \hline & & & \text{irrelevant} \end{matrix}$$

Eigenvalues of matrix A

$$\sqrt{\lambda_1} \quad \sqrt{\lambda_2} \quad \dots \quad \sqrt{\lambda_n}$$

$$\sigma_1 \quad \sigma_2 \quad \dots \quad \sigma_n$$

Singular values of matrix A

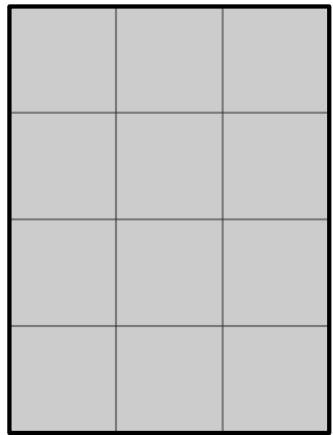
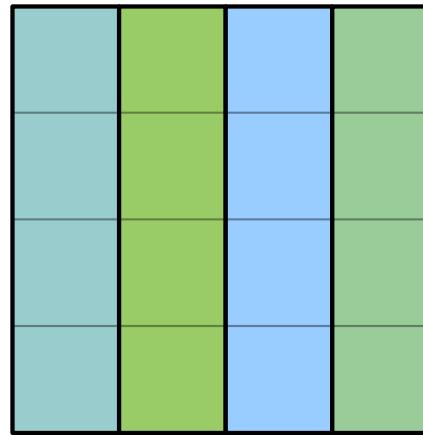
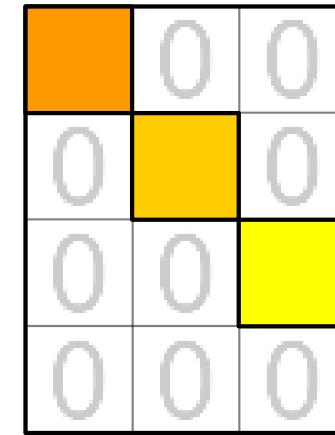
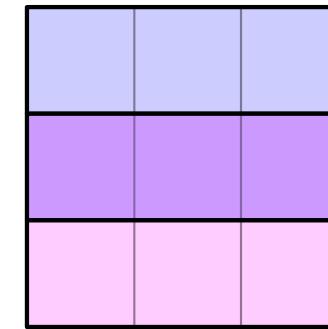
$$svd(AA^T)?$$

$$A = U\Sigma V^*$$

$$AA^T = (U\Sigma V^*)(V\Sigma U^*)$$

$$AA^T = (U\Sigma^2 U^T)$$

$$svd(AA^T) = eig(AA^T) = pca(AA^T)[if \ A \ is \ standardized]$$

A  U  Σ  V^* 

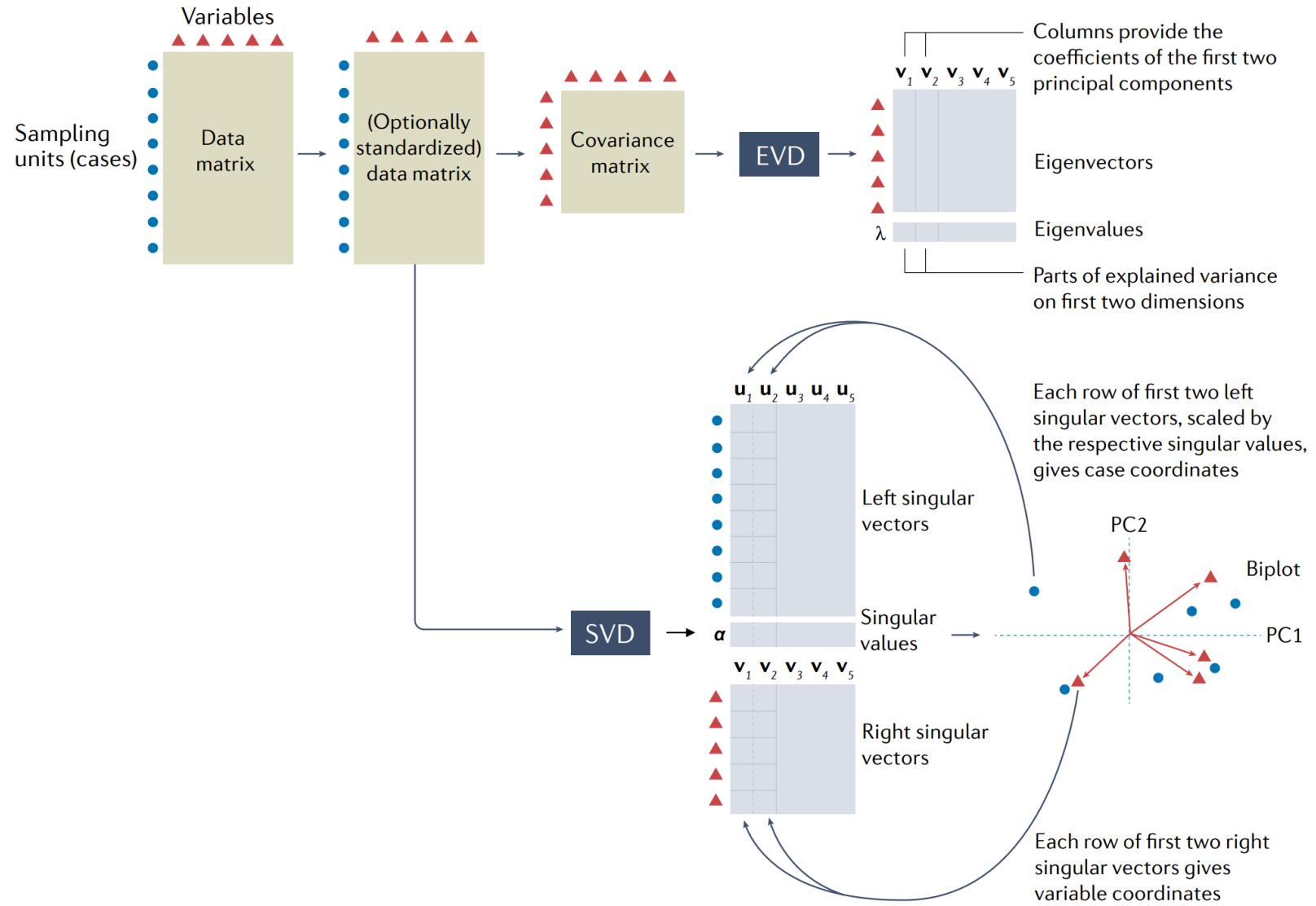
Eigenvectors of

 S_L 

Eigenvectors of

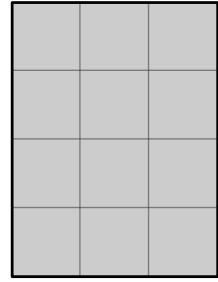
 S_R

Eigenvalues

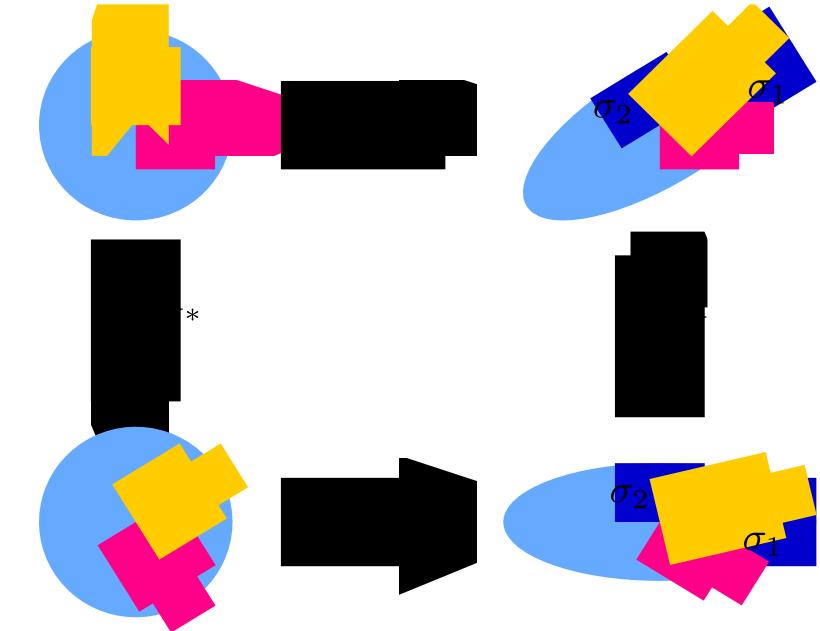
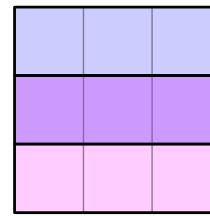
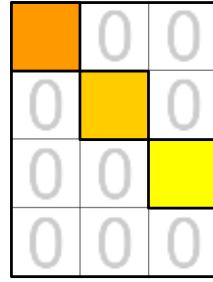


Singular Value Decomposition

No restriction on Symmetry, Dimension, Rank



$$M_{m \times n} = U_{m \times m} \Sigma_{m \times n} V^*_{n \times n}$$



$$M = U \cdot \Sigma \cdot V^*$$



Low rank approximation with SVD

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ v_3^T \end{bmatrix}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \sigma_1 \begin{bmatrix} v_1^T \\ u_1 \end{bmatrix} + \sigma_2 \begin{bmatrix} v_2^T \\ u_2 \end{bmatrix} + \sigma_3 \begin{bmatrix} v_3^T \\ u_3 \end{bmatrix}$$

Low rank approximation with SVD



Low rank approximation with SVD

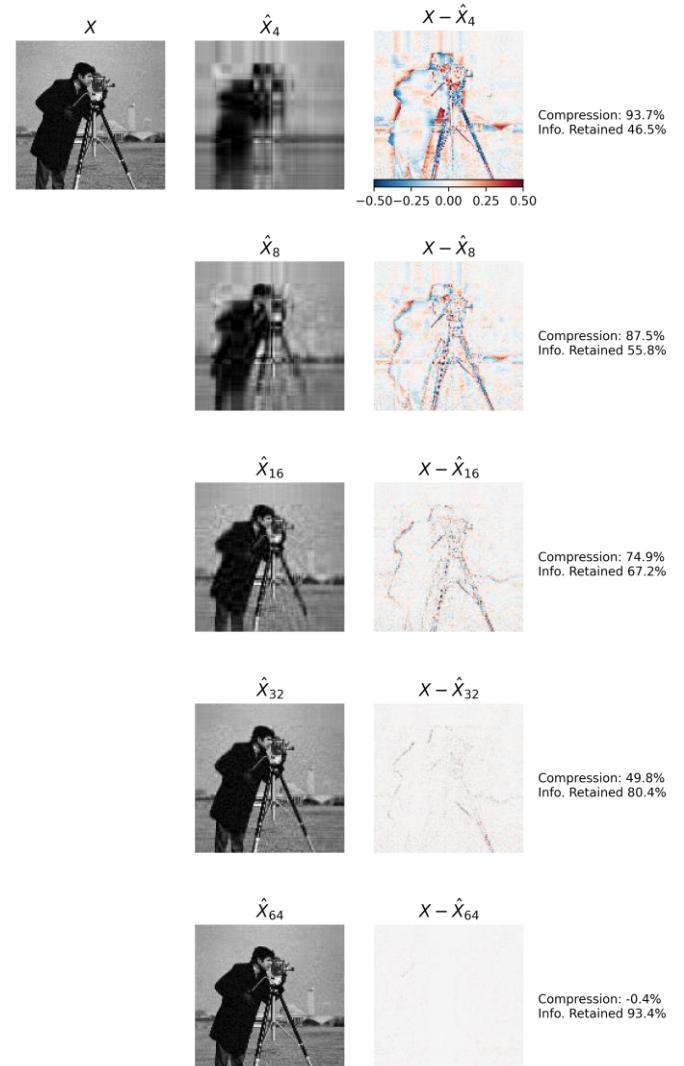
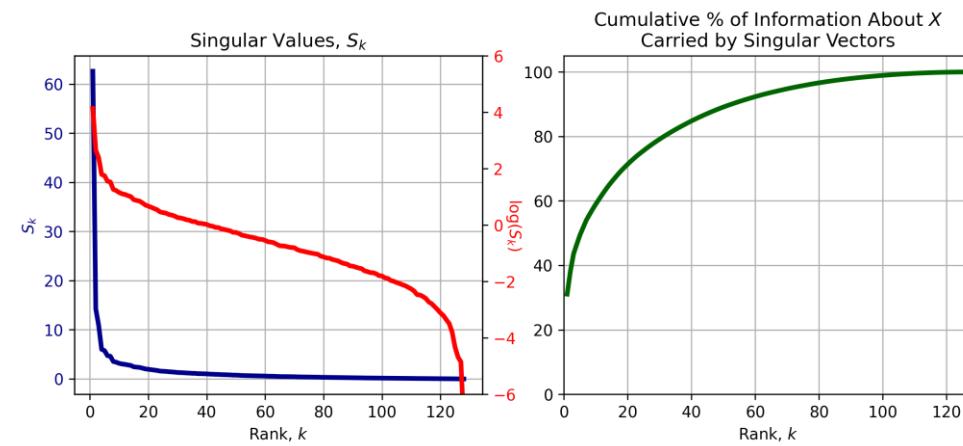
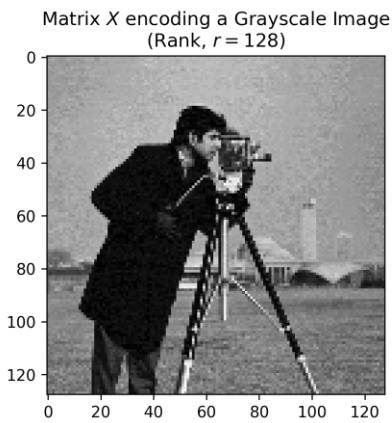


Figure 3: Singular Value Decomposition of an image X . **Left:** A Grayscale image can be interpreted as a matrix X . **Center:** the singular values (blue) and their log (red) as a function of rank k . Singular values decrease exponentially with rank, with earlier singular values being much larger than later ones. **Right:** The total information about X encoded in all the singular values up to k . A majority of information is encoded in the first singular vectors returned by SVD.

Eckard-Young Theorem

The theorem states that for any given matrix A and a positive integer k , the best rank- k approximation A_k of A , in terms of minimizing the Frobenius norm of the difference between A and A_k , is given by the first k terms of the singular value decomposition (SVD) of A . This approximation is expressed as:

$$A_k = U_k \Sigma_k V_k^T$$

where:

- U_k and V_k are the first k columns of the matrices U and V from the SVD of A , respectively, which correspond to the k largest singular values.
- Σ_k is the diagonal matrix containing the k largest singular values.

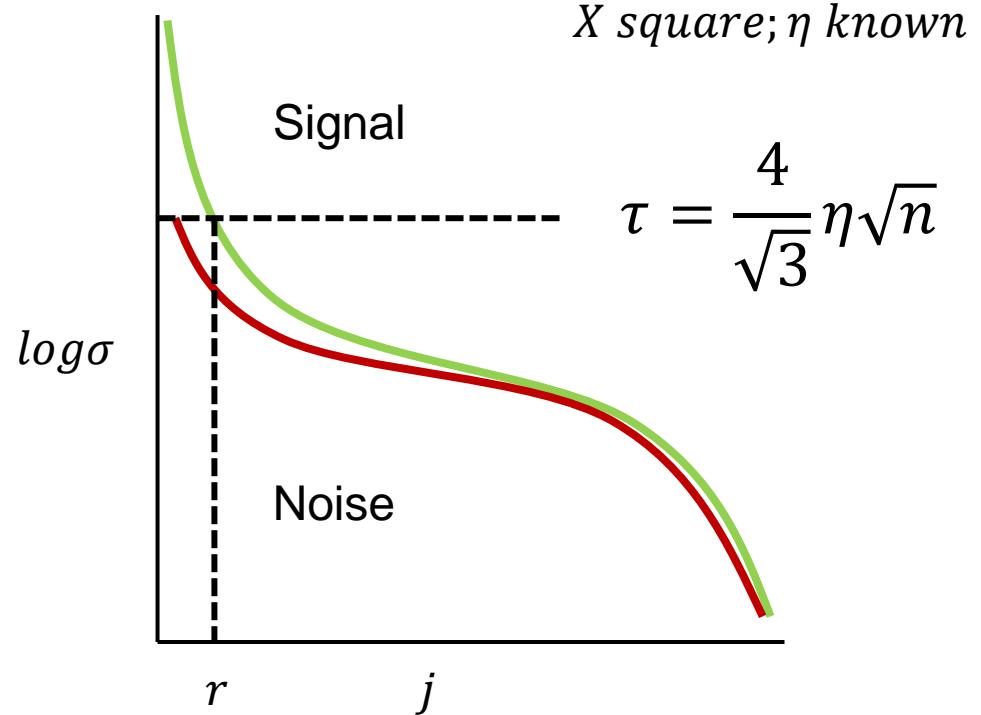
Optimal Truncation

$$X = X_{true} + \eta X_{noise}$$

$N(0,1)$

Our central observation is as follows.

When a data singular value y_i is too small, then its associated singular vectors $\mathbf{u}_i, \mathbf{v}_i$ are so noisy that the component $y_i \mathbf{u}_i \mathbf{v}'_i$ should not included in \hat{X} . In our asymptotic framework, which models large, low-rank matrices observed in white noise, the cutoff below which y_i is too small is exactly $(4/\sqrt{3})\sqrt{n}\sigma$ (for square matrices).



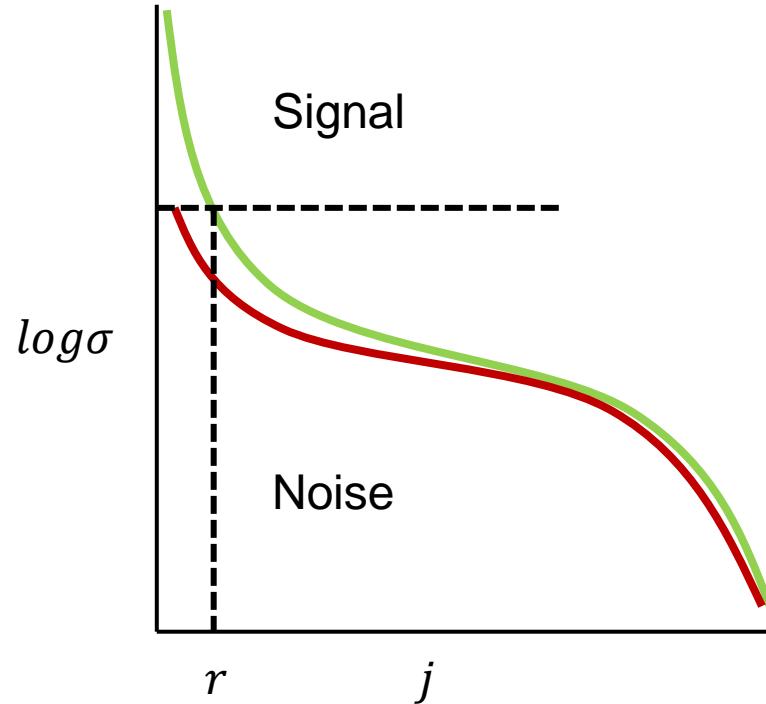
Optimal Truncation

$$X = X_{true} + \eta X_{noise}$$

$N(0,1)$

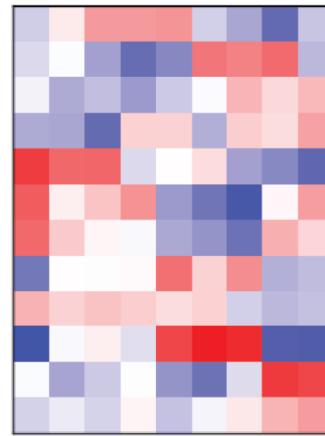
X rectangular ($\beta = \frac{n}{m}$)

η unknown?



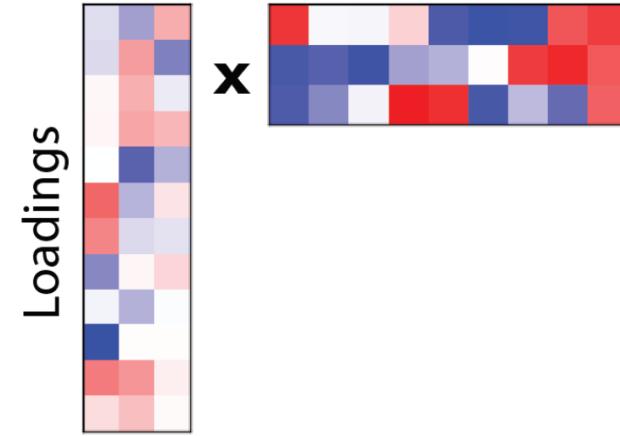


Original Data

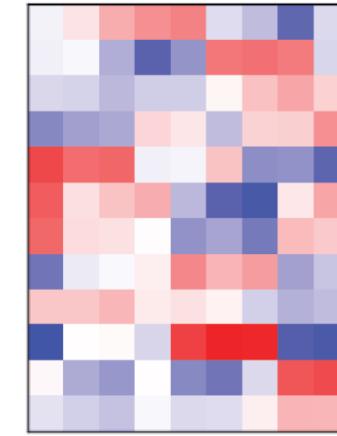


\approx

Components

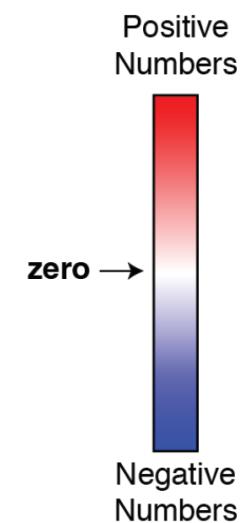
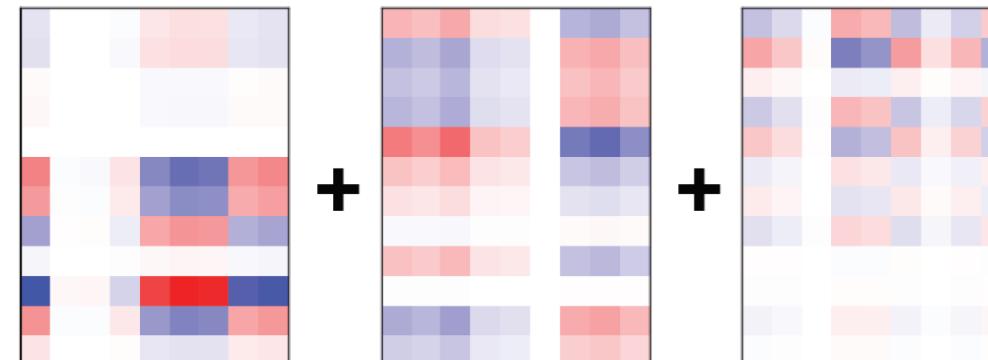


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax



neurosyntax

Session 6

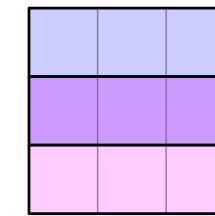
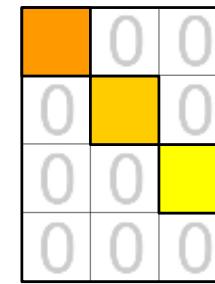
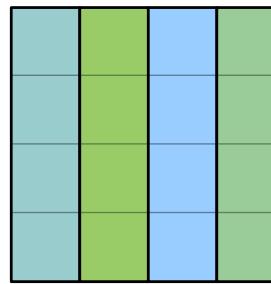
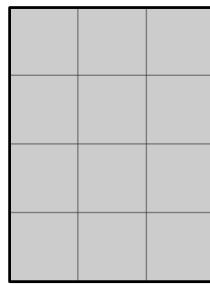
Lecturer: Saman Abbaspoor

Singular Value Decomposition SVD

Singular Value Decomposition

No restriction on Symmetry, Dimension, Rank

$$M = U\Sigma V^*$$



$$\begin{matrix} \mathbf{M} &= & \mathbf{U} & \Sigma & \mathbf{V}^* \\ m \times n & & m \times m & m \times n & n \times n \end{matrix}$$

Eigenvalues of matrix A

$$\sqrt{\lambda_1} \quad \sqrt{\lambda_2} \quad \dots \quad \sqrt{\lambda_n}$$

$$\sigma_1 \quad \sigma_2 \quad \dots \quad \sigma_n$$

Singular values of matrix A

$$A \begin{bmatrix} M \times N \end{bmatrix}$$

$$\begin{bmatrix} M \times M \end{bmatrix}$$

$$AA^T = S_L$$

$$V_1 \quad V_2 \quad \dots \quad V_m$$

$$\begin{bmatrix} N \times N \end{bmatrix}$$

$$A^TA = S_R$$

$$V_1 \quad V_2 \quad \dots \quad V_n$$

$$T \times T$$
$$\begin{bmatrix} & \\ & \end{bmatrix}$$

$$AA^T$$

$$Data$$
$$\begin{bmatrix} & \\ & M \times N \\ & \end{bmatrix}$$

$$N \times N$$
$$\begin{bmatrix} & \\ & \end{bmatrix}$$

$$A^T A$$

$$SVD = U \Sigma V^*$$



Low rank approximation with SVD

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ v_3^T \end{bmatrix}$$

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \sigma_1 \begin{bmatrix} v_1^T \\ u_1 \end{bmatrix} + \sigma_2 \begin{bmatrix} v_2^T \\ u_2 \end{bmatrix} + \sigma_3 \begin{bmatrix} v_3^T \\ u_3 \end{bmatrix}$$

Low rank approximation with SVD

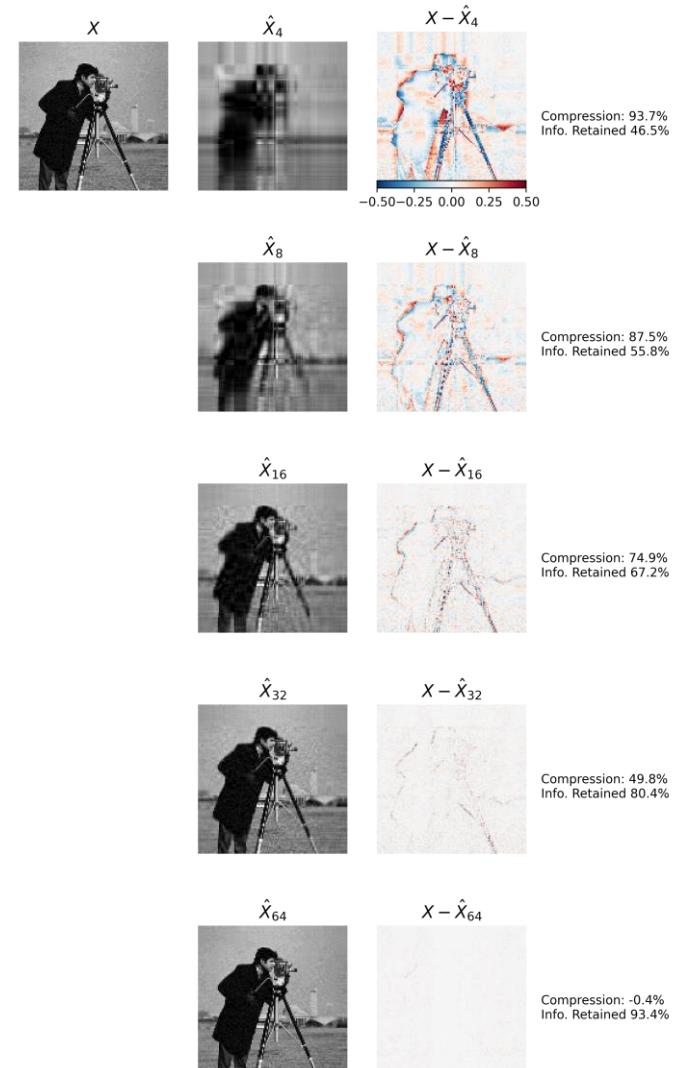
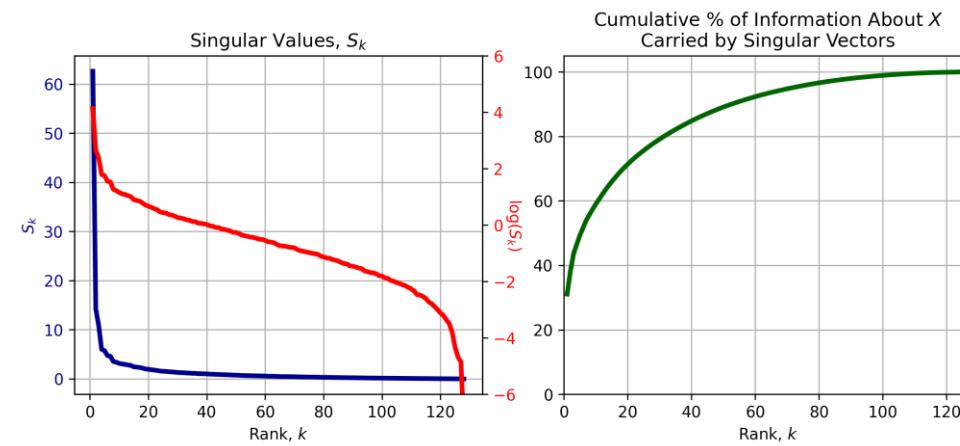
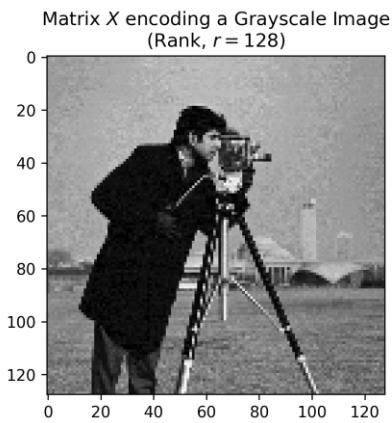
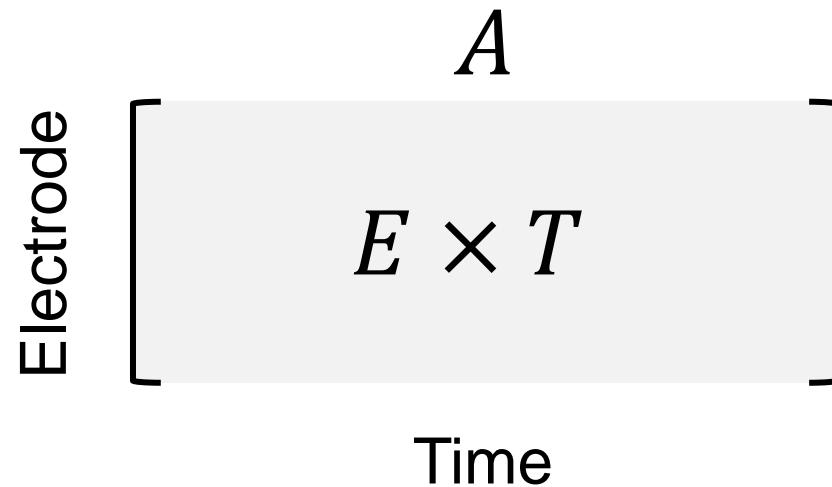
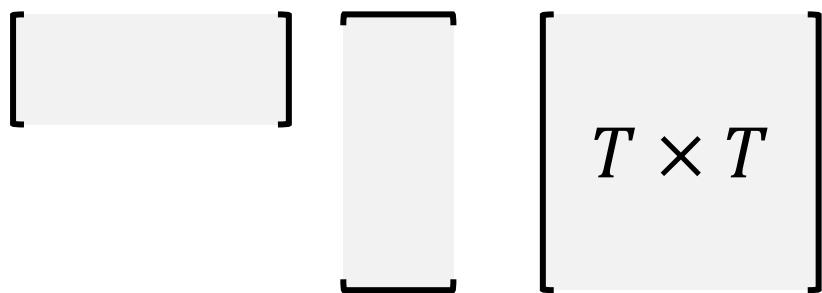


Figure 3: Singular Value Decomposition of an image X . **Left:** A Grayscale image can be interpreted as a matrix X . **Center:** the singular values (blue) and their log (red) as a function of rank k . Singular values decrease exponentially with rank, with earlier singular values being much larger than later ones. **Right:** The total information about X encoded in all the singular values up to k . A majority of information is encoded in the first singular vectors returned by SVD.

Dominant correlation patterns from SVD

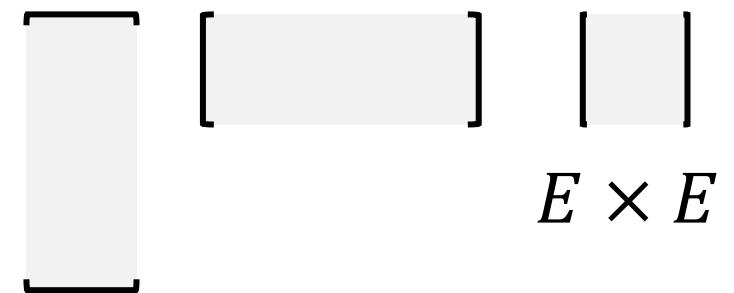


Temporal Modes



$$AA^T = S_L$$

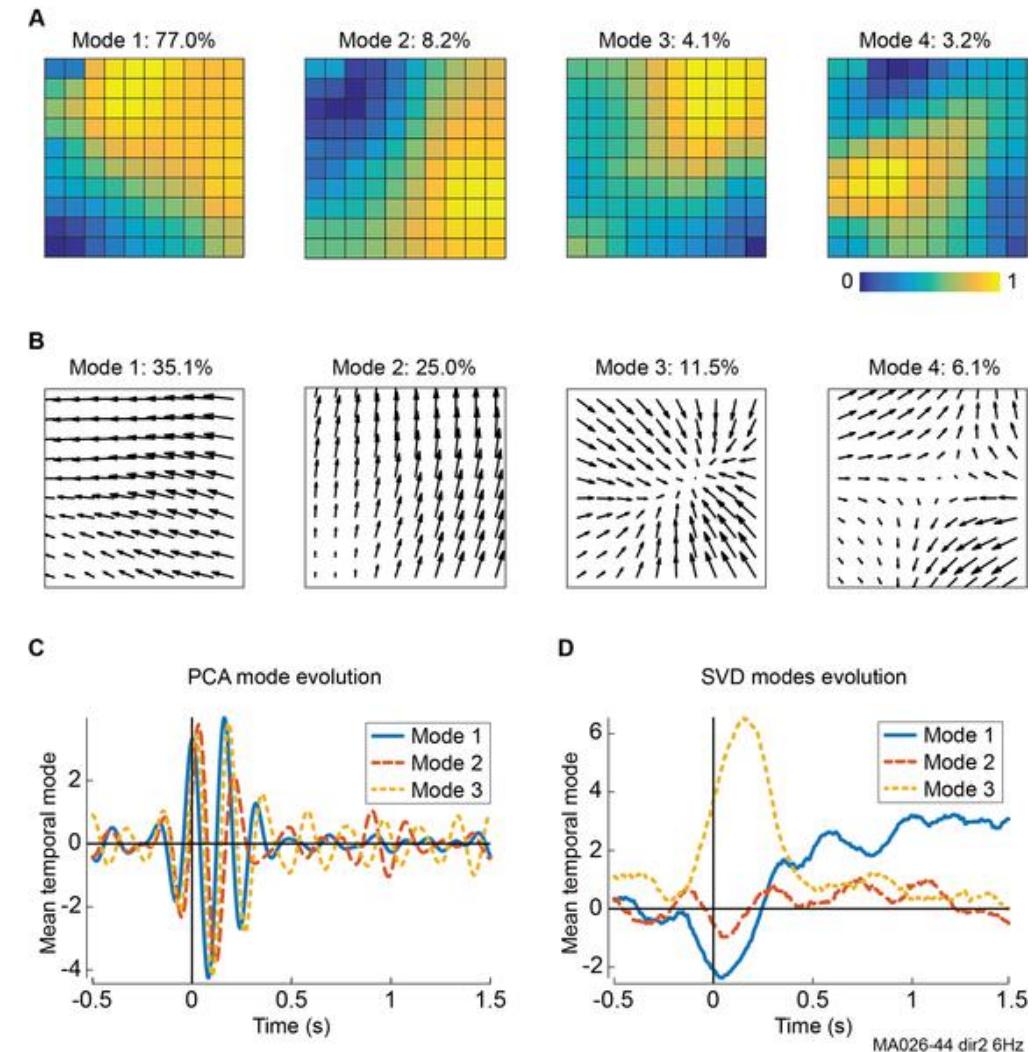
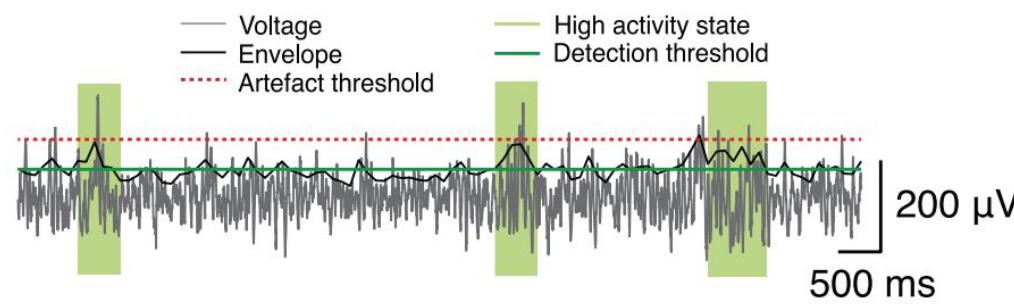
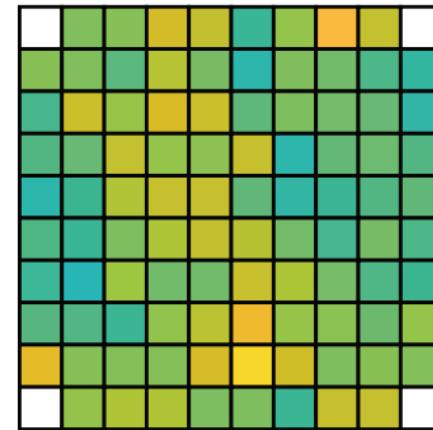
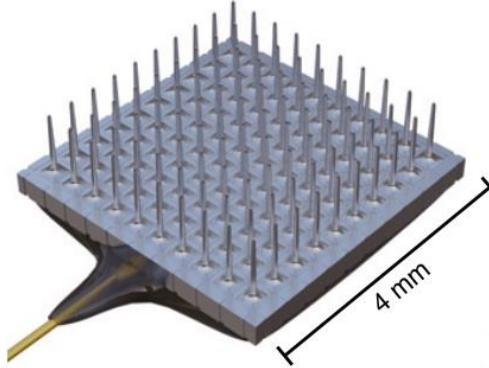
Spatial Modes



$$A^T A = S_R$$

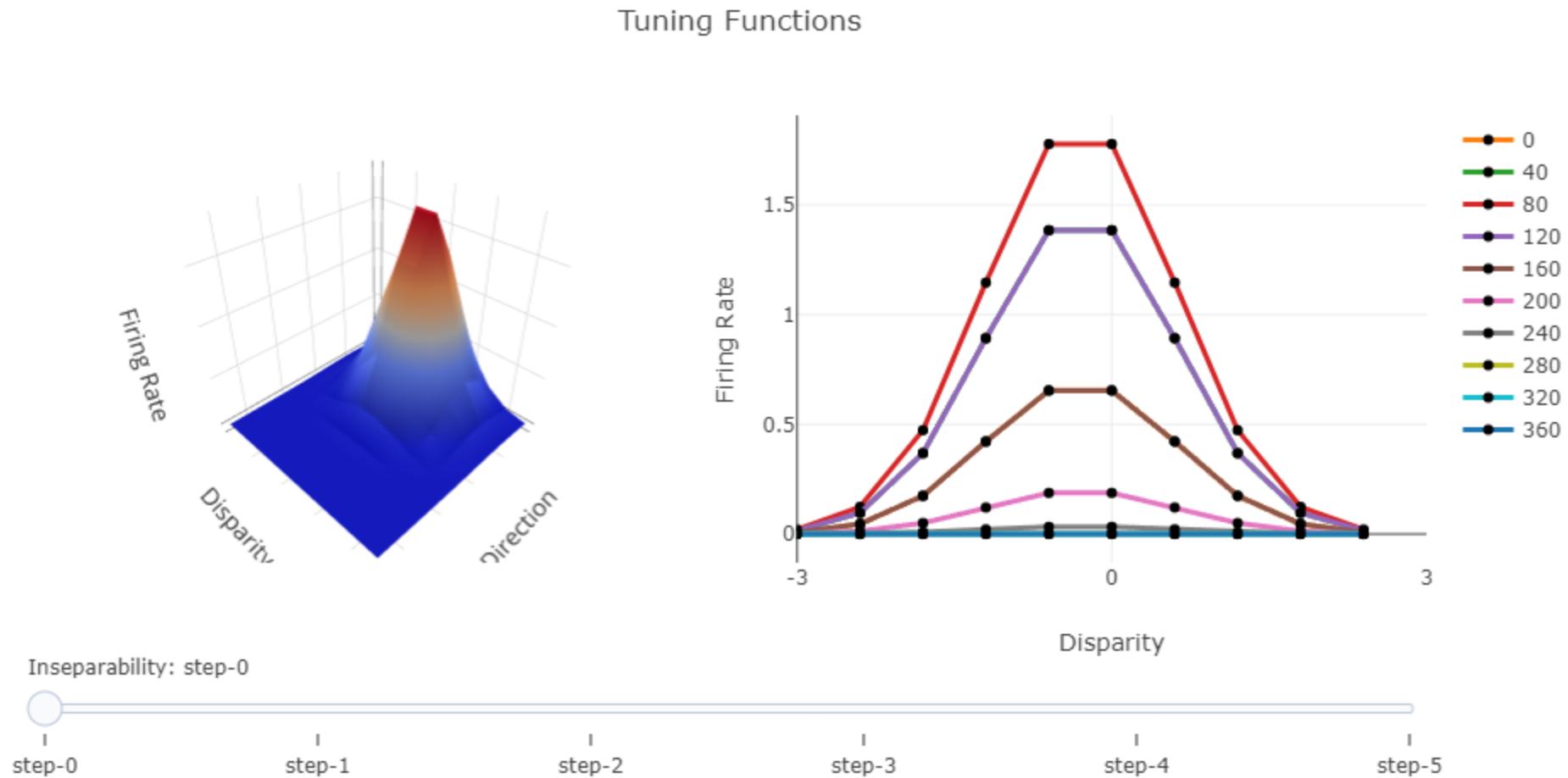
Dominant spatiotemporal modes in LFP

Higher density (96 ch.)

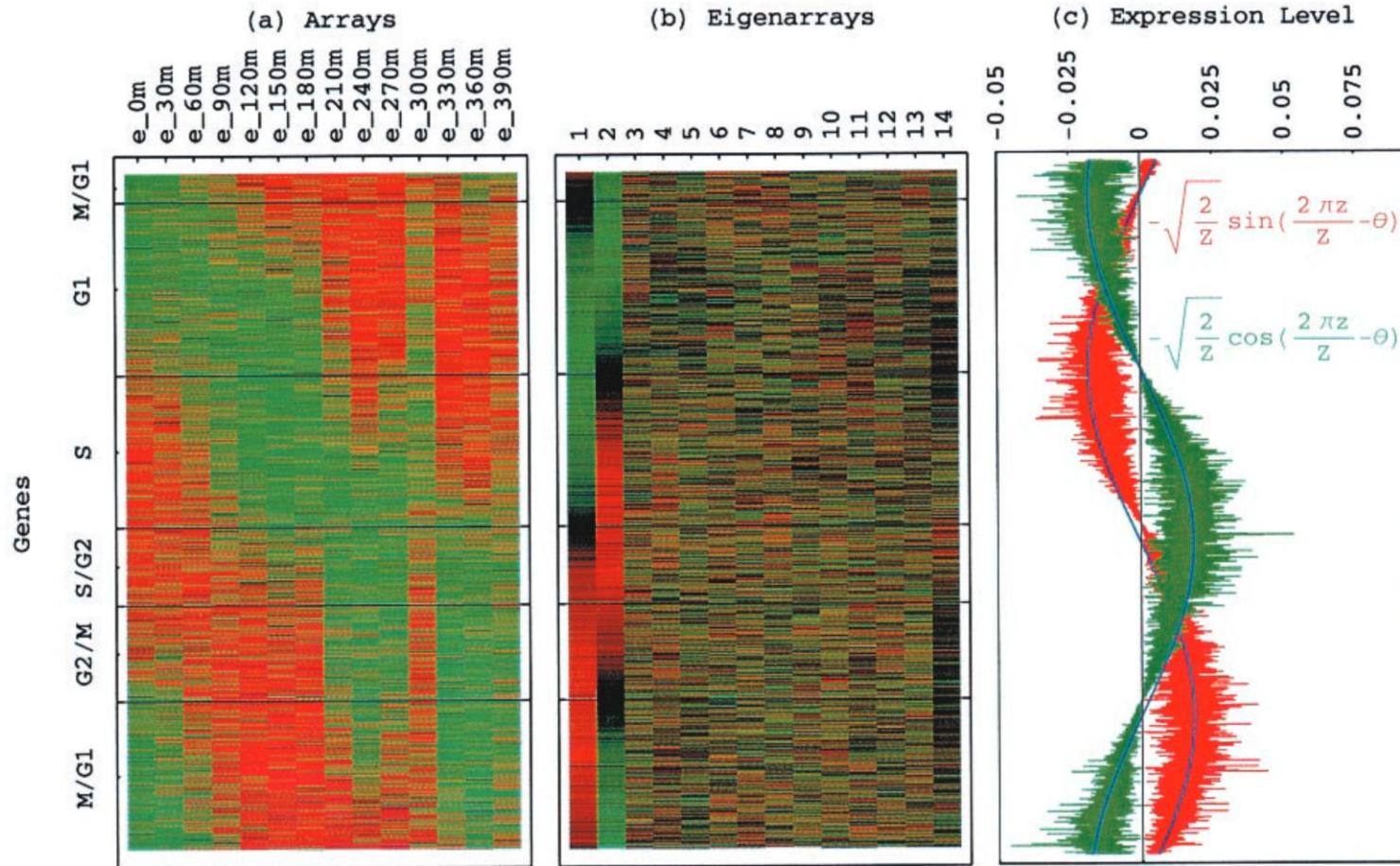




Application of SVD in neuroscience: Firing rate tuning



SVD for genome-wide expression data processing and modeling



Alter et al., PNAS 2000

Sparse PCA

Principal Component Analysis (PCA)

$$\begin{matrix} & \times \\ \text{Data transposed} \\ \text{(mean-subtracted)} & \times \\ \text{Data} \\ \text{(mean-subtracted)} & = \\ & = \\ \text{Covariance} \\ \text{matrix} & \times \\ \text{Eigenvalues} \\ (\lambda) & \times \\ \text{Eigenvectors} \\ \text{(loadings)} \\ \text{transposed} & \times \\ \text{Eigenvectors} \\ \text{(loadings)} \end{matrix}$$

The diagram illustrates the mathematical decomposition of a covariance matrix. It starts with two matrices multiplied together: 'Data transposed (mean-subtracted)' and 'Data (mean-subtracted)'. This equals the 'Covariance matrix'. This covariance matrix is then decomposed into three components: 'Eigenvalues (λ)' (represented by a grid of numbers), 'Eigenvalues (λ)' (represented by a single value λ), and 'Eigenvectors (loadings) transposed' (represented by a vertical stack of colored bars). Finally, the product of 'Eigenvalues (λ)' and 'Eigenvectors (loadings)' is multiplied by 'Eigenvalues (λ)' again to yield the original covariance matrix.

Principal component analysis (PCA) is widely used in data processing and dimensionality reduction. However, PCA suffers from the fact that each principal component is a linear combination of all the original variables, thus it is often difficult to interpret the results.

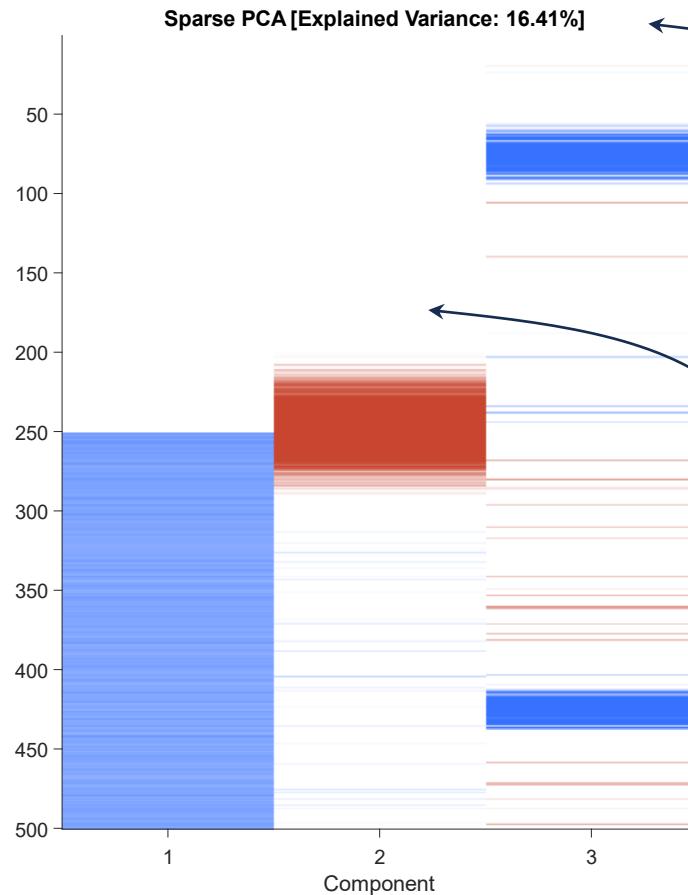
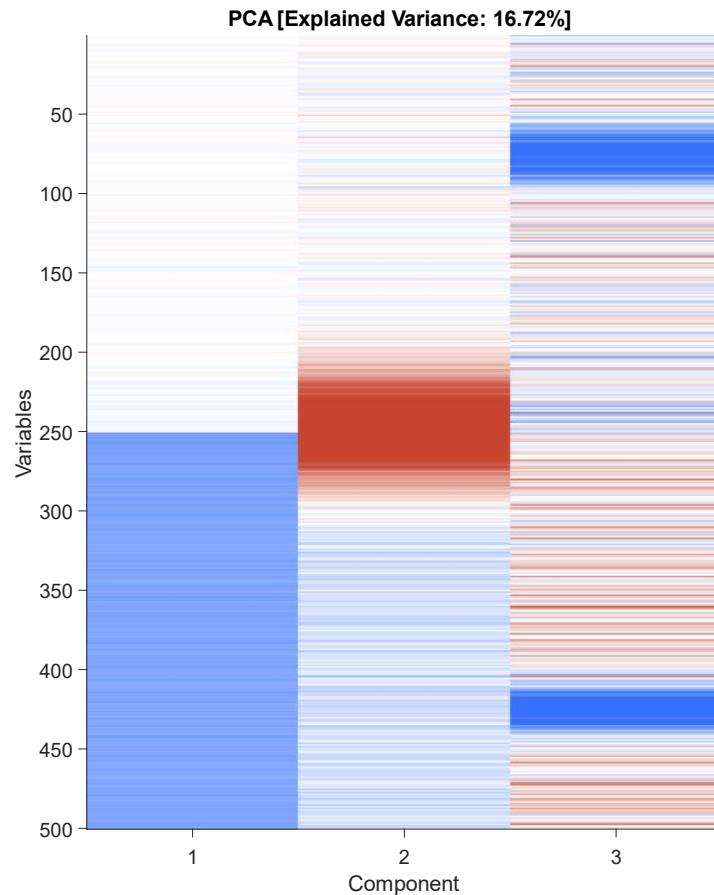
Sparsity in statistics

In statistics, sparsity refers to situations where data or model parameters predominantly contain zero or near-zero values. This characteristic often indicates that only a small number of features or variables significantly contribute to the model or analysis, enhancing interpretability, reducing complexity, and potentially improving computational efficiency.

Sparsity in Data: When a dataset is sparse, most of its elements are zeros (e.g., word counts where most words do not appear in most documents)

Sparsity in Models: Statistical and machine learning models can exploit sparsity to achieve more efficient and interpretable outcomes. Techniques such as Lasso regression or sparse PCA are designed to encourage sparsity in model coefficients, effectively performing feature selection by driving the coefficients of less relevant features to zero.

Sparse PCA



Without loss of much explained variance

Most variables in the loadings have 0 weights

Statistics > Methodology

[Submitted on 11 Dec 2017 (v1), last revised 23 May 2018 (this version, v2)]

Statistical sparsity

Peter McCullagh, Nicholas Polson

The main contribution of this paper is a mathematical definition of statistical sparsity, which is expressed as a limiting property of a sequence of probability distributions. The limit is characterized by an exceedance measure~ H and a rate parameter~ $\rho > 0$, both of which are unrelated to sample size. The definition is sufficient to encompass all sparsity models that have been suggested in the signal-detection literature. Sparsity implies that ρ is small, and a sparse approximation is asymptotic in the rate parameter, typically with error $o(\rho)$ in the sparse limit $\rho \rightarrow 0$. To first order in sparsity, the sparse signal plus Gaussian noise convolution depends on the signal distribution only through its rate parameter and exceedance measure. This is one of several asymptotic approximations implied by the definition, each of which is most conveniently expressed in terms of the zeta-transformation of the exceedance measure. One implication is that two sparse families having the same exceedance measure are inferentially equivalent, and cannot be distinguished to first order. A converse implication for methodological strategy is that it may be more fruitful to focus on the exceedance measure, ignoring aspects of the signal distribution that have negligible effect on observables and on inferences. From this point of view, scale models and inverse-power measures seem particularly attractive.

Access Paper:

- [View PDF](#)
- [TeX Source](#)
- [Other Formats](#)

[view license](#)

Current browse context:

stat.ME

[< prev](#) | [next >](#)

[new](#) | [recent](#) | [2017-12](#)

Change to browse by:

stat

References & Citations

- [NASA ADS](#)
- [Google Scholar](#)
- [Semantic Scholar](#)

[Export BibTeX Citation](#)

Bookmark



Sparse Principal Component Analysis

Hui ZOU, Trevor HASTIE, and Robert TIBSHIRANI

Principal component analysis (PCA) is widely used in data processing and dimensionality reduction. However, PCA suffers from the fact that each principal component is a linear combination of all the original variables, thus it is often difficult to interpret the results. We introduce a new method called sparse principal component analysis (SPCA) using the *lasso* (*elastic net*) to produce modified principal components with sparse loadings. We first show that PCA can be formulated as a regression-type optimization problem; sparse loadings are then obtained by imposing the lasso (elastic net) constraint on the regression coefficients. Efficient algorithms are proposed to fit our SPCA models for both regular multivariate data and gene expression arrays. We also give a new formula to compute the total variance of modified principal components. As illustrations, SPCA is applied to real and simulated data with encouraging results.

Key Words: Arrays; Gene expression; Lasso/elastic net; Multivariate analysis; Singular value decomposition; Thresholding.



[Willow Project](#)
[Sierra Project](#)
[Thoth Project](#)

[About](#)
[Downloads](#)
[Documentation](#)
[FAQ](#)
[Contacts](#)
[Known Bugs](#)

SPAMS

About

For any question related to the use or development of SPAMS, you can contact us at "**spams.dev'AT'inria.fr**" (replace 'AT' by @).

What is SPAMS?

SPAMS (SPArse Modeling Software) is an optimization toolbox for solving various sparse estimation problems.

- Dictionary learning and matrix factorization (NMF, sparse PCA, ...)
- Solving sparse decomposition problems with LARS, coordinate descent, OMP, SOMP, proximal methods
- Solving structured sparse decomposition problems (l_1/l_2 , l_1/linf , sparse group lasso, tree-structured regularization, structured sparsity with overlapping groups,...).

It is developed and maintained by [Julien Mairal](#) (Inria), and contains sparse estimation methods resulting from collaborations with various people: notably, [Francis Bach](#), [Jean Ponce](#), Guillermo Sapiro, [Rodolphe Jenatton](#) and [Guillaume Obozinski](#).

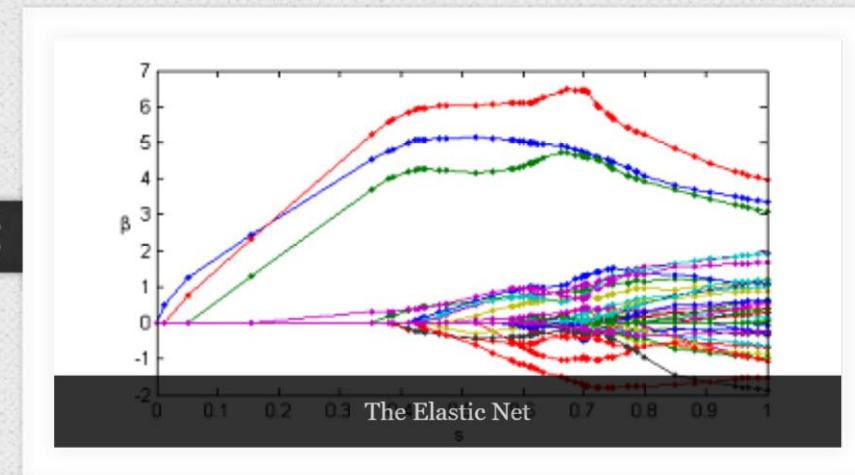
It is coded in C++ with a Matlab interface. Interfaces for R and Python have been developed by Jean-Paul Chieze, and archetypal analysis was written by Yuansi Chen. Release of version 2.6/2.6.1 and porting to R-3.x and Python3.x was done by [Ghislain Durif](#) (Inria). Version 2.6.2 (Python only) modifications were proposed by [François Rheault](#) and [Samuel Saint-](#)

<https://thoth.inrialpes.fr/people/mairal/spams/>

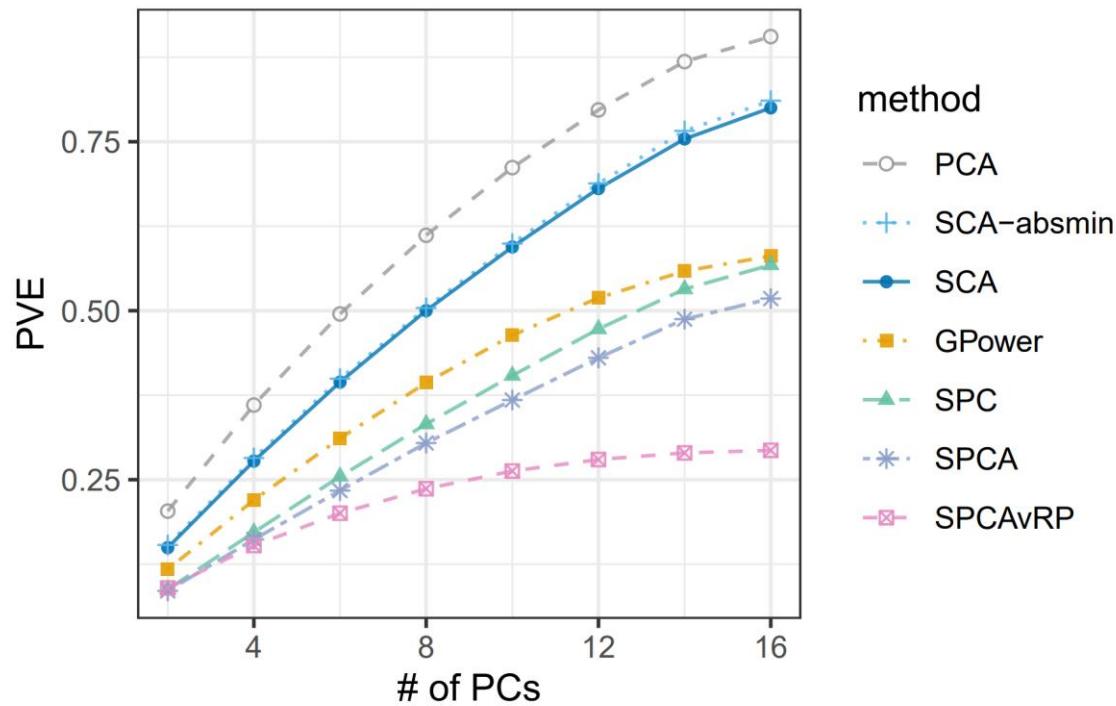
SpaSM

SpaSM is a Matlab toolbox for performing sparse regression, classification and principal component analysis. The toolbox has been developed at the Department of Informatics at the Technical University of Denmark. Development started in 2004 and the toolbox receives regular updates.

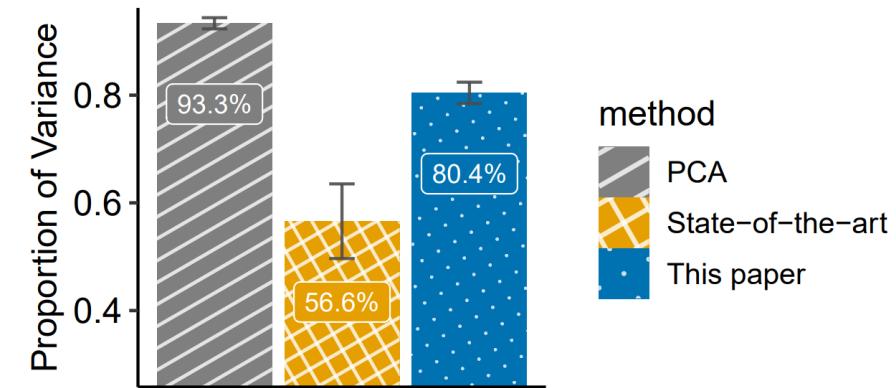
The code is well documented and consists of a series of pure Matlab functions. Examples, test cases and utility functions are also included. The algorithms are based on the regularization path-following paradigm developed mainly at Stanford University (see publication list below).



Variance-Sparsity trade-off



By allowing for a rotated basis, sparse PCA can explain nearly as much variance as the ordinary PCA.



Parameter Space

Algorithm 1. General SPCA Algorithm

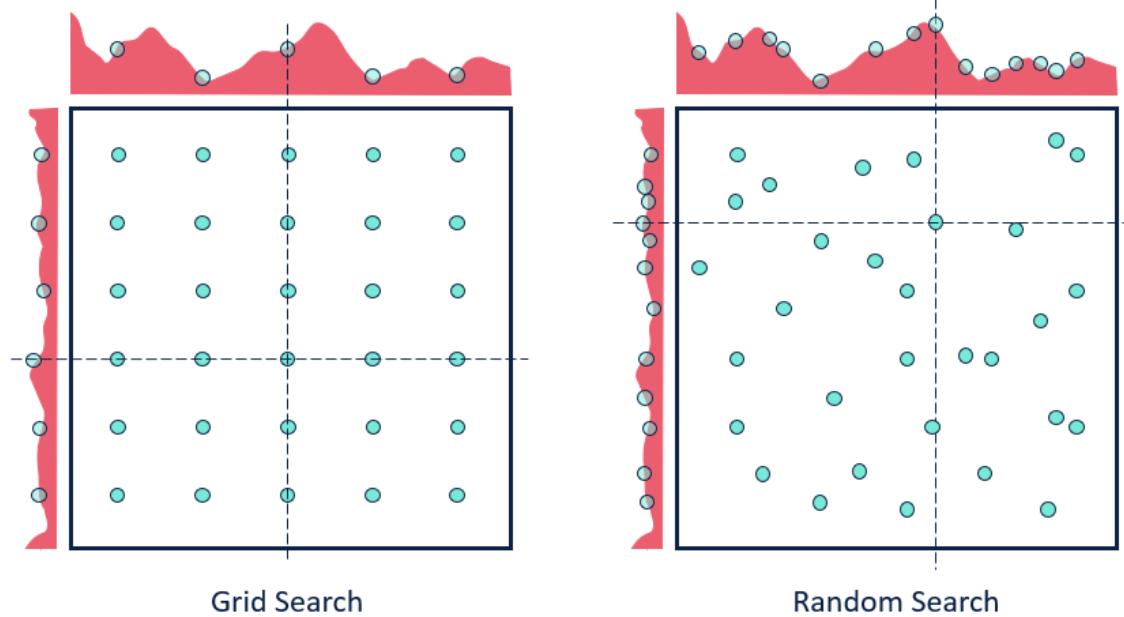
1. Let \mathbf{A} start at $\mathbf{V}[, 1 : k]$, the loadings of the first k ordinary principal components.
2. Given a fixed $\mathbf{A} = [\alpha_1, \dots, \alpha_k]$, solve the following elastic net problem for $j = 1, 2, \dots, k$

$$\beta_j = \arg \min_{\beta} (\alpha_j - \beta)^T \mathbf{X}^T \mathbf{X} (\alpha_j - \beta) + \lambda \|\beta\|^2 + \lambda_{1,j} \|\beta\|_1$$

3. For a fixed $\mathbf{B} = [\beta_1, \dots, \beta_k]$, compute the SVD of $\mathbf{X}^T \mathbf{X} \mathbf{B} = \mathbf{U} \mathbf{D} \mathbf{V}^T$, then update $\mathbf{A} = \mathbf{U} \mathbf{V}^T$.
4. Repeat Steps 2–3, until convergence.
5. Normalization: $\hat{V}_j = \frac{\beta_j}{\|\beta_j\|}, j = 1, \dots, k$.

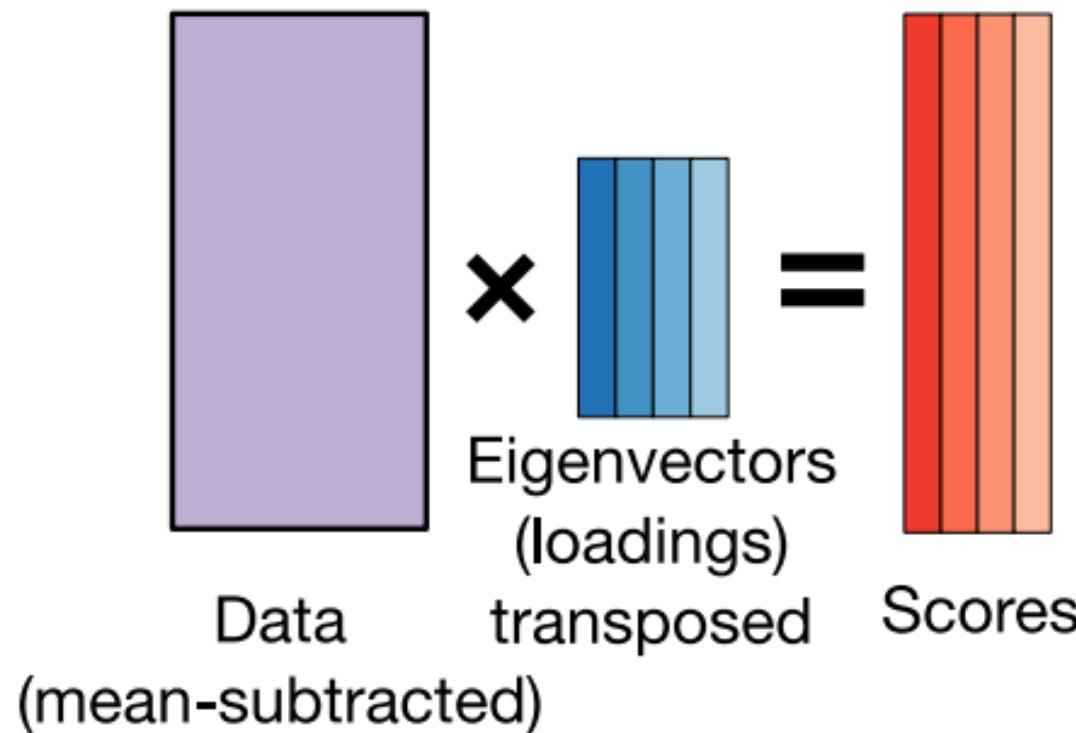
Parameter Tuning

In machine learning, parameter optimization or tuning is the problem of choosing a set of optimal parameters for a learning algorithm. A parameter is a value used to control the learning process.



PCA as a regression problem

$$\hat{\mathbf{y}} = \hat{\beta}_0 + \mathbf{x}_1 \hat{\beta}_1 + \dots + \mathbf{x}_p \hat{\beta}_p$$

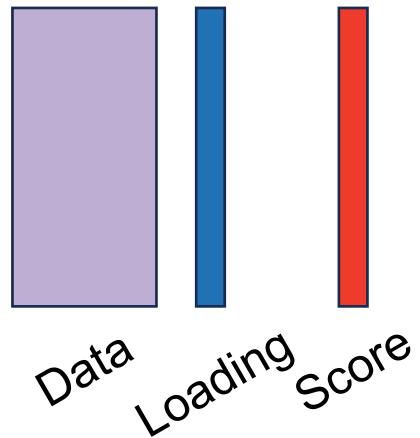


Regularization and variable selection via the elastic net Tuning [LARS-EN]

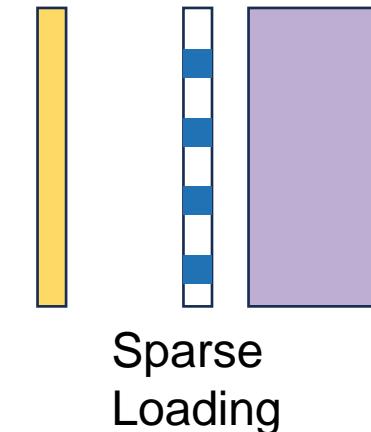
$$\hat{\mathbf{y}} = \hat{\beta}_0 + \mathbf{x}_1 \hat{\beta}_1 + \dots + \mathbf{x}_p \hat{\beta}_p$$

$$L(\lambda_1, \lambda_2, \boldsymbol{\beta}) = |\mathbf{y} - \mathbf{X}\boldsymbol{\beta}|^2 + \lambda_2 |\boldsymbol{\beta}|^2 + \lambda_1 |\boldsymbol{\beta}|_1$$

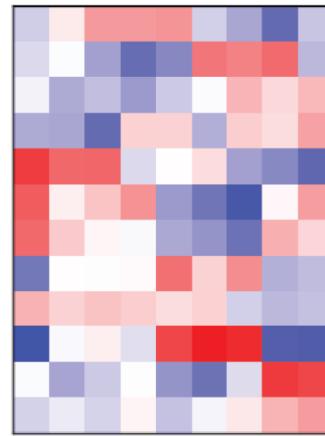
PCA



Sparse
PCA

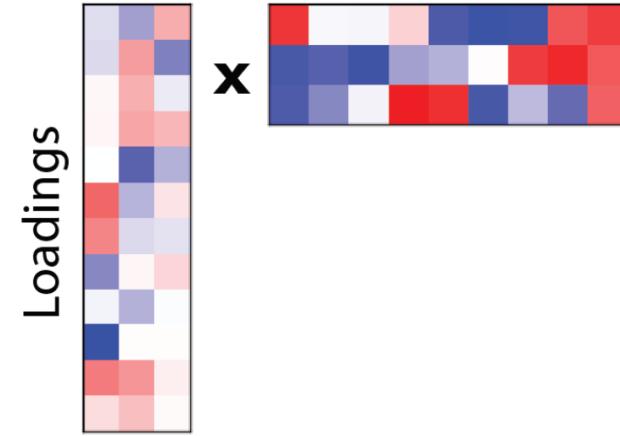


Original Data

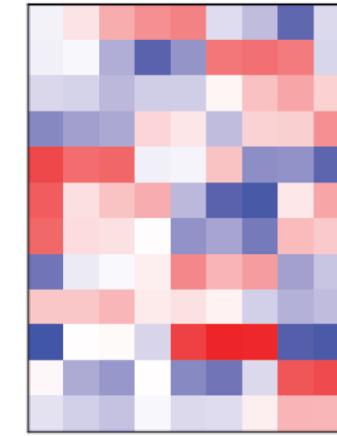


\approx

Components

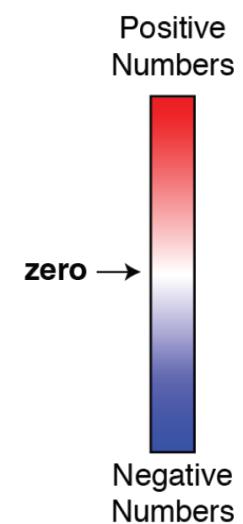
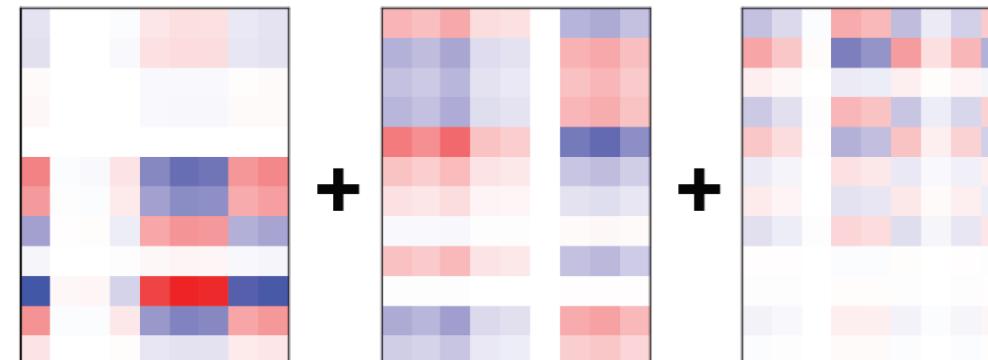


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax



neurosyntax

Session 7

Lecturer: Saman Abbaspoor

Nonnegative Matrix Factorization

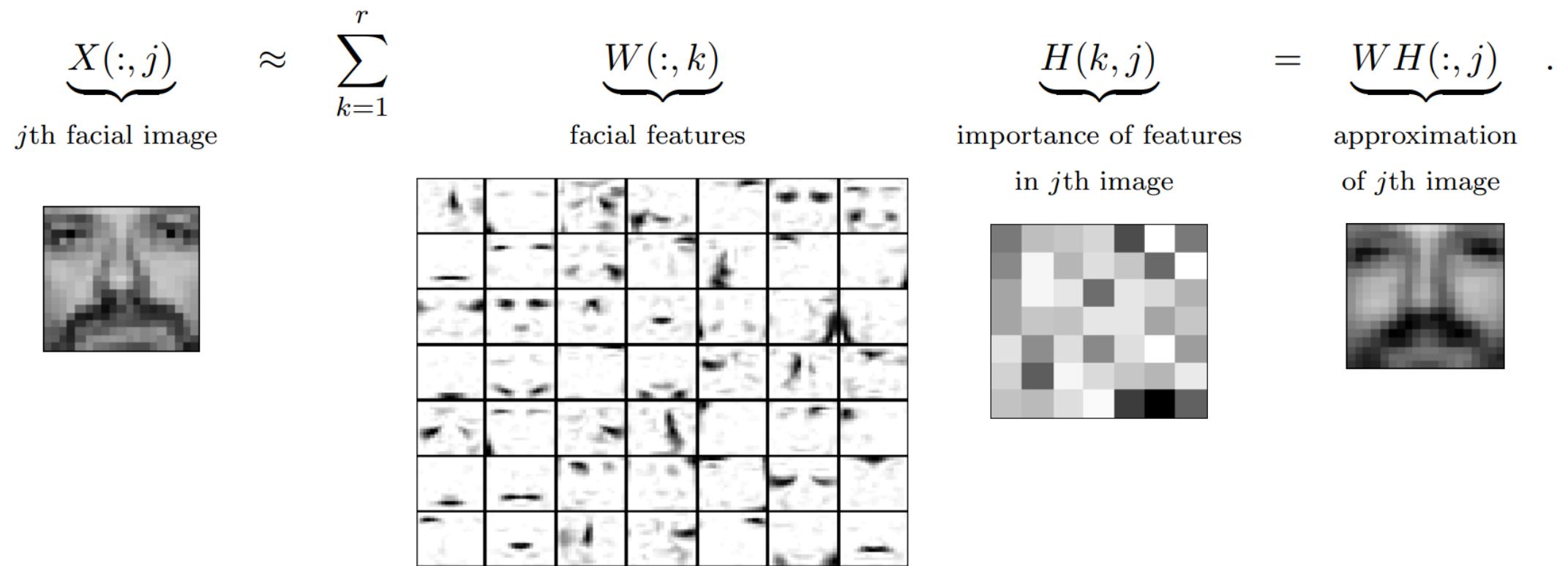
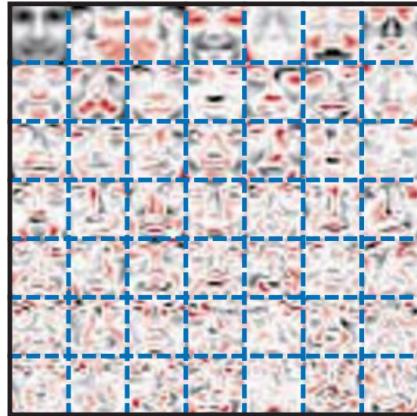


Figure 1: Decomposition of the CBCL face database, MIT Center For Biological and Computation Learning (2429 gray-level 19-by-19 pixels images) using $r = 49$ as in [79].

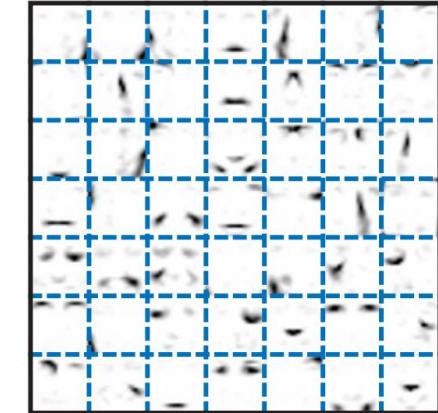
Nonnegative Matrix Factorization (NMF)

Non-negative matrix factorization (NMF) learns a parts-based representation of data, whereas principal components analysis (PCA) learn holistic representations.

PCA



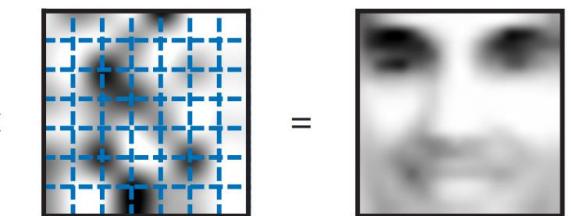
NMF



Original



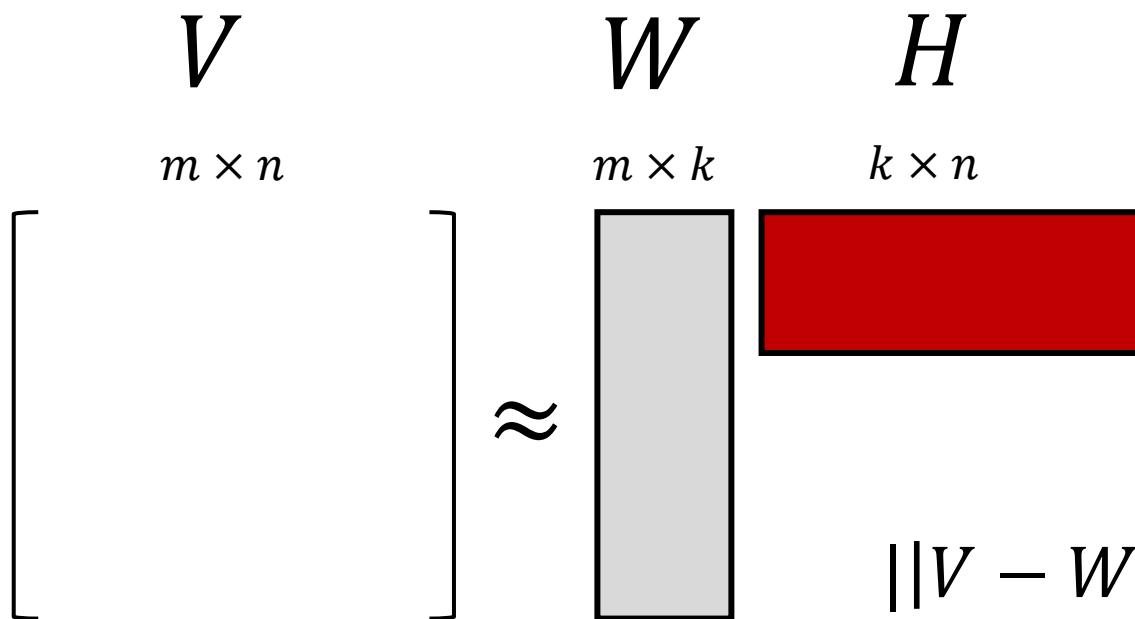
=



Nonnegative Matrix Factorization (NMF)

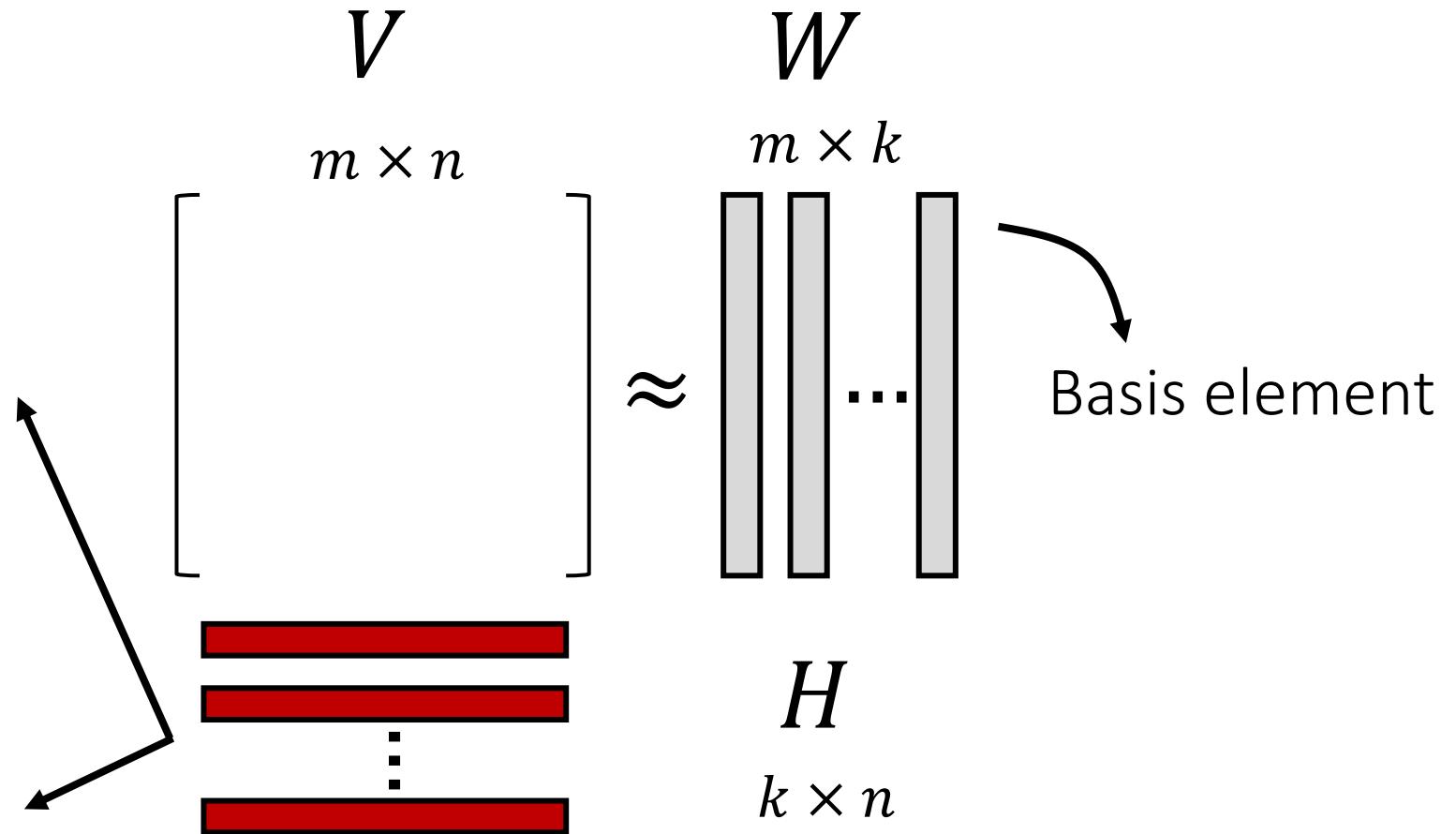
NMF approximates a matrix V with a **low-rank matrix approximation**
 $V \approx WH$

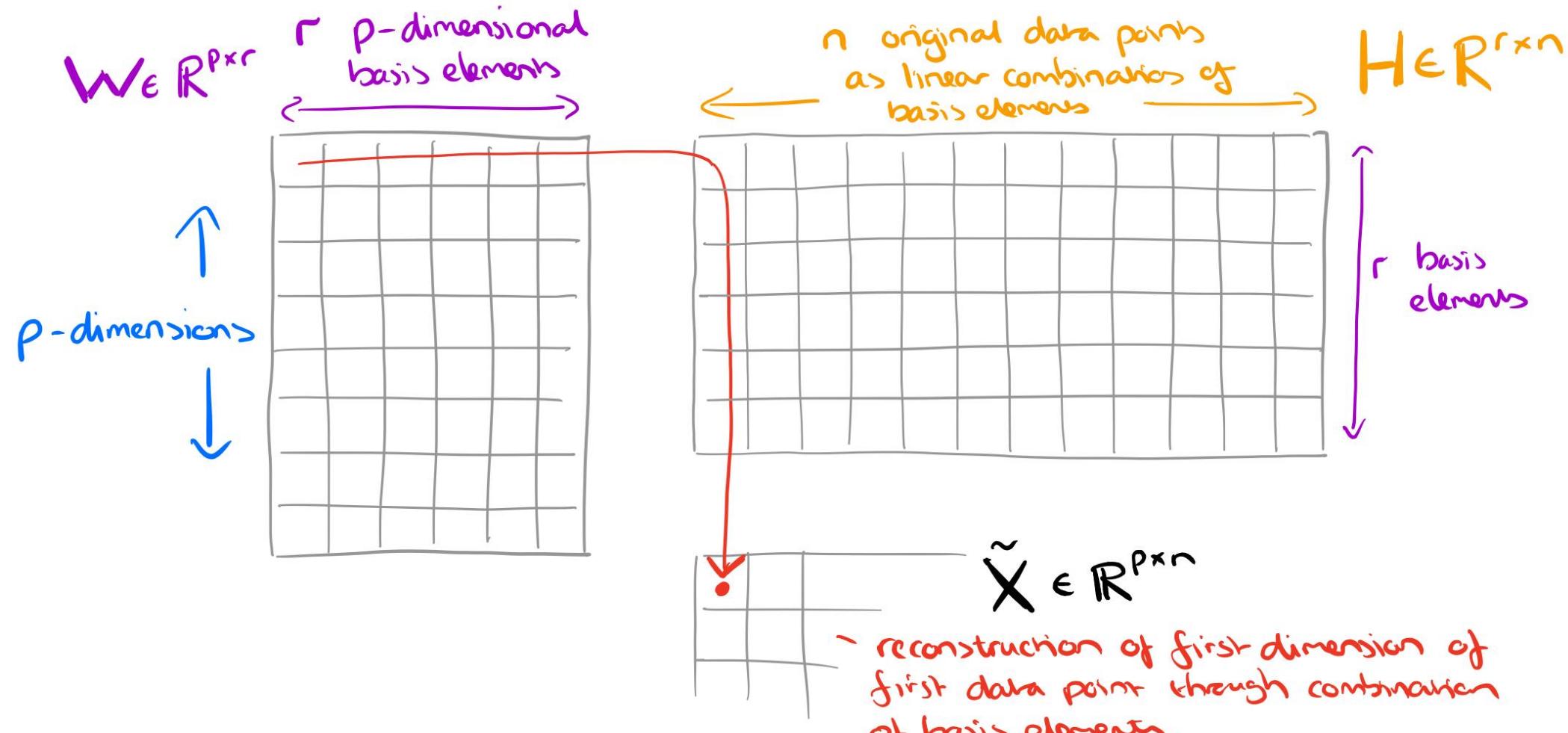
It automatically extracts **sparse** and meaningful features from a set of nonnegative data vectors.

$$\begin{matrix} V \\ m \times n \end{matrix} \quad \begin{matrix} W \\ m \times k \end{matrix} \quad \begin{matrix} H \\ k \times n \end{matrix}$$

$$V \approx WH; v_i \approx Wh_i$$
$$||V - WH||_F, \text{ subject to } W \geq 0, H \geq 0$$

Reconstruct an approximation to the original data point from a linear combination of the building blocks in W

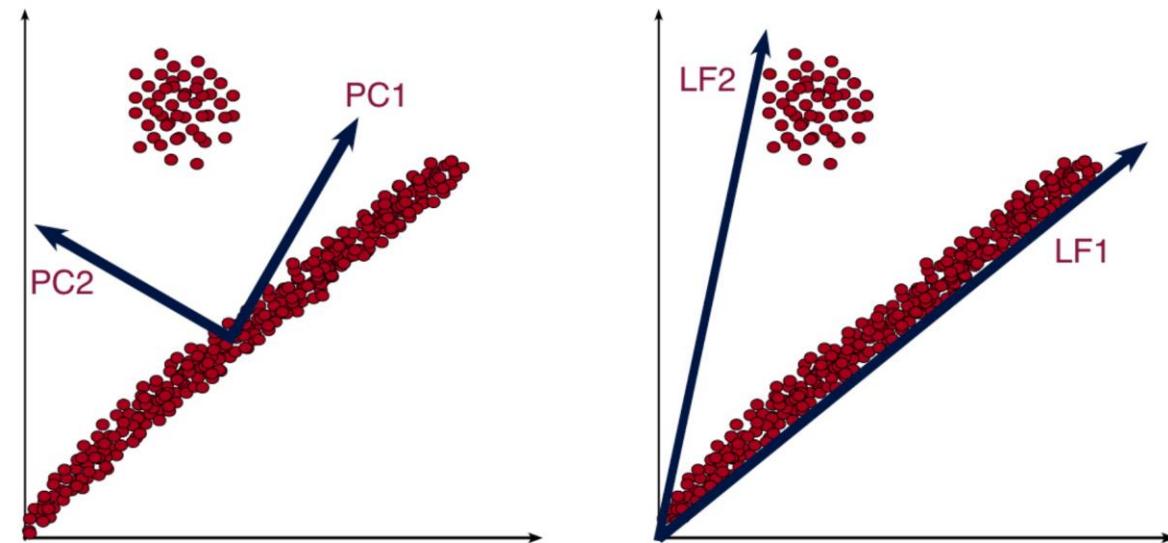
The ‘coordinates of a data point’ in the basis W



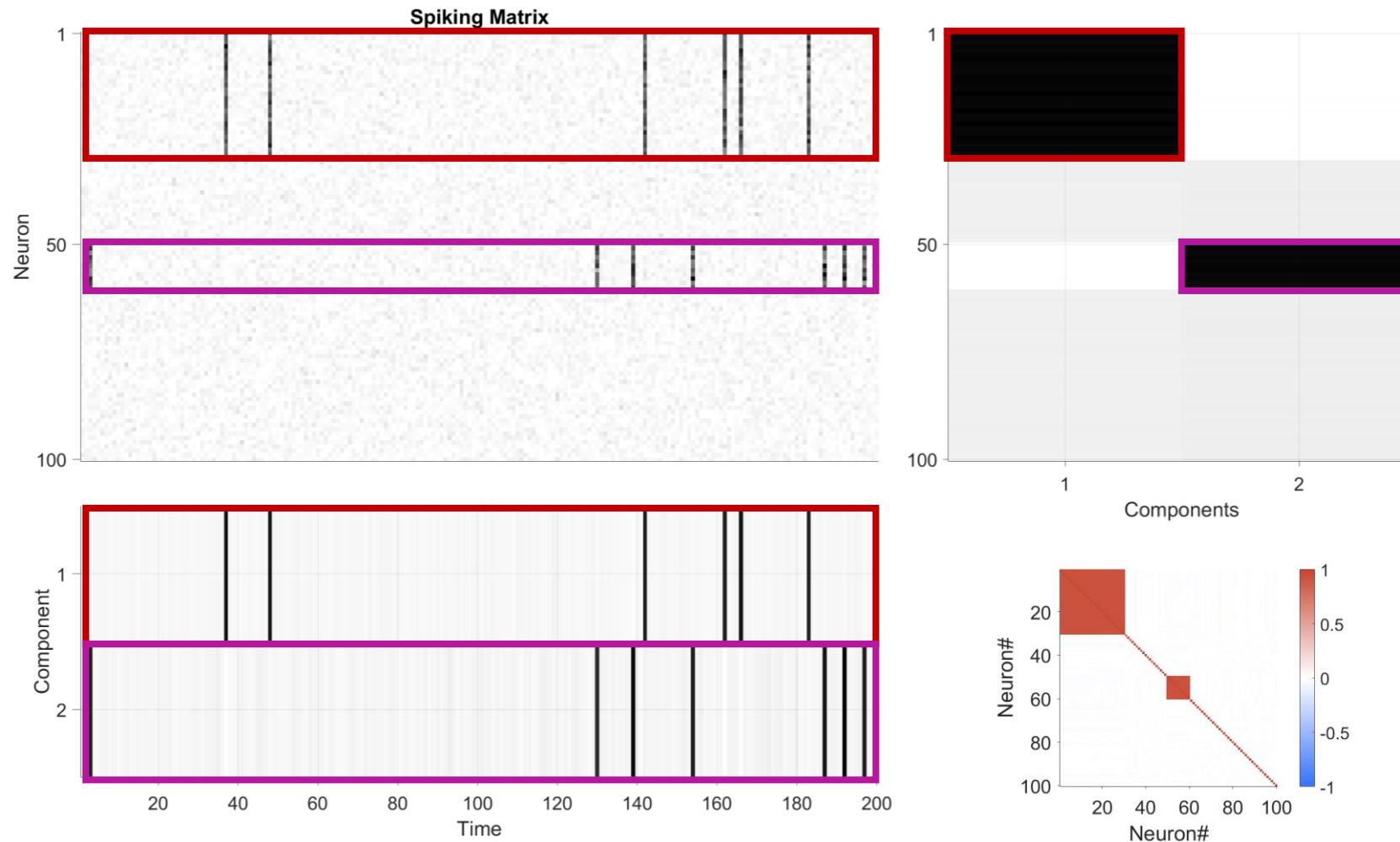


Nonnegative Matrix Factorization (NMF)

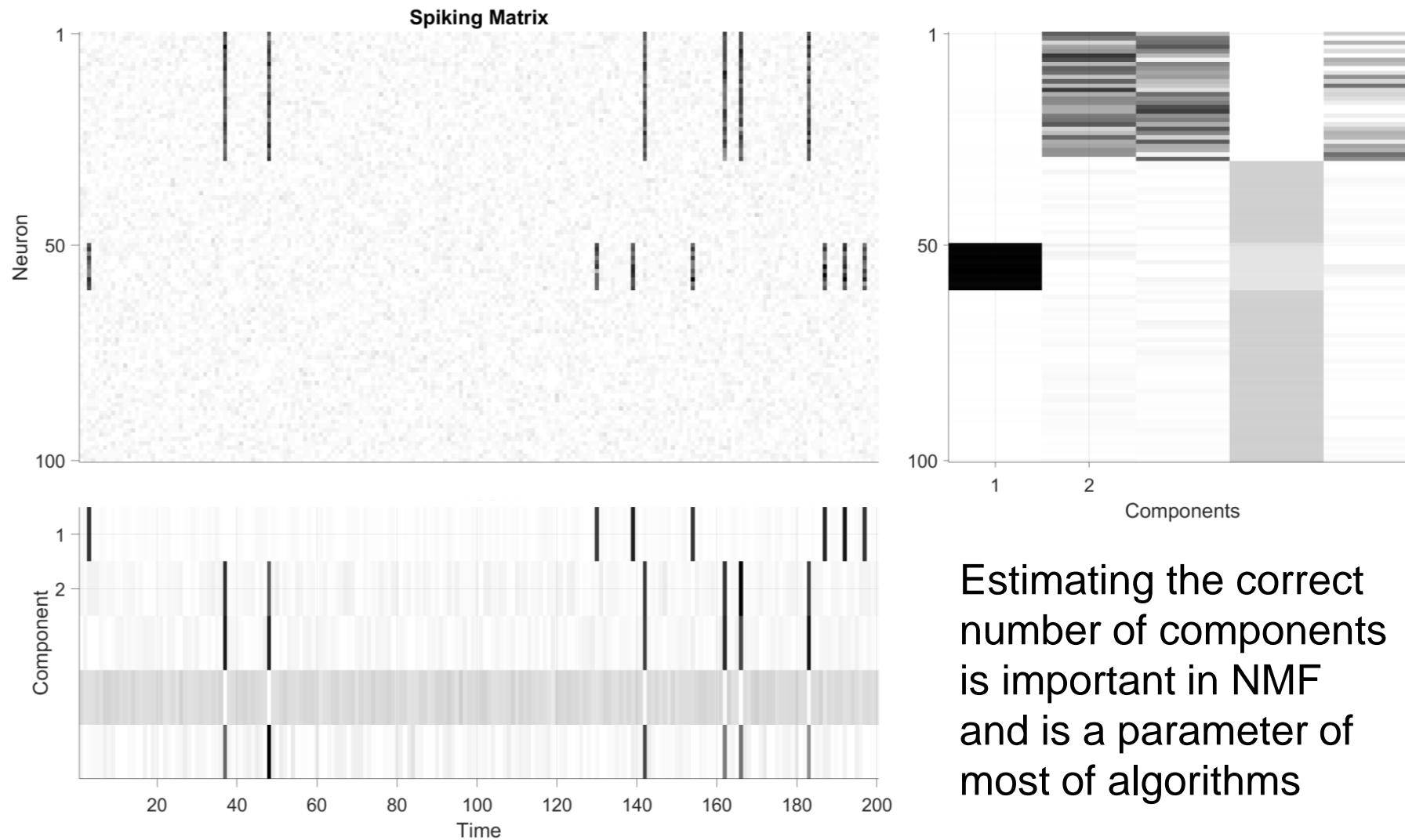
Nonnegative Matrix Factorization (NMF) does not inherently have the assumption of orthogonality.



Timeseries Patterns and Nonnegative Matrix Factorization (NMF)

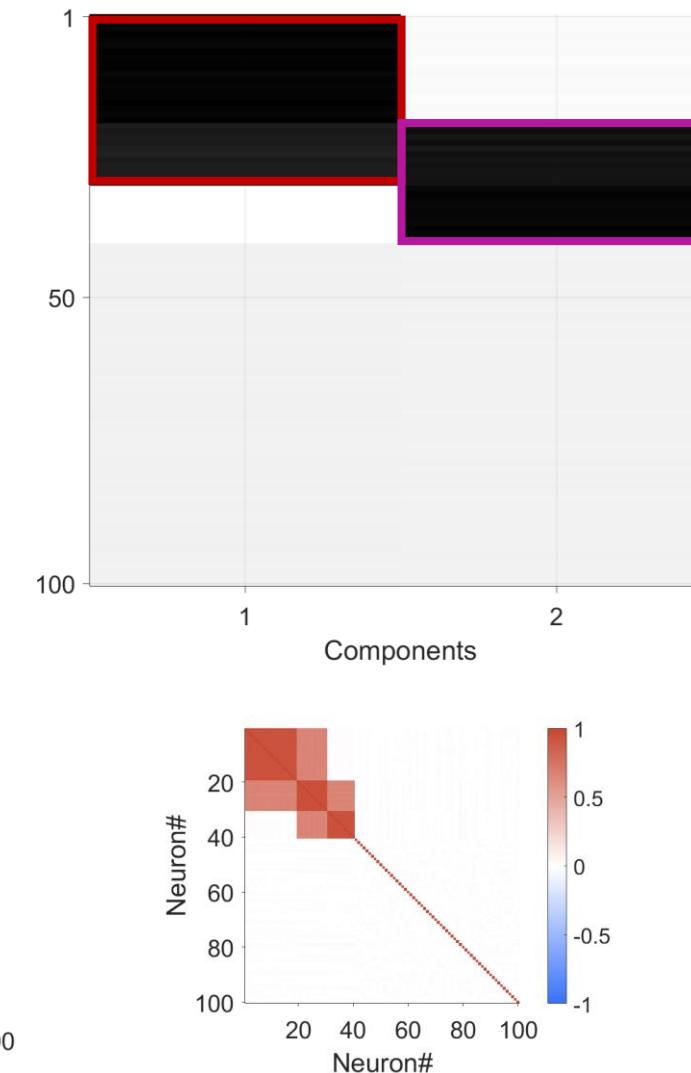
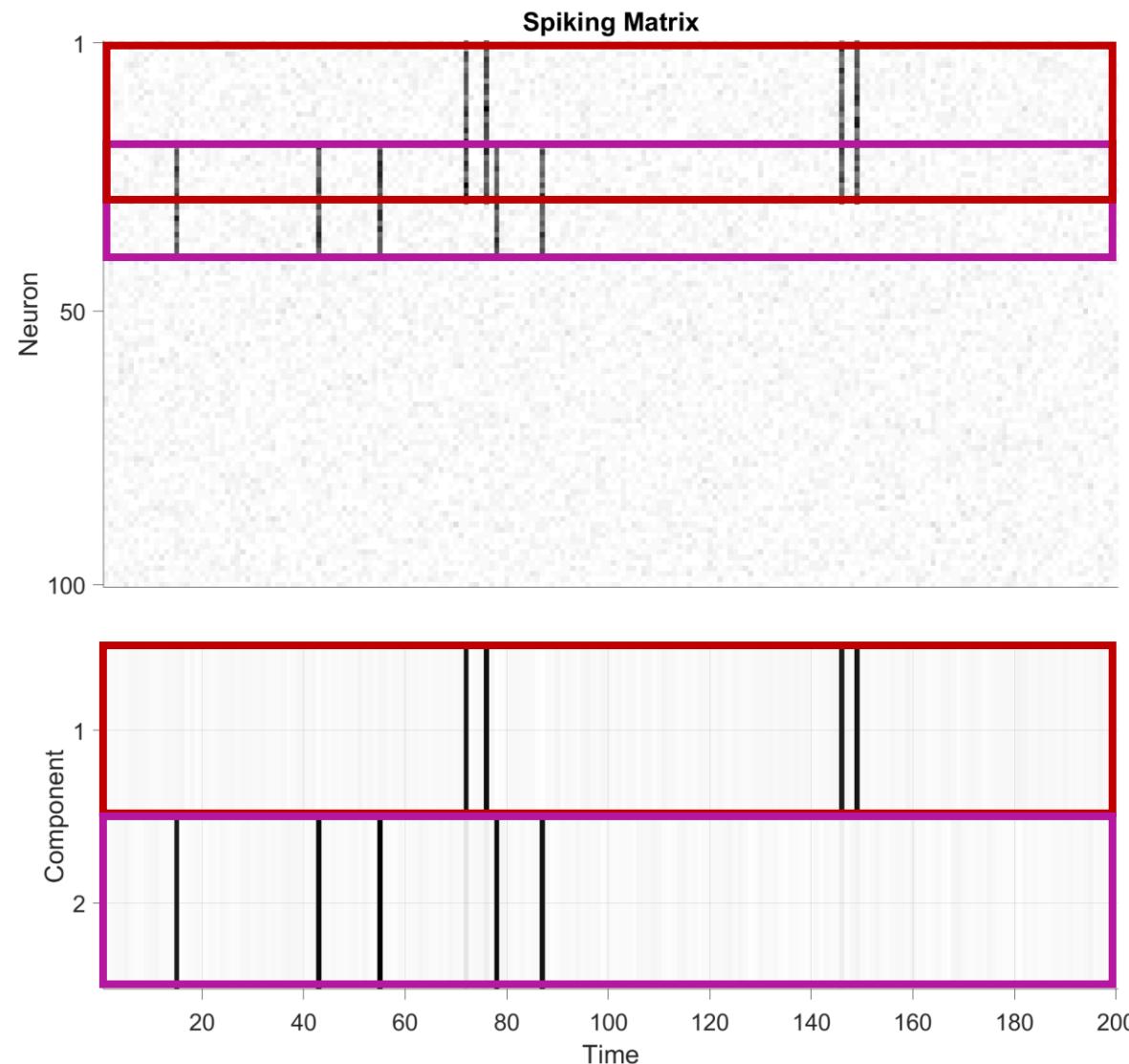


Timeseries Patterns and Nonnegative Matrix Factorization (NMF)

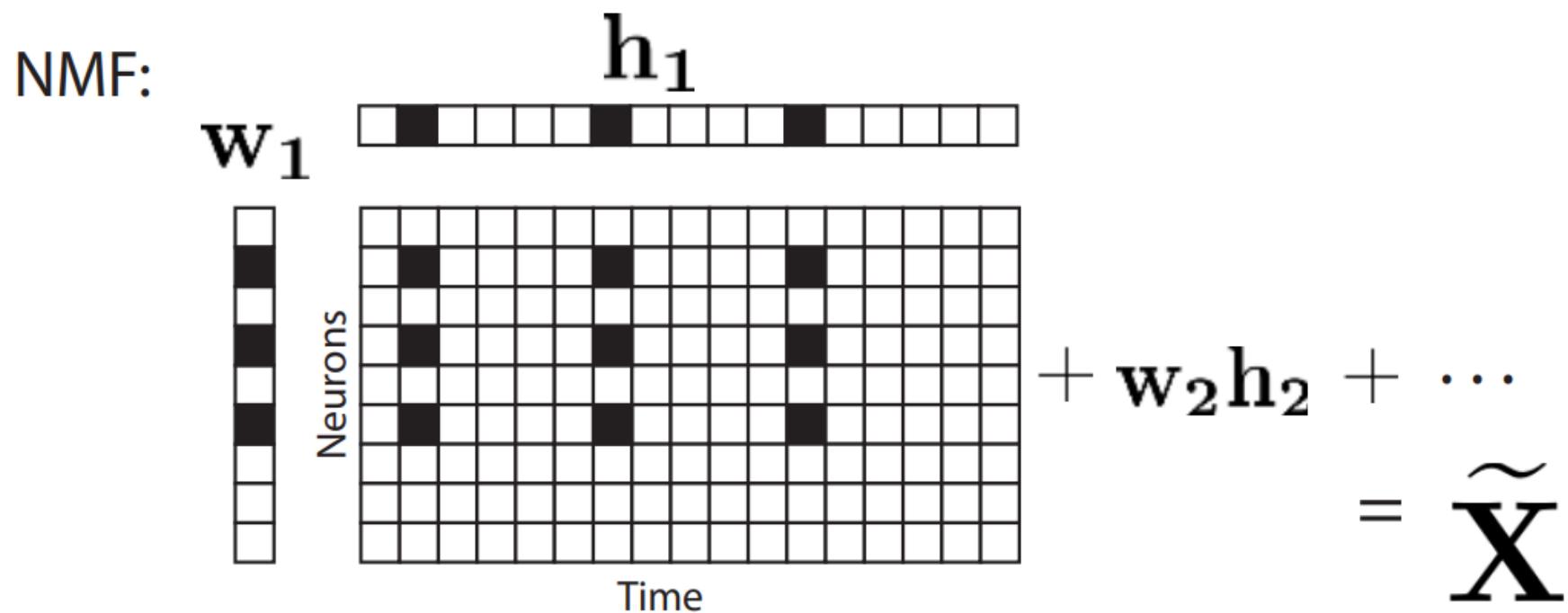


Estimating the correct number of components is important in NMF and is a parameter of most of algorithms

Timeseries Patterns and Nonnegative Matrix Factorization (NMF)



Nonnegative Matrix Factorization (NMF)



NMFLibrary: Non-negative Matrix Factorization Library

Authors: [Hiroyuki Kasai](#)

Last page update: July 22, 2022

Latest library version: 2.1 (see [Release notes](#) for more info)

Announcement

We are very welcome to your contribution. Please tell us

- NMF solvers written by MATLAB,
- application MATLAB files using NMF solvers, and
- your comments and suggestions.

Syntax

```
[W,H] = nnmf(A,k)
[W,H] = nnmf(A,k,Name,Value)
[W,H,D] = nnmf( __ )
```

Description

[W,H] = nnmf(A,k) factors the n -by- m matrix A into nonnegative factors W (n -by- k) and H (k -by- m). The factorization is not exact; W*H is a lower-rank approximation to A. The factors W and H minimize the root mean square residual D between A and W*H.

[example](#)

```
D = norm(A - W*H, 'fro')/sqrt(n*m)
```

The factorization uses an iterative algorithm starting with random initial values for W and H. Because the root mean square residual D might have local minima, repeated factorizations might yield different W and H. Sometimes the algorithm converges to a solution of lower rank than k , which can indicate that the result is not optimal.

[W,H] = nnmf(A,k,Name,Value) modifies the factorization using one or more name-value pair arguments. For example, you can request repeated factorizations by setting 'Replicates' to an integer value greater than 1.

[example](#)



Assumptions of NNMF

Non-negativity: The most fundamental assumption of NMF is that the data represented in the matrix V and the matrices W and H it factors into must be non-negative. This constraint leads to a parts-based representation because only additive, not subtractive, combinations are allowed.

Additivity: NMF assumes that the features extracted in the components matrix W combine additively to form the original dataset. This is suitable for data where the features are often additive (e.g., pixels in images or topics in documents).

Assumptions of NNMF

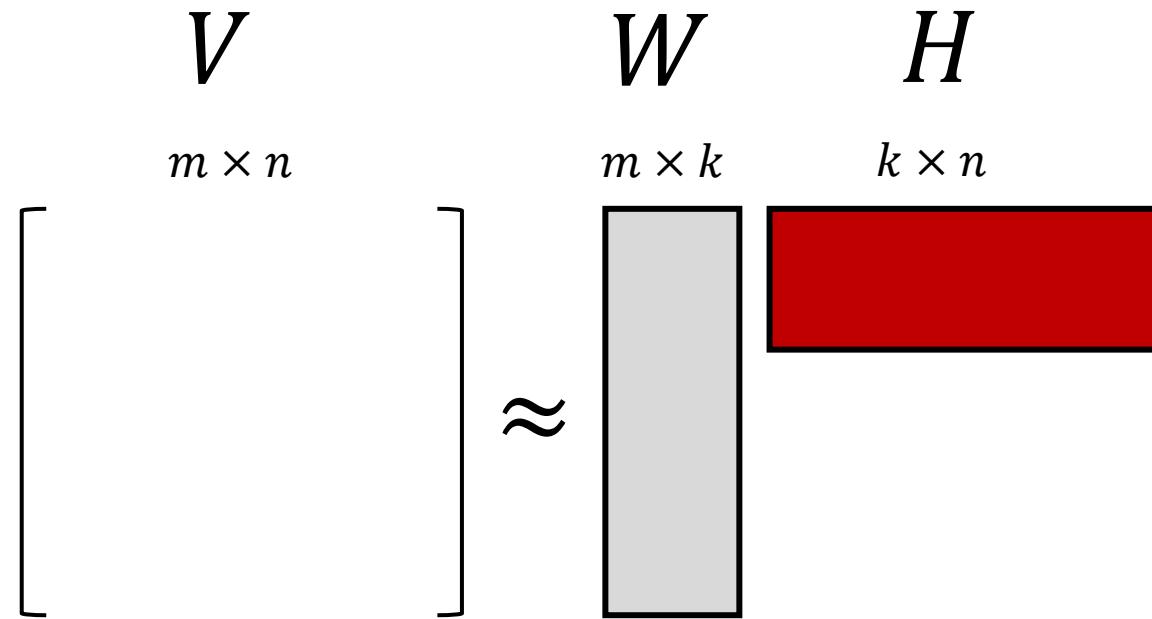
Independence and Sparsity: NMF often works best when the underlying factors are somewhat independent or sparse. In practice, this means that the features represented by W should not overlap too much, and many entries in W and H are zero or near-zero, leading to sparse representations.

Stationarity: NMF assumes that the data does not change its underlying structure over time, which makes it less suitable for time-series data where underlying factors might change.

Linearity: The model assumes a linear relationship between the components in W and the reconstructions in V . This might not hold for complex datasets where interactions between components are non-linear.

NNMF

Implementation

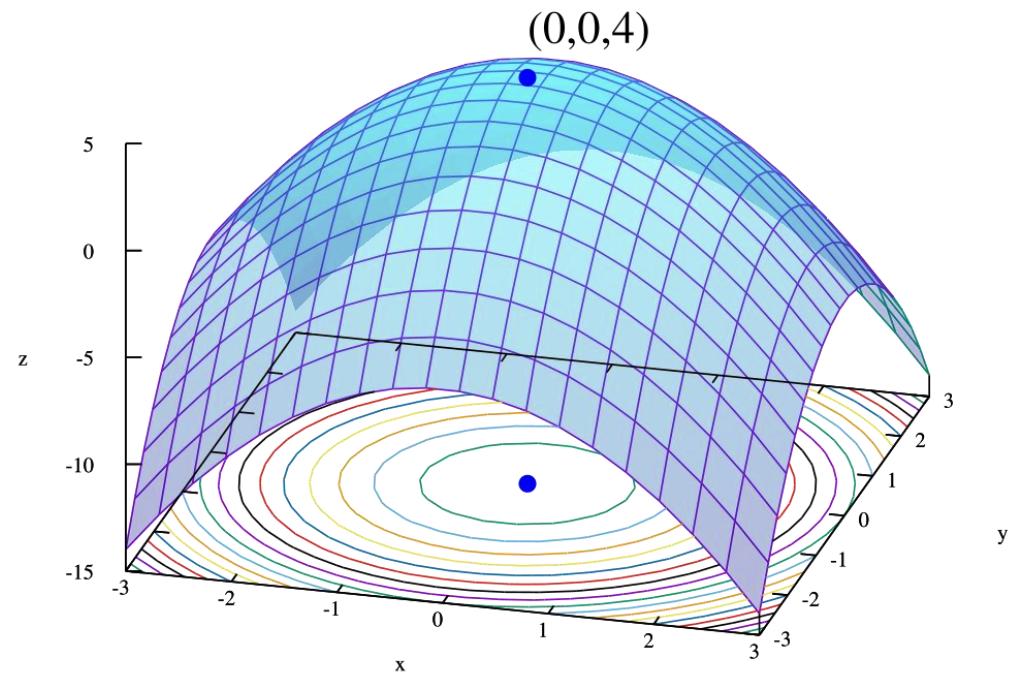


$$\min_{W \in \mathbb{R}^{p \times r}, H \in \mathbb{R}^{r \times n}} \|X - WH\|_F^2 \quad \text{subject to} \quad W \geq 0, H \geq 0$$

Optimization

In the more general approach, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function.

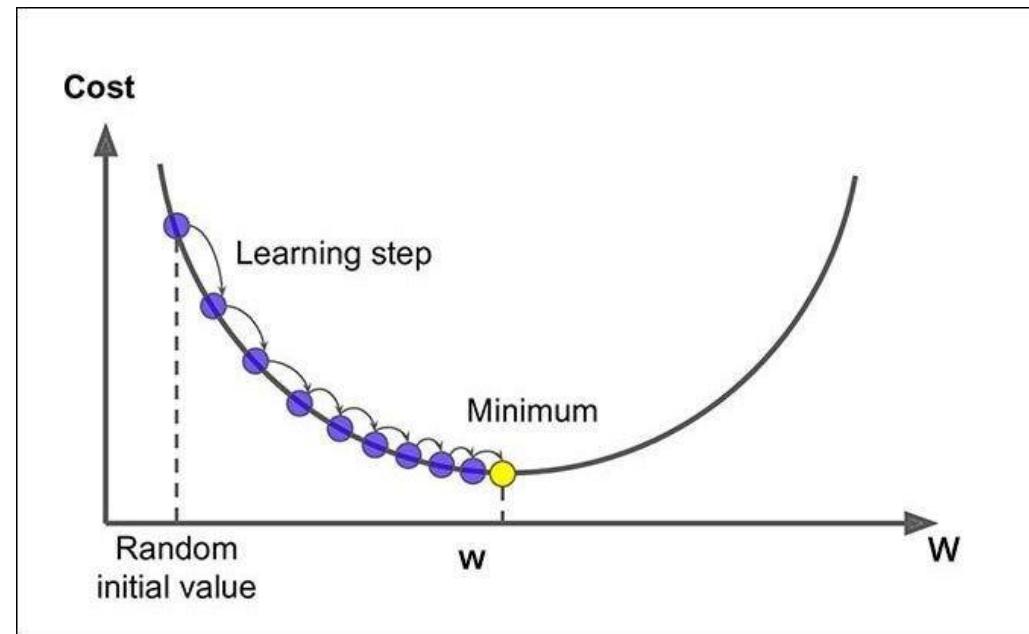
Optimization is the selection of a best element, with regard to some criteria, from some set of available alternatives.



Optimization

In the more general approach, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function.

Optimization is the selection of a best element, with regard to some criteria, from some set of available alternatives.



Key Concepts in Optimization

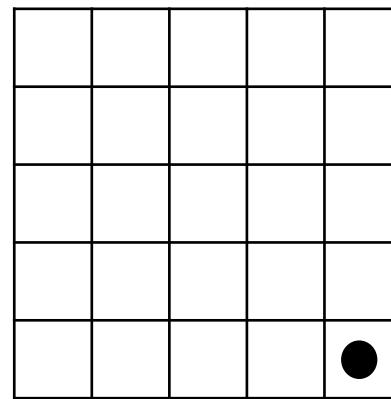
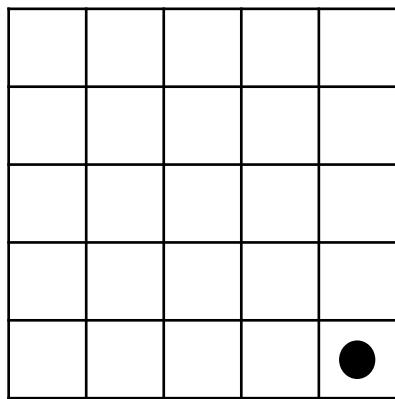
Objective/Cost Function:

This is the function that needs to be minimized or maximized. It represents a measure of "goodness" or "cost" that needs to be optimized.

$$\min_{W \in \mathbb{R}^{p \times r}, H \in \mathbb{R}^{r \times n}} \|X - WH\|_F^2$$

Frobenius norm

$$\|X - WH\|_F^2 = \sum_{i,j} (X - WH)_{ij}^2$$



$$(X - WH)_{ij}^2$$

Frobenius norm and SVD

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$$

$$\|A\|_F = \|\Sigma\|_F = \sqrt{\sum_{i=1}^r \sigma_i^2}$$

$$\|A\|_F = \|U\Sigma V^T\|_F$$

$$\|U\Sigma V^T\|_F = \|\Sigma\|_F$$

$$\|\Sigma\|_F = \sqrt{\sum_{i=1}^r \sigma_i^2}$$

Using the property that the Frobenius norm is invariant under multiplication by orthogonal matrices (U and V are orthogonal), we get:

Frobenius norm and SVD

$$\|A\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_r^2}$$

Key Concepts in Optimization

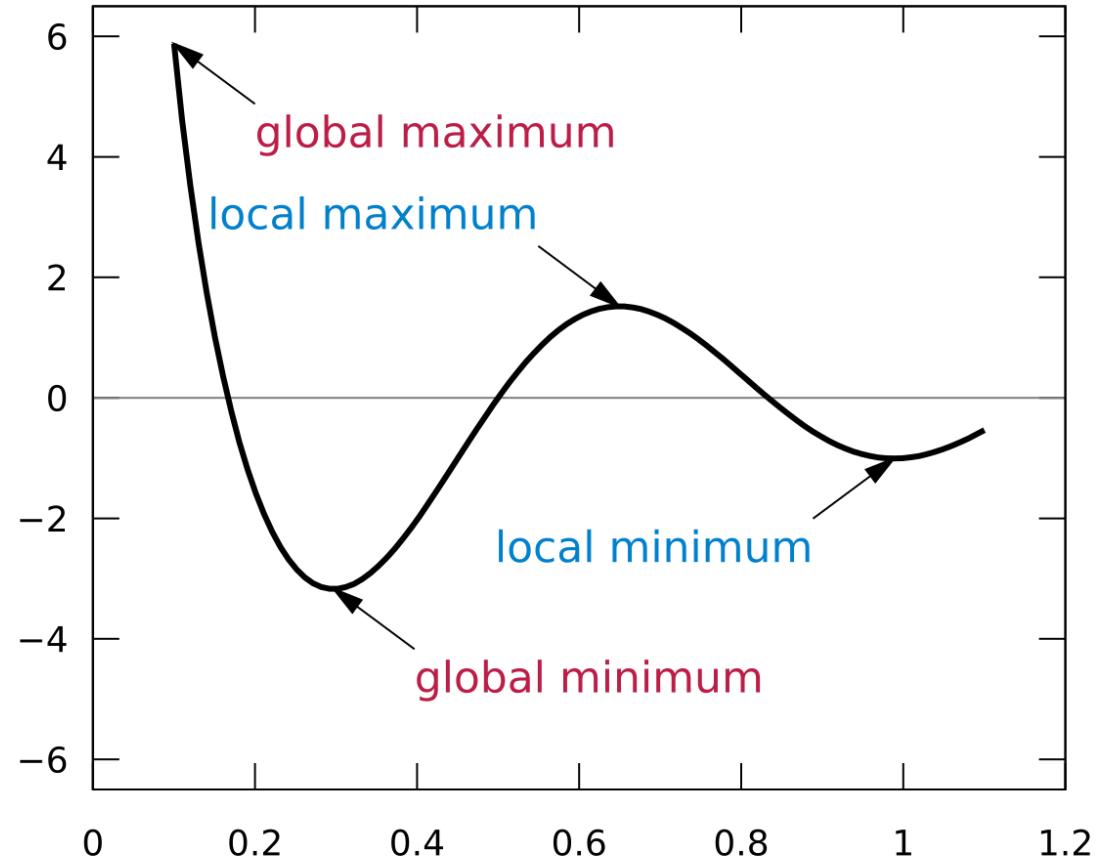
Constraints: These are the limitations or requirements that the solution must satisfy.

$$\min_{W \in \mathbb{R}^{p \times r}, H \in \mathbb{R}^{r \times n}} \|X - WH\|_F^2$$

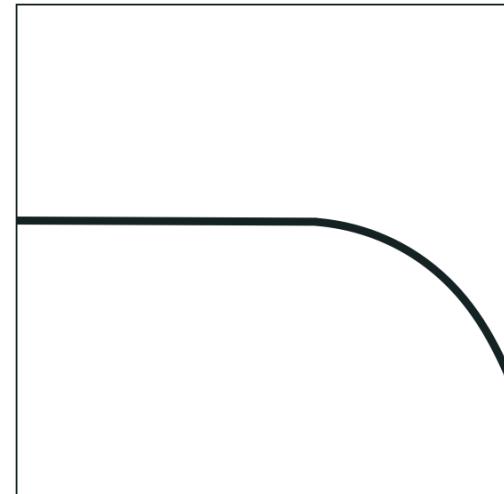
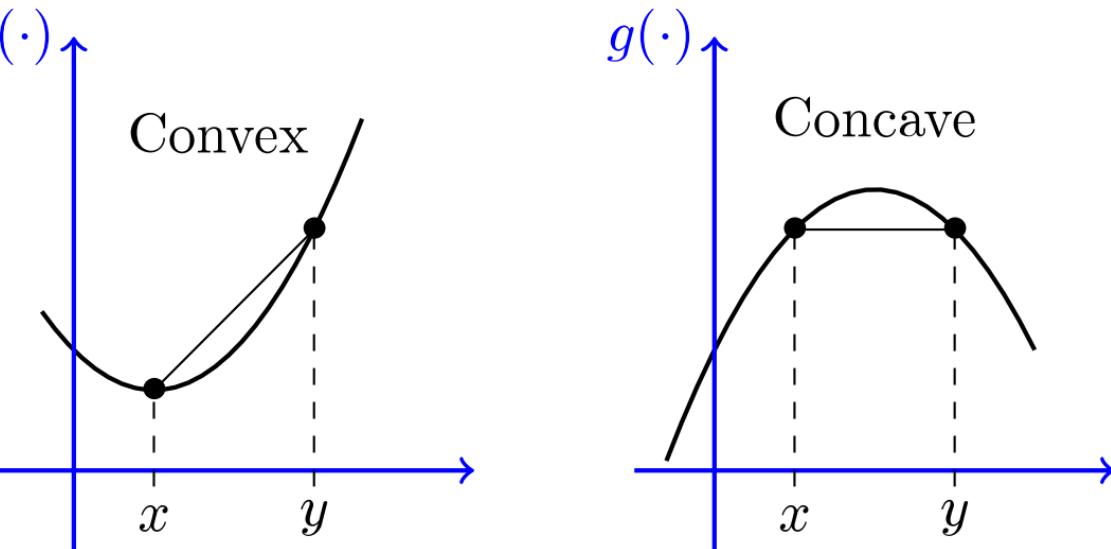
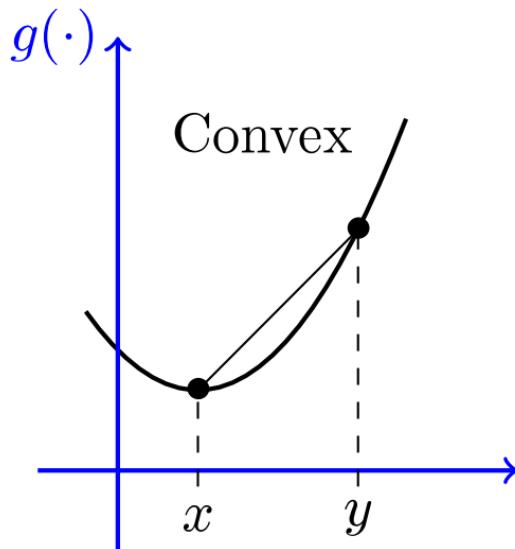
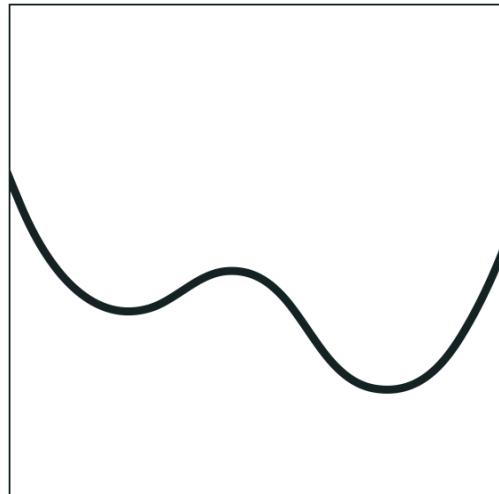
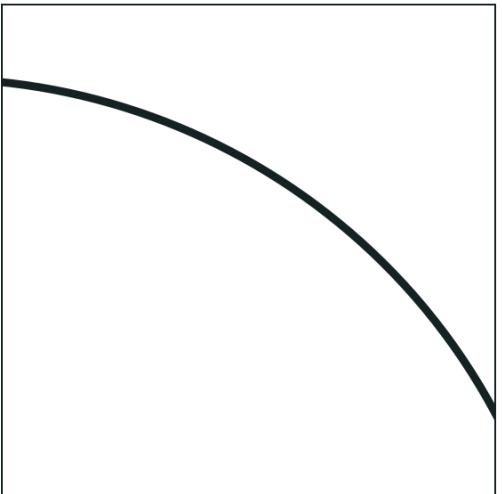
subject to $W \geq 0, H \geq 0$

Key Concepts in Optimization

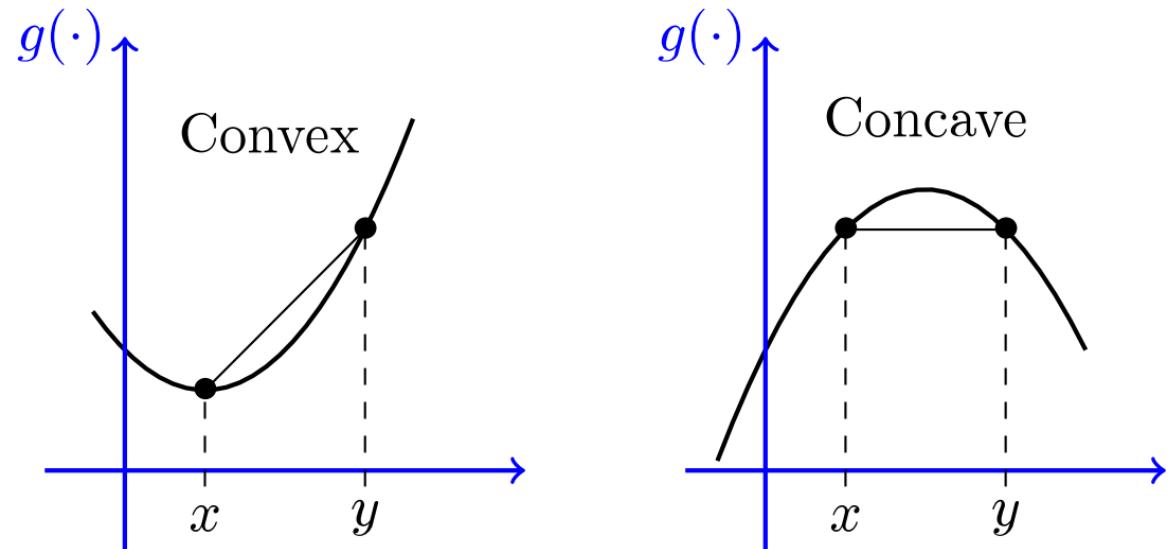
Global vs. Local Optima: In optimization, a global optimum is the best possible solution out of all possible solutions, while local optima are the best solutions within a neighborhood of solutions. For many problems, finding the global optimum can be challenging, especially in non-convex optimization landscapes.



Concave and Convex functions



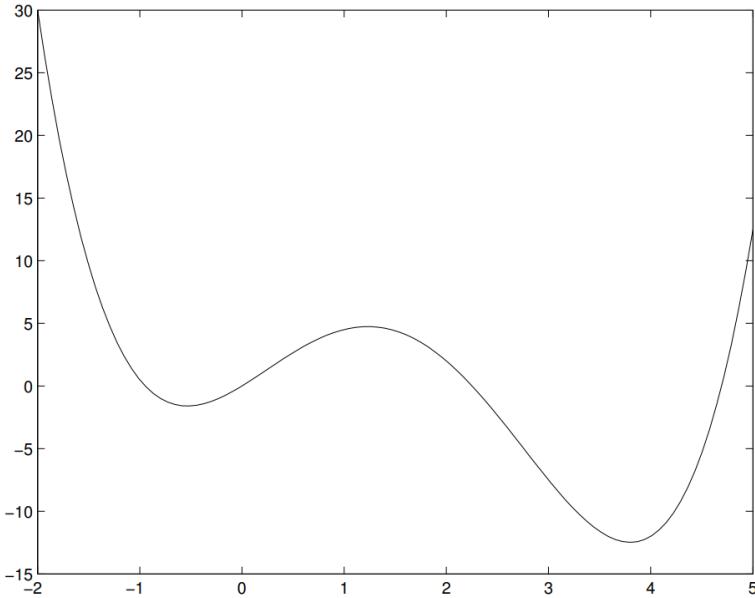
Concave and Convex functions



Convex functions are particularly important because they have a unique global minimum. This means that if we want to optimize a convex function, we can be sure that we will always find the best solution by searching for the minimum value of the function. This makes optimization easier and more reliable.

Caveats

- The NMF problem is **non-convex**.



- The algorithm is only guaranteed to find a local optimum.
- The algorithm is sensitive to choice of initialization.



Almost all NMF algorithms use a two-block coordinate descent scheme (exact or inexact), that is, they optimize alternatively over one of the two factors, W or H , while keeping the other fixed. The reason is that the subproblem in one factor is convex. More precisely, it is a nonnegative least squares problem (NNLS). Many algorithms exist to solve the NNLS problem; and NMF algorithms based on two-block coordinate descent differ by which NNLS algorithm is used.

Limitations of NNMF

Requirement for Non-negativity: Since NMF requires non-negative data, it cannot be directly applied to data that contains negative values without preprocessing (e.g., shifting all values to be positive).

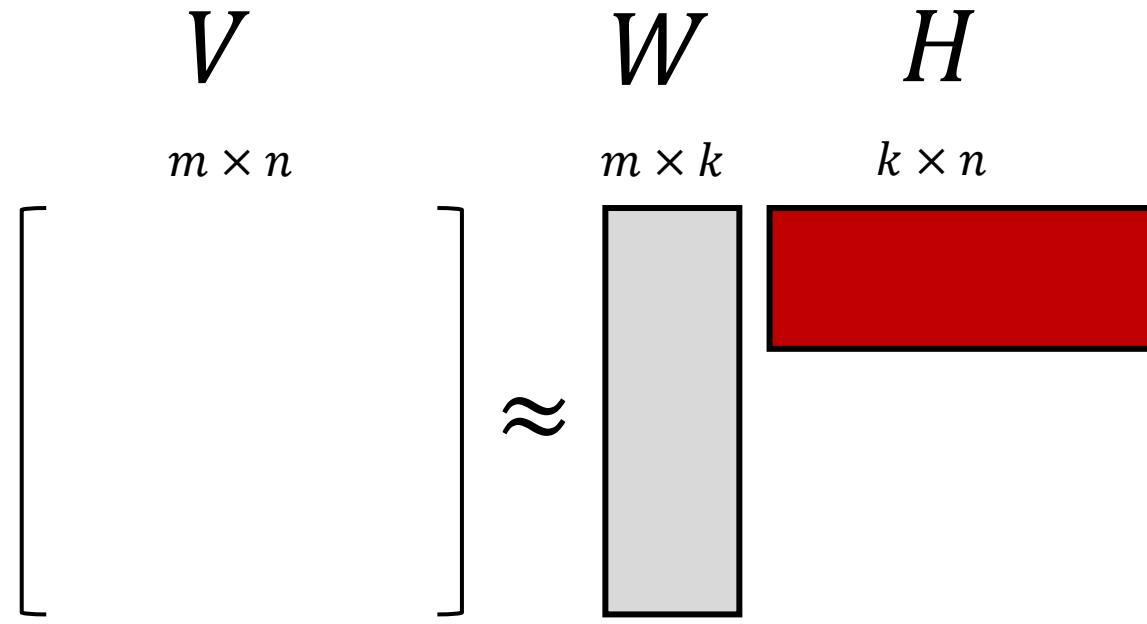
Determining the Rank: Choosing the number of components k (the rank of matrices W and H) is not straightforward and typically requires domain knowledge or experimentation. Incorrect choices of k can lead to overfitting or underfitting.

Local Minima: NMF involves non-convex optimization problems (typically solved using iterative methods like alternating least squares or gradient descent), which can converge to local, rather than global, minima. The initialization of W and H can significantly affect the results.

Limitations of NNMF

Scalability: NMF computations can be intensive, particularly for large datasets, although there have been developments in more scalable algorithms.

Absence of Consensus on Evaluation Metrics: There is no universally agreed-upon way to evaluate the quality of the factorization, particularly in unsupervised learning scenarios. Common practices include using reconstruction error, but this might not always reflect the meaningfulness or usefulness of the extracted features.



$$\min_{W \in \mathbb{R}^{p \times r}, H \in \mathbb{R}^{r \times n}} \|X - WH\|_F^2 \quad \text{subject to} \quad W \geq 0, H \geq 0$$

An important issue with NMF is that there is not guaranteed to be a single unique decomposition (in general, there might be many schemes for defining sets of basis elements).

NMF implementation

$$\min_{W,H \geq 0} D(V||WH)$$

$$D(V||\hat{V}) = \sum_{i,j} (V_{ij} - \hat{V}_{ij})^2$$

$$D(V||\hat{V}) = \sum_{i,j} \left(V_{ij} \log \frac{V_{ij}}{\hat{V}_{ij}} - V_{ij} + \hat{V}_{ij} \right)$$

Algorithms for NMF

Using $D(\mathbf{V} || \mathbf{W} \mathbf{H}) = D(\mathbf{V}^T || \mathbf{H}^T \mathbf{W}^T)$, we obtain a similar update for \mathbf{W} .

Now we just iterate between:

- ① Updating \mathbf{W} .
- ② Updating \mathbf{H} .
- ③ Checking $D(\mathbf{V} || \mathbf{W} \mathbf{H})$. If the change since the last iteration is small, then declare convergence.

The algorithm is summarized below:

Algorithm KL-NMF

initialize \mathbf{W}, \mathbf{H}

repeat

$$\mathbf{H} \leftarrow \mathbf{H} \cdot * \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}^T \mathbf{H}}}{\mathbf{W}^T \mathbf{1}}$$
$$\mathbf{W} \leftarrow \mathbf{W} \cdot * \frac{\mathbf{V} \frac{\mathbf{H}^T}{\mathbf{1} \mathbf{H}^T}}{\mathbf{1}}$$

until convergence **return** \mathbf{W}, \mathbf{H}

NMF implementation

How does one solve

$$\underset{\mathbf{W}, \mathbf{H} \geq 0}{\text{minimize}} \quad \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{W} \mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W} \mathbf{H})_{ij} \right) ?$$

Tricks of the trade for minimizing a function $f(\mathbf{x})$.

- closed-form solutions: solve $\nabla f(\mathbf{x}) = 0$.
- gradient descent: iteratively move in steepest descent dir.

$$\mathbf{x}^{(\ell+1)} \leftarrow \mathbf{x}^{(\ell)} - \eta \nabla f(\mathbf{x}^{(\ell)}).$$

- Newton's method: iteratively minimize quadratic approx.

$$\begin{aligned} \mathbf{x}^{(\ell+1)} &\leftarrow \underset{\mathbf{x}}{\operatorname{argmin}} \quad f(\mathbf{x}^{(\ell)}) + \nabla f(\mathbf{x}^{(\ell)})^T (\mathbf{x} - \mathbf{x}^{(\ell)}) \\ &\quad + \frac{1}{2} (\mathbf{x} - \mathbf{x}^{(\ell)})^T \nabla^2 f(\mathbf{x}^{(\ell)}) (\mathbf{x} - \mathbf{x}^{(\ell)}) \end{aligned}$$

Coordinate descent

- Instead of minimizing $f(\mathbf{x})$, minimize $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ and cycle over i .
- Useful when $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ can be minimized in closed form.

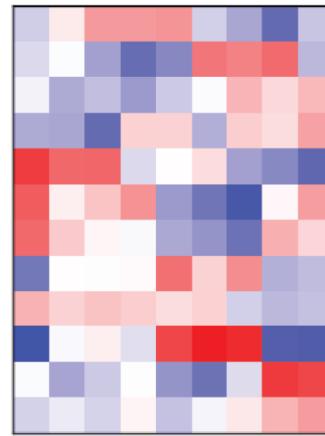
Majorization-minimization

- ① Find a majorizing function g for f at current iterate $\mathbf{x}^{(\ell)}$.
 - $f(\mathbf{x}) < g(\mathbf{x}; \mathbf{x}^{(\ell)})$ for all $\mathbf{x} \neq \mathbf{x}^{(\ell)}$
 - $f(\mathbf{x}^{(\ell)}) = g(\mathbf{x}^{(\ell)}; \mathbf{x}^{(\ell)})$
- ② Minimize the majorizing function to obtain $\mathbf{x}^{(\ell+1)}$.

References

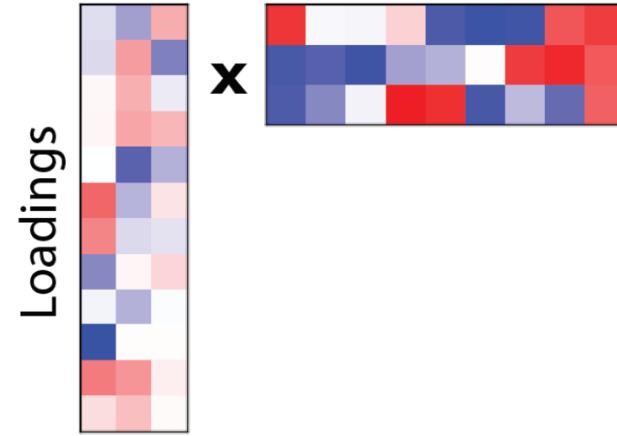
- Paatero, Pentti, and Unto Tapper (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics* 5.2: 111-126.
- Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755), 788-791.
- Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems* (pp. 556-562).
- Cichocki, A., Zdunek, R., Phan, A. H., & Amari, S. (2009). *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley.
- Gillis, N. (2014). Introduction to Nonnegative Matrix Factorization. *arXiv preprint arXiv:1401.5226*.
- Berry, M. W., Browne, M., Langville, A. N., Pauca, V. P., & Plemmons, R. J. (2007). Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1), 155-173.

Original Data

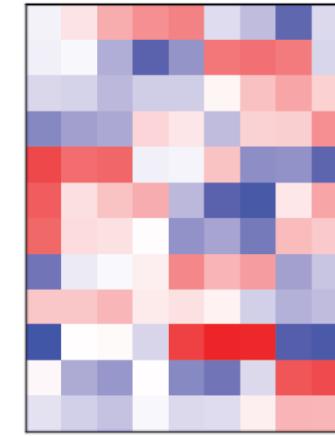


\approx

Components

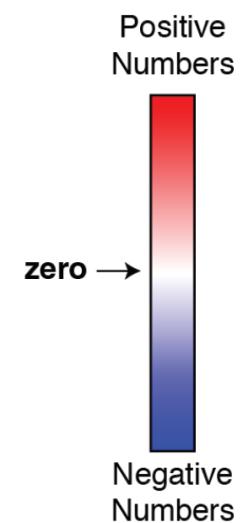
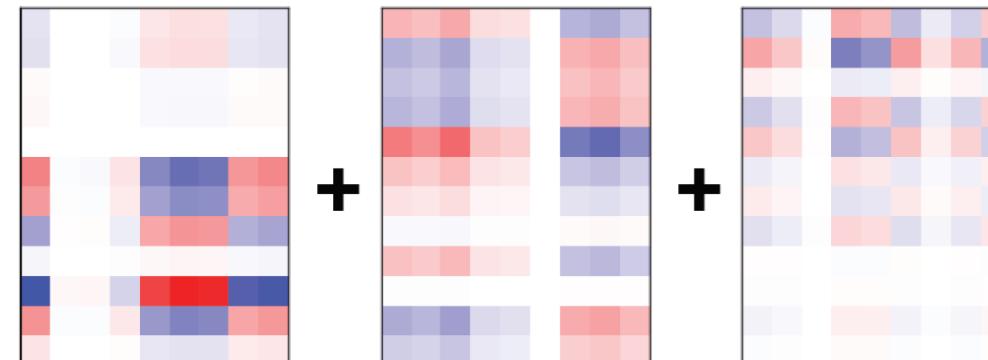


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax



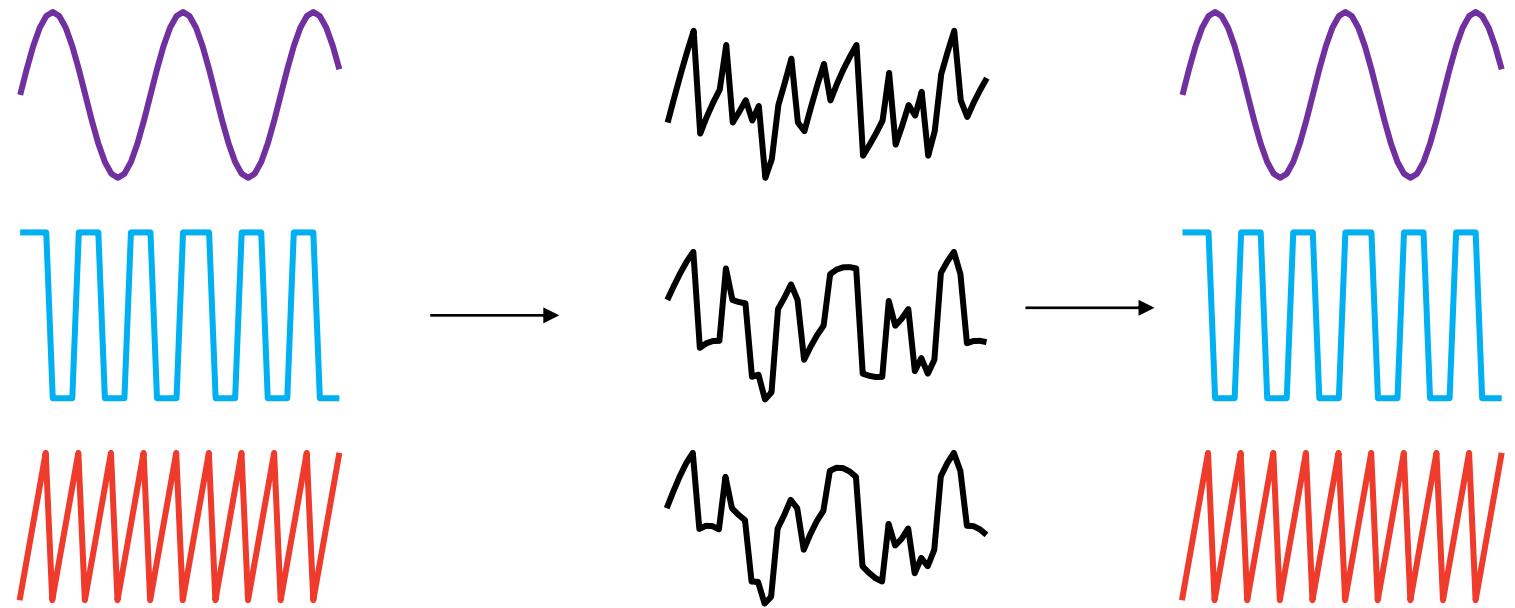
neurosyntax

Session 8

Lecturer: Saman Abbaspoor

Independent Component Analysis

Cocktail Party Problem



Source separation, blind signal separation (BSS) or blind source separation, is the separation of a set of source signals from a set of mixed signals, without the aid of information (or with very little information) about the source signals or the mixing process.

$$\begin{bmatrix} & \\ & \end{bmatrix}$$

$$S \times N$$

$$\begin{bmatrix} & \\ & \end{bmatrix}$$

$$N \times M$$

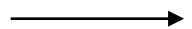
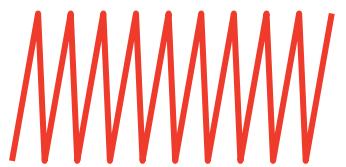
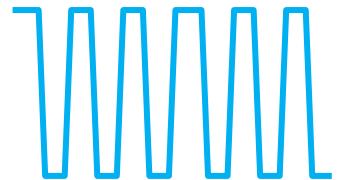
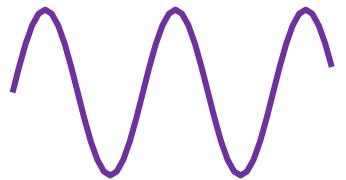
$$\begin{bmatrix} & \\ & \end{bmatrix}$$

$$S \times M$$

$$\begin{bmatrix} & \\ & \end{bmatrix}$$

$$S \times N$$

Cocktail Party Problem



$$x_1(t) = a_{11}s_1 + a_{12}s_2 + a_{13}s_3$$

$$x_2(t) = a_{21}s_1 + a_{22}s_2 + a_{23}s_3$$

$$x_3(t) = a_{31}s_1 + a_{32}s_2 + a_{33}s_3$$

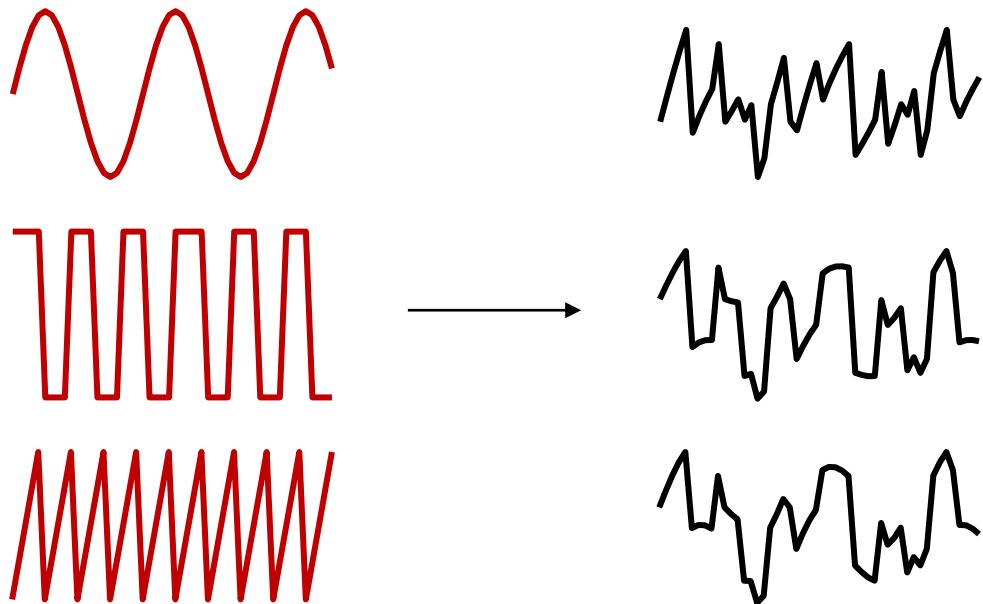
Cocktail Party Problem

$$x_1(t) = a_{11}s_1 + a_{12}s_2 + a_{13}s_3$$

$$x_2(t) = a_{21}s_1 + a_{22}s_2 + a_{23}s_3 \longrightarrow X = AS$$

$$x_3(t) = a_{31}s_1 + a_{32}s_2 + a_{33}s_3$$

Statistical Properties of Independent and Mixed Signals



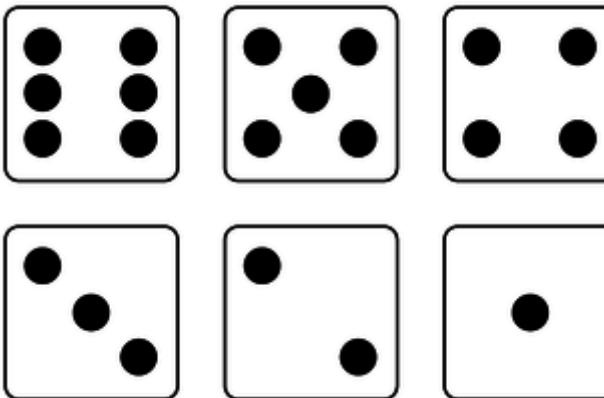
$$X = AS$$

One approach to solving this problem would be to use some information on the statistical properties of the signals $s_i(t)$ to estimate the a_{ii} . Actually, and perhaps surprisingly, it turns out that it is enough to assume that $s_1(t)$, $s_2(t)$ and $s_3(t)$, at each time instant t , are statistically independent. This is not an unrealistic assumption in many cases, and it need not be exactly true in practice.

Dice Roller

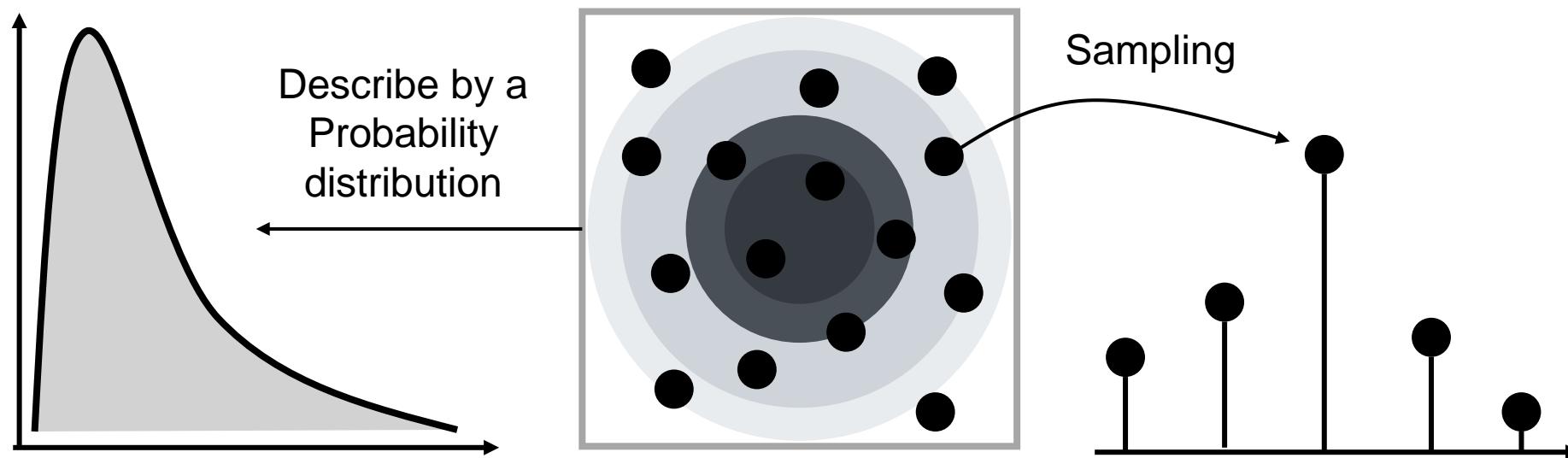


4	1	6	3	1	2	4	5	5	3
---	---	---	---	---	---	---	---	---	---

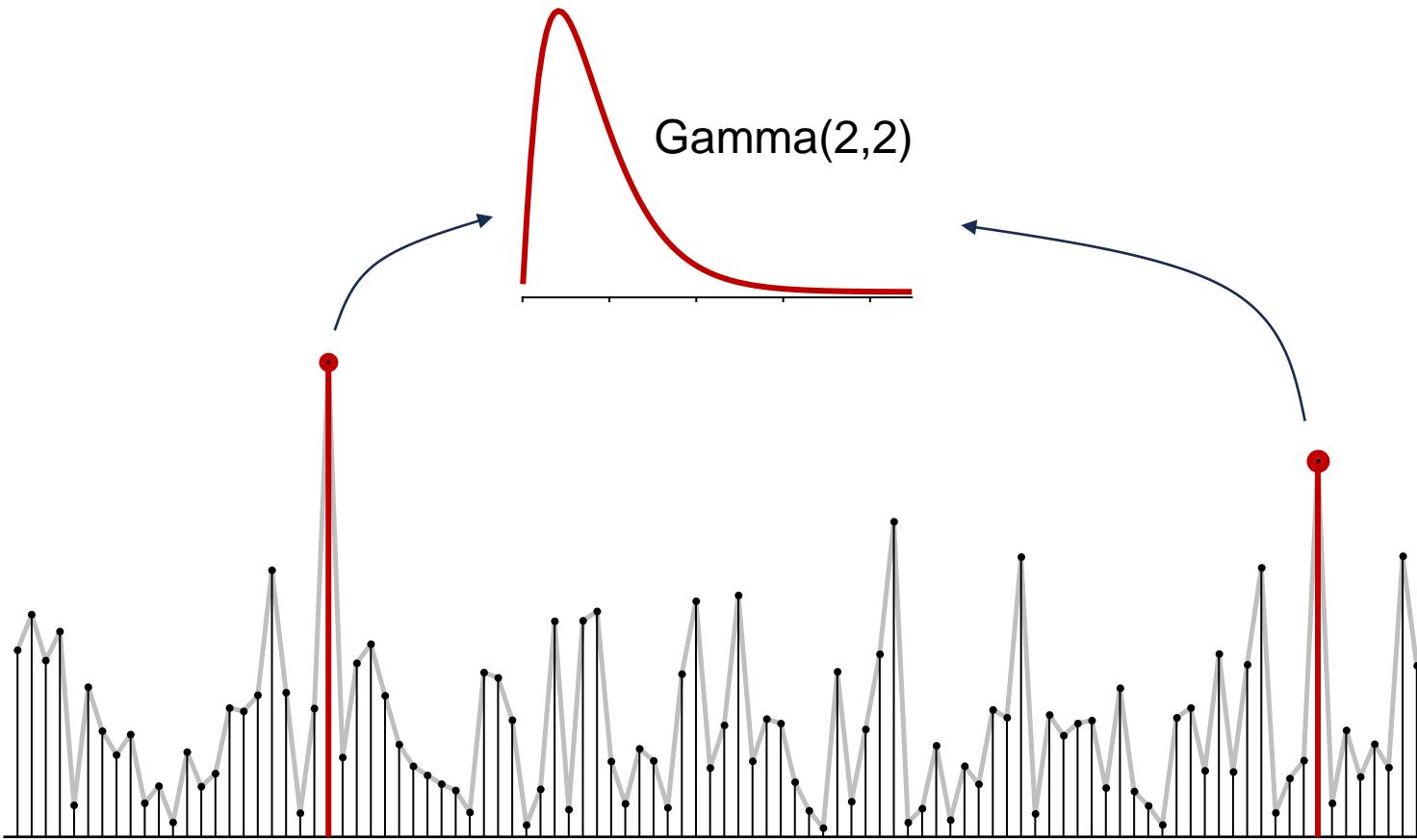


Random Variable

In the context of probability theory, a random variable is formally defined as a measurable function that maps outcomes from a sample space to the real numbers.

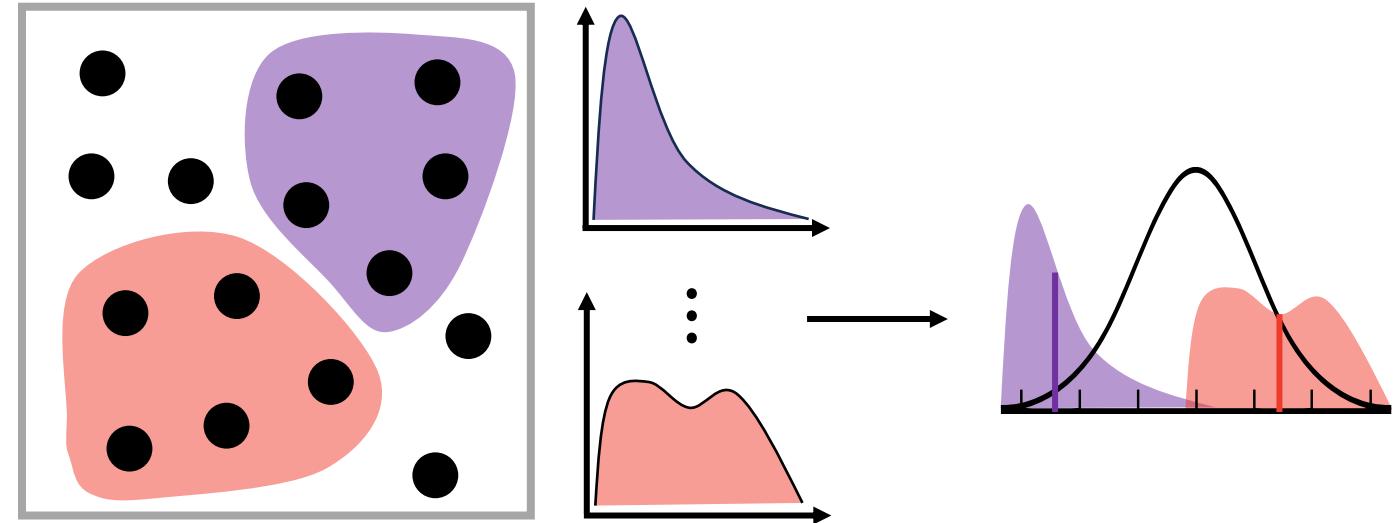


Modelling timeseries as random variables



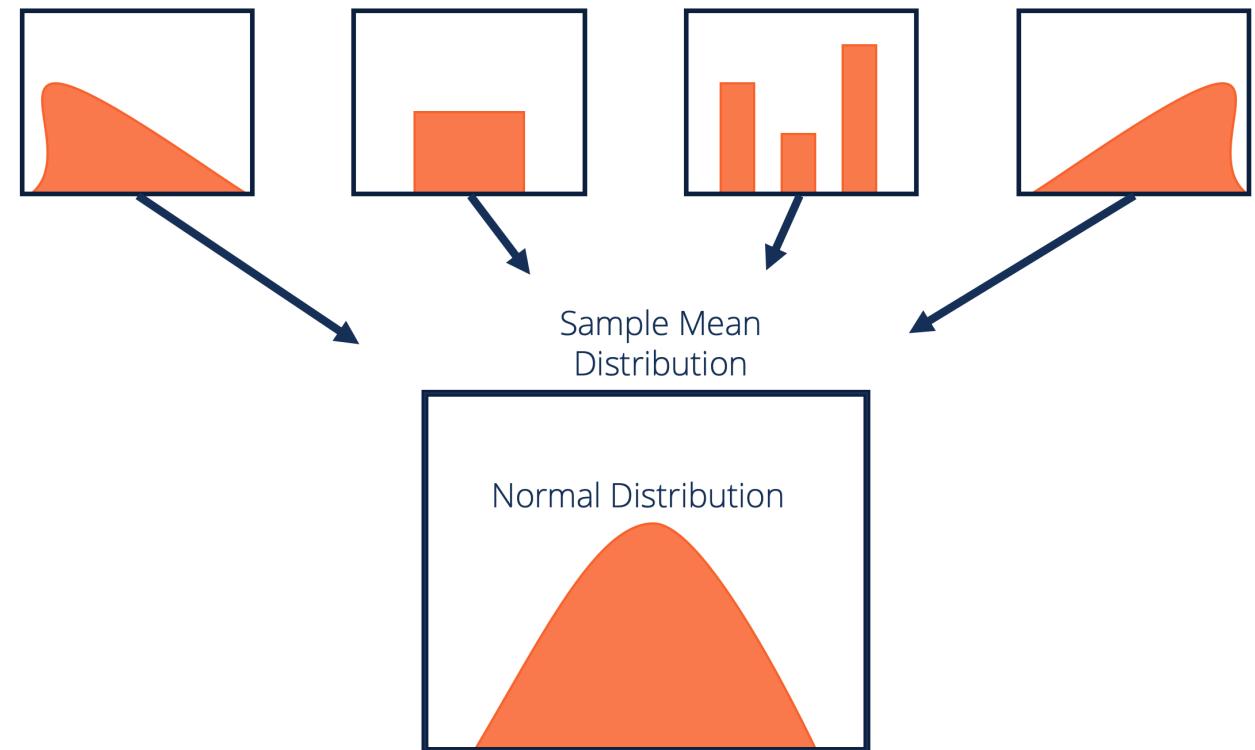
Central Limit Theorem

In probability theory, the central limit theorem (CLT) states that, under appropriate conditions, the distribution of a normalized version of the sample mean converges to a standard normal distribution. This holds even if the original variables themselves are not normally distributed.



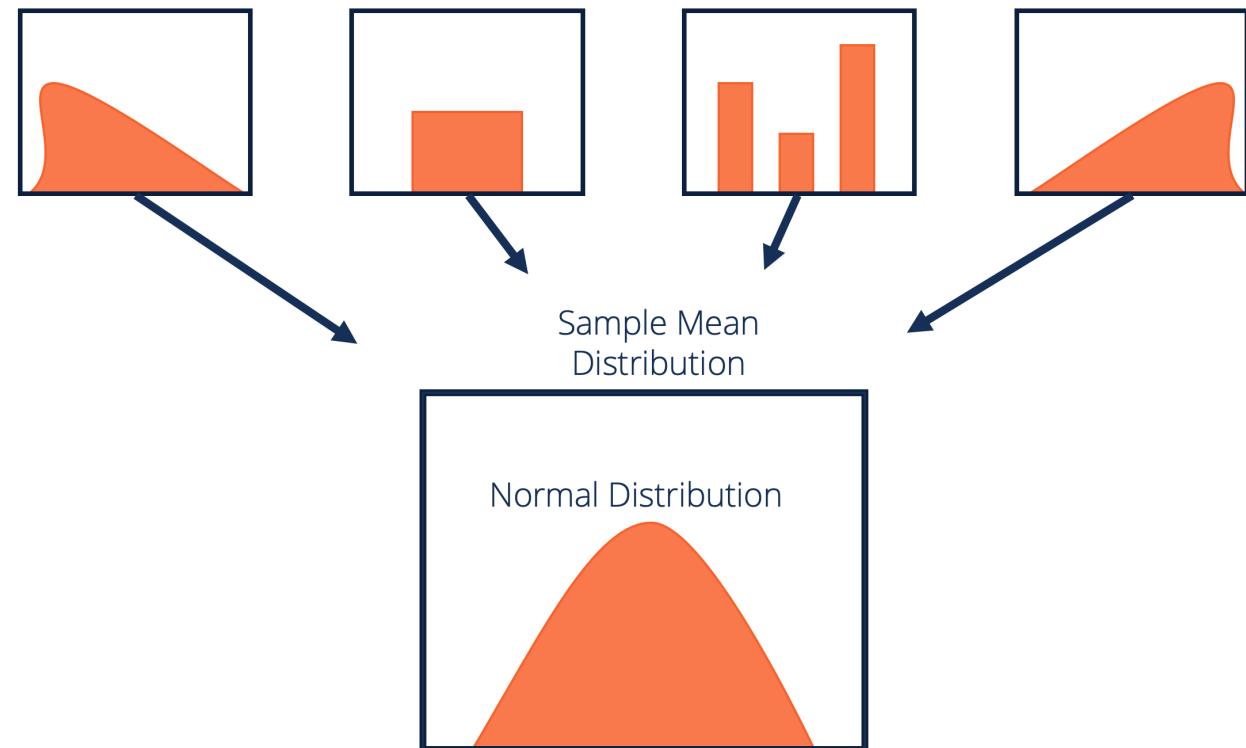
Central Limit Theorem

In probability theory, the central limit theorem (CLT) states that, under appropriate conditions, the distribution of a normalized version of the sample mean converges to a standard normal distribution. This holds even if the original variables themselves are not normally distributed.

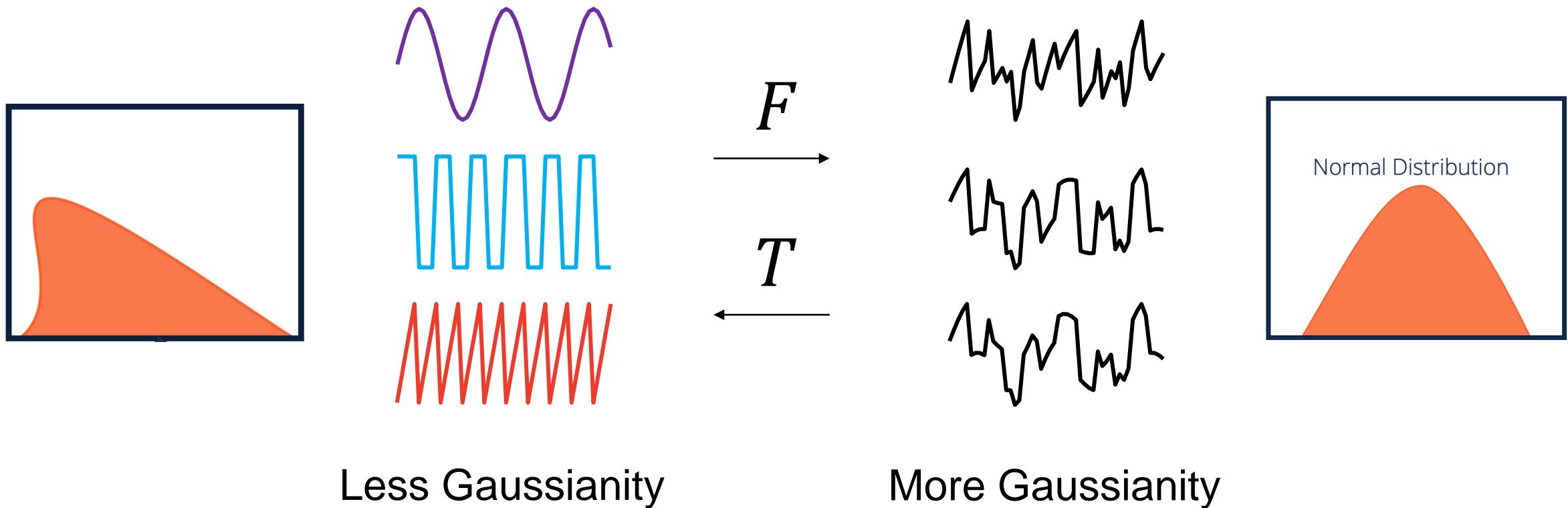


Central Limit Theorem, Non-Gaussianity and Independence

The Central Limit Theorem, a classical result in probability theory, tells that the distribution of a sum of independent random variables tends toward a Gaussian distribution, under certain conditions. **Thus, a sum of two independent random variables usually has a distribution that is closer to Gaussian than any of the two original random variables.**



Statistical Properties of Independent and Mixed Signals



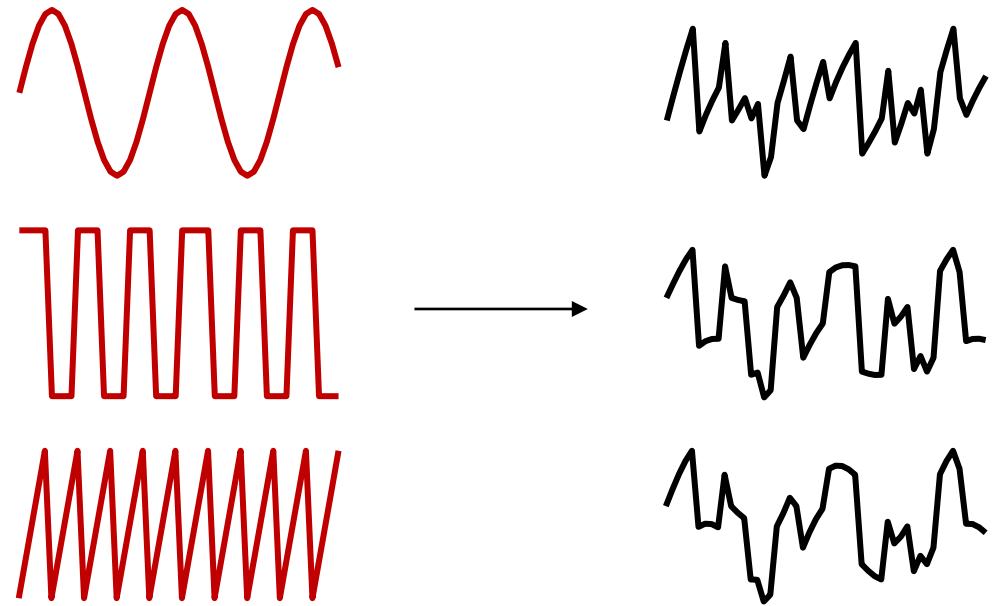


Optimization problem for ICA

$$X = AS$$

$$S = WX, W = A^{-1}$$

$$\max_W I(WX)$$



The goal is to find a linear representation of non-Gaussian data so that the components are statistically independent, or as independent as possible.

Measures of non-Gaussianity/Independence

- Kurtosis

$$\text{Kurtosis}(\mathbf{X}) = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^n \left(\frac{x_i - \bar{x}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}$$

- Negative entropy

$$J(Y) = H(Y_{\text{gauss}}) - H(Y)$$

- Minimization of mutual information

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

Kurtosis

Kurtosis is defined as:

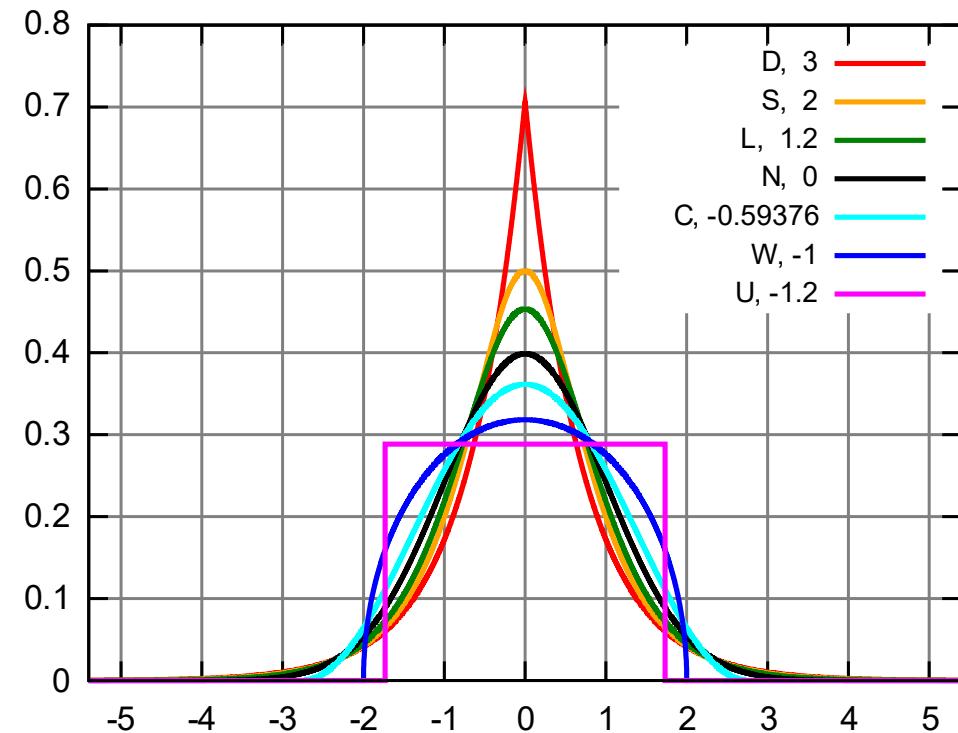
$$\text{Kurtosis} = \frac{E[(X - \mu)^4]}{\sigma^4}$$

where:

- X is a random variable,
- μ is the mean of X ,
- σ^2 is the variance of X ,
- E denotes the expectation.

For a normal distribution $X \sim N(\mu, \sigma^2)$, let's compute $E[(X - \mu)^4]$.

Probability density functions for selected distributions with mean 0, variance 1 and different excess kurtosis



Kurtosis

Let $Z = \frac{X-\mu}{\sigma}$, so $Z \sim N(0, 1)$. The fourth central moment of X is:

$$E[(X - \mu)^4] = E[(\sigma Z)^4] = \sigma^4 E[Z^4]$$

So, the kurtosis can be rewritten as:

$$\text{Kurtosis} = \frac{\sigma^4 E[Z^4]}{\sigma^4} = E[Z^4]$$

Thus, the problem reduces to computing $E[Z^4]$ where $Z \sim N(0, 1)$.

Kurtosis

The fourth moment of a standard normal distribution can be computed using the fact that for any standard normal variable $Z \sim N(0, 1)$, the even moments are given by:

$$E[Z^{2n}] = \frac{(2n)!}{2^n n!}$$

For $n = 2$, we have:

$$E[Z^4] = \frac{4!}{2^2 \cdot 2!} = \frac{24}{4 \cdot 2} = \frac{24}{8} = 3$$

Kurtosis

The kurtosis for the standard normal distribution Z is:

$$\text{Kurtosis} = E[Z^4] = 3$$

This matches the kurtosis for the general normal distribution X , which also has a kurtosis of 3.

Thus, the **kurtosis** for a normal distribution is 3, and the **excess kurtosis** (which subtracts 3) is 0. This confirms that the normal distribution has a baseline kurtosis of 3.

Independent Component Analysis (ICA) and Blind Source Separation (BSS)

The FastICA package for MATLAB

The FastICA package is a free (GPL) MATLAB program that implements the fast fixed-point algorithm for independent component analysis and projection pursuit. It features an easy-to-use graphical user interface, and a computationally powerful algorithm.



[Download software package](#)

Related software

[ICASSO: analysing and visualising the reliability of independent components](#)

[ISCTEST: principled statistical testing of independent components](#)

FastICA for other environments

[FastICA in R](#)

[FastICA in C++ \(part of IT++ package\)](#)

[FastICA in Python as part of MDP package](#)

[FastICA in Python as part of scikit-learn package](#)

Independent Component Analysis (ICA) and Blind Source Separation (BSS)

Documentation Examples Functions Blocks Apps More ▾ Videos Answers

Trial Software Product Updates

rica

Feature extraction by using reconstruction ICA

R2024a

[collapse all in page](#)

Syntax

```
Mdl = rica(X,q)  
Mdl = rica(X,q,Name,Value)
```

Description

`Mdl = rica(X,q)` returns a reconstruction independent component analysis (RICA) model object that contains the results from applying RICA to the table or matrix of predictor data `X` containing p variables. `q` is the number of features to extract from `X`, therefore `rica` learns a p -by- q matrix of transformation weights. For undercomplete or overcomplete feature representations, `q` can be less than or greater than the number of predictor variables, respectively.

- To access the learned transformation weights, use `Mdl.TransformWeights`.
- To transform `X` to the new set of features by using the learned transformation, pass `Mdl` and `X` to `transform`.

`Mdl = rica(X,q,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments. For example, you can standardize the predictor data or specify the value of the penalty coefficient in the reconstruction term of the objective function.

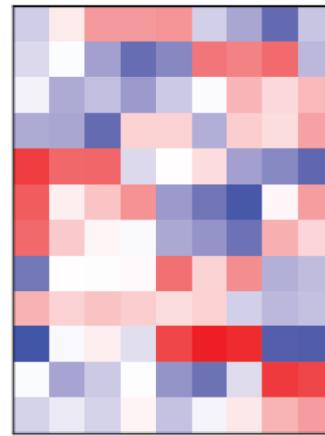
[example](#)



Assumptions of ICA

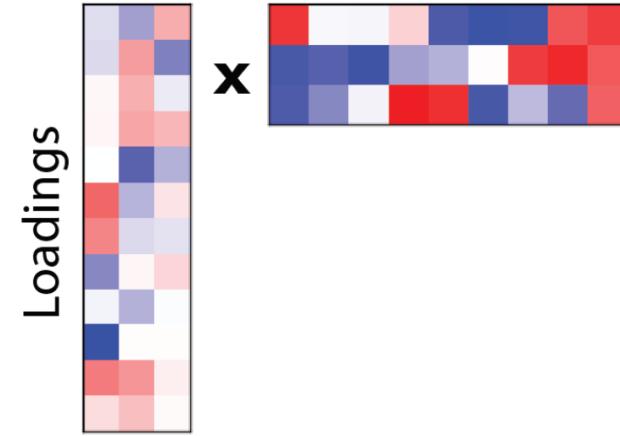
- **Statistical Independence of Components:** The starting point for ICA is the very simple assumption that the components are statistically independent
- **Non-Gaussianity:** It will be seen below that we must also assume that the independent component must have non-Gaussian distributions. However, in the basic model we do not assume these distributions known (if they are known, the problem is considerably simplified).
- **Stationarity:** Many ICA algorithms assume that the statistical properties of the sources (mean and variance, at least) do not change over time. This assumption simplifies the model but can be violated in practical situations where source statistics may evolve.

Original Data

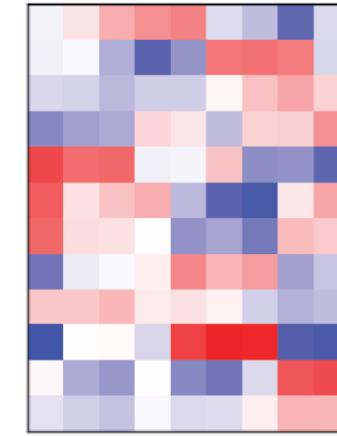


\approx

Components

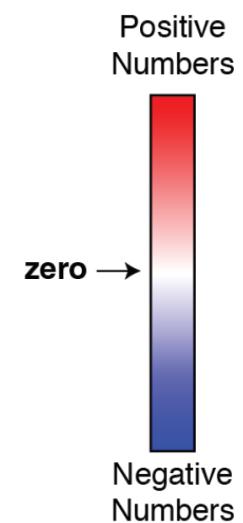
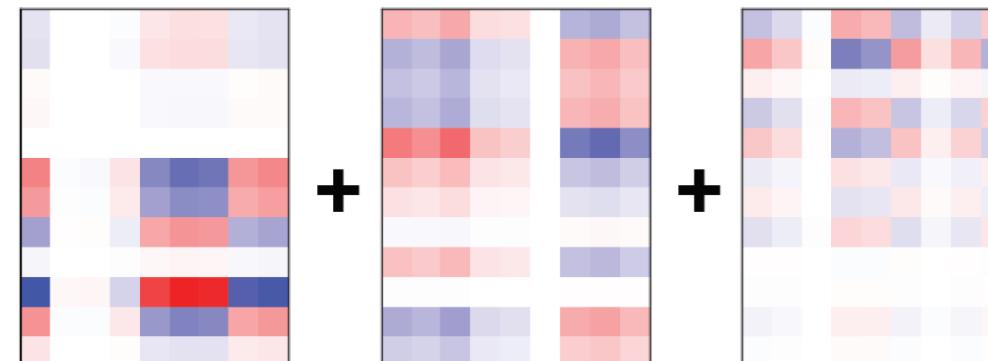


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax



neurosyntax

Session 9

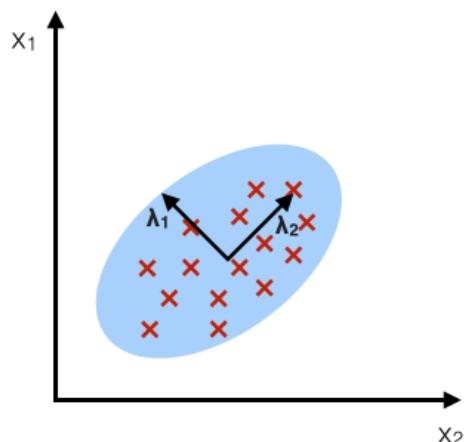
Lecturer: Saman Abbaspoor

Linear Discriminant Analysis

Linear Discriminant Analysis

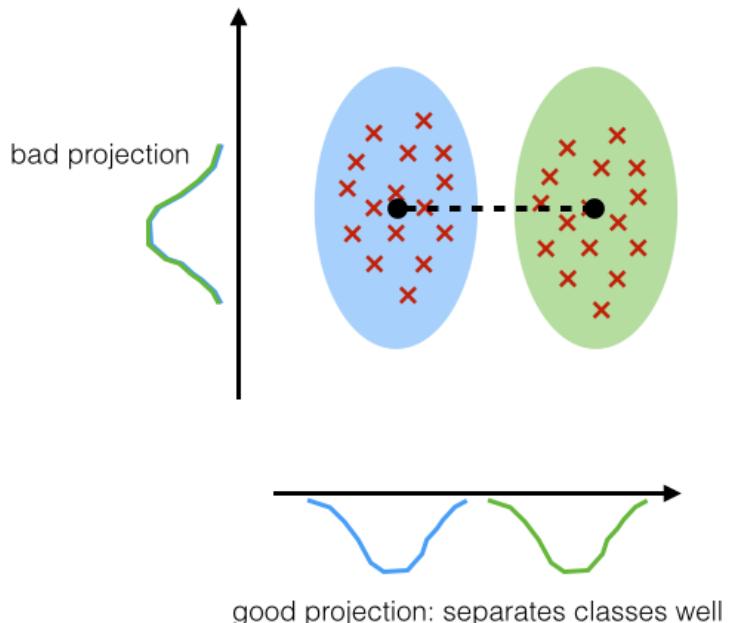
PCA:

component axes that maximize the variance



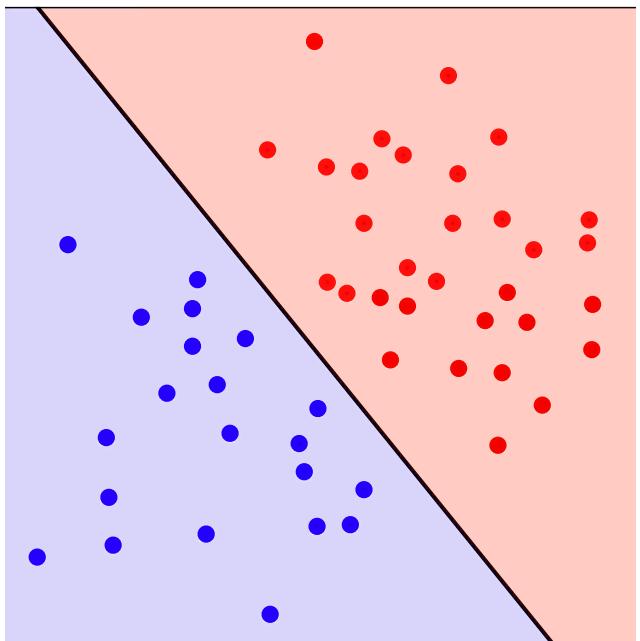
LDA:

maximizing the component axes for class-separation

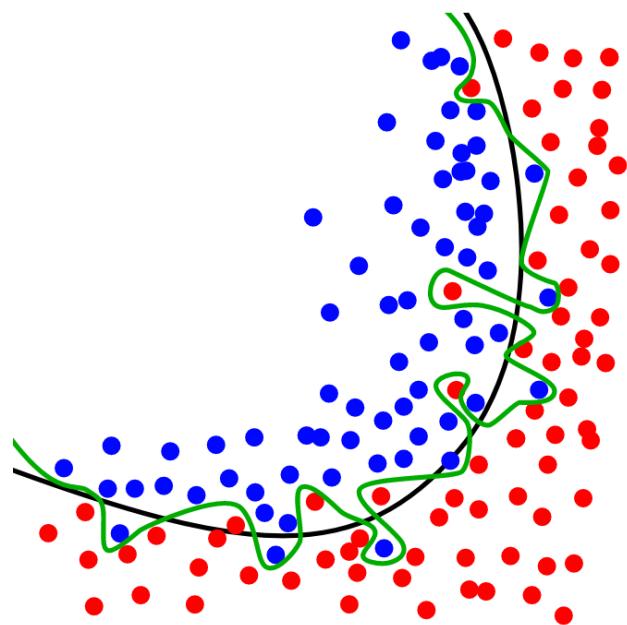
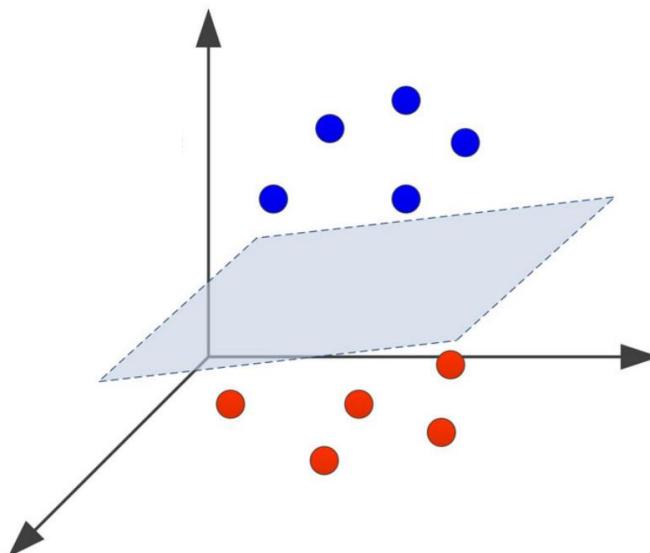


Linear discrimination

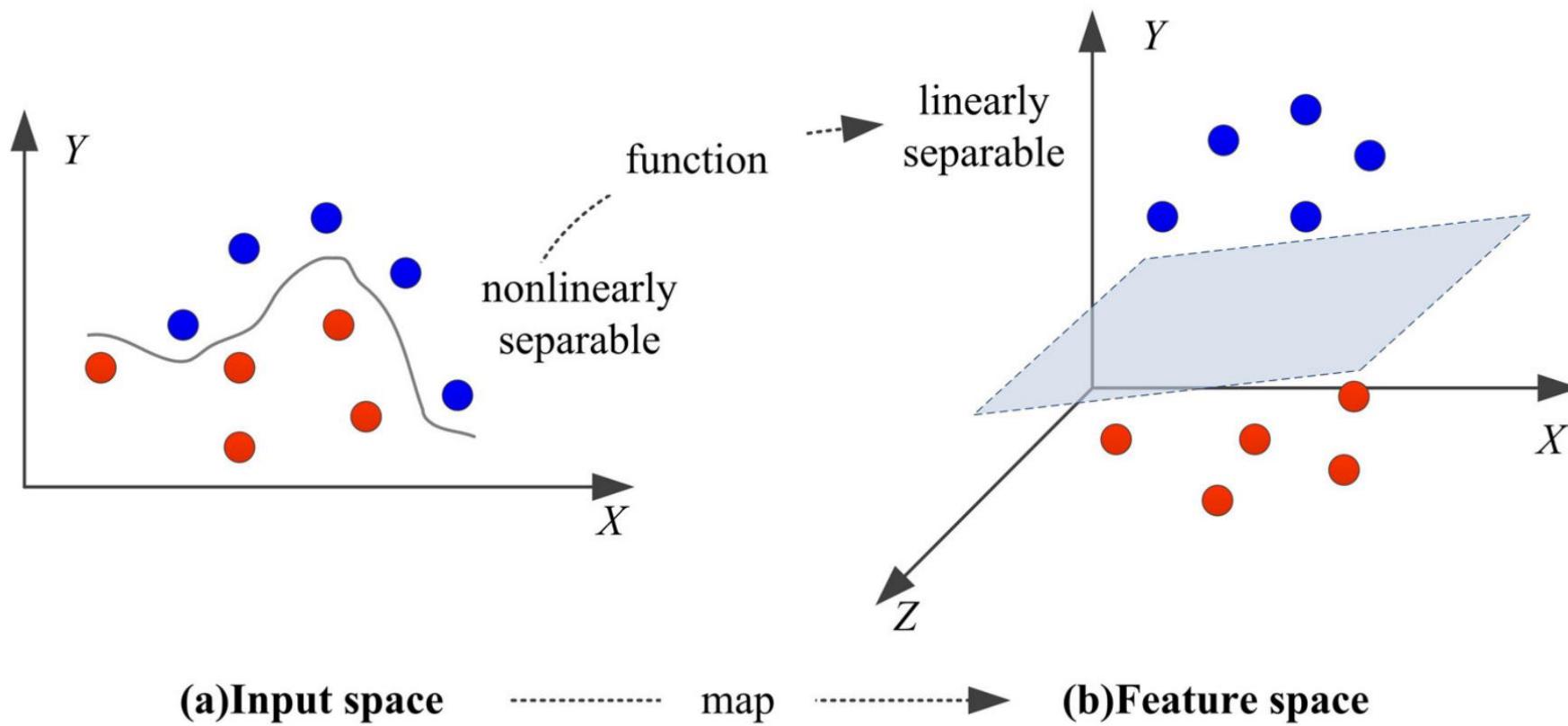
Linear
Separation



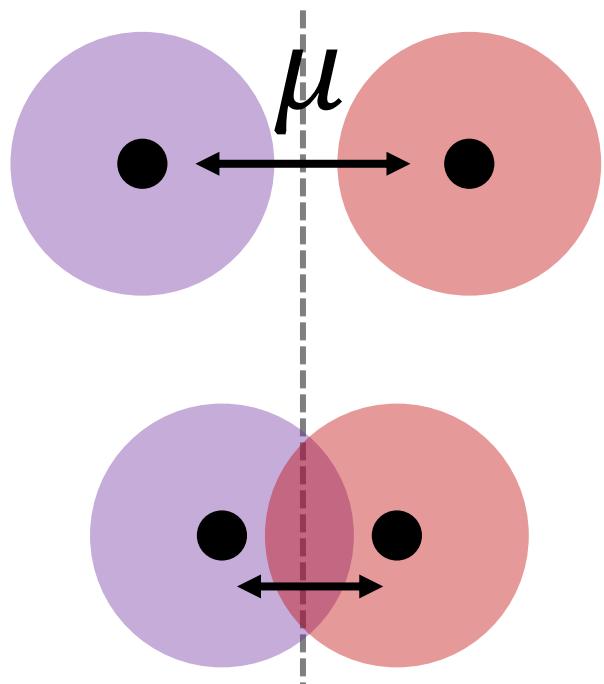
Nonlinear
Separation



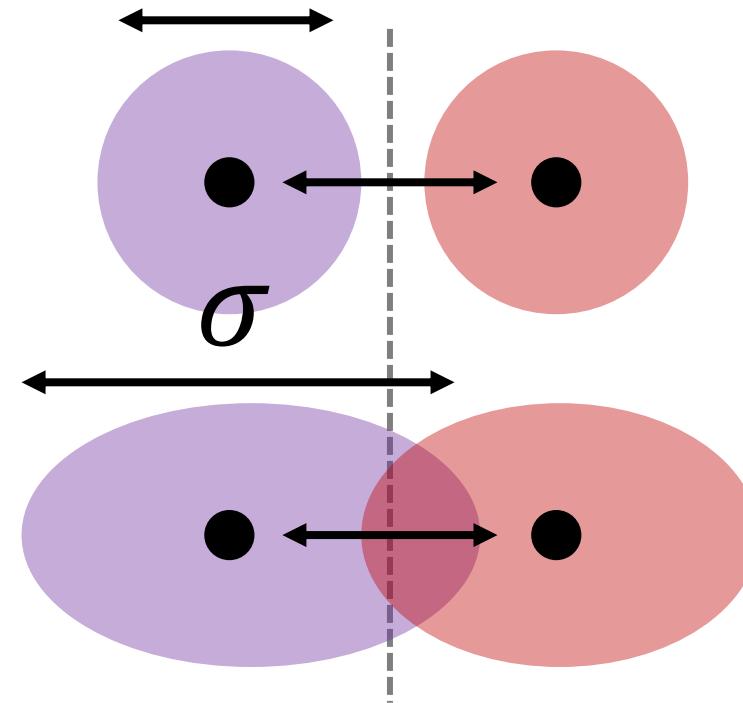
Linear discrimination



Variability and Separability



Between-class variability

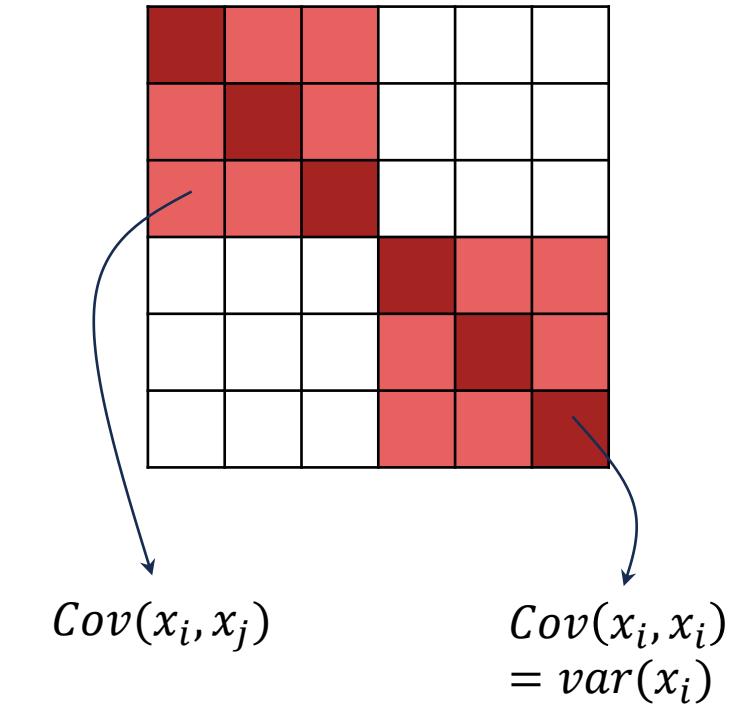


Within-class variability

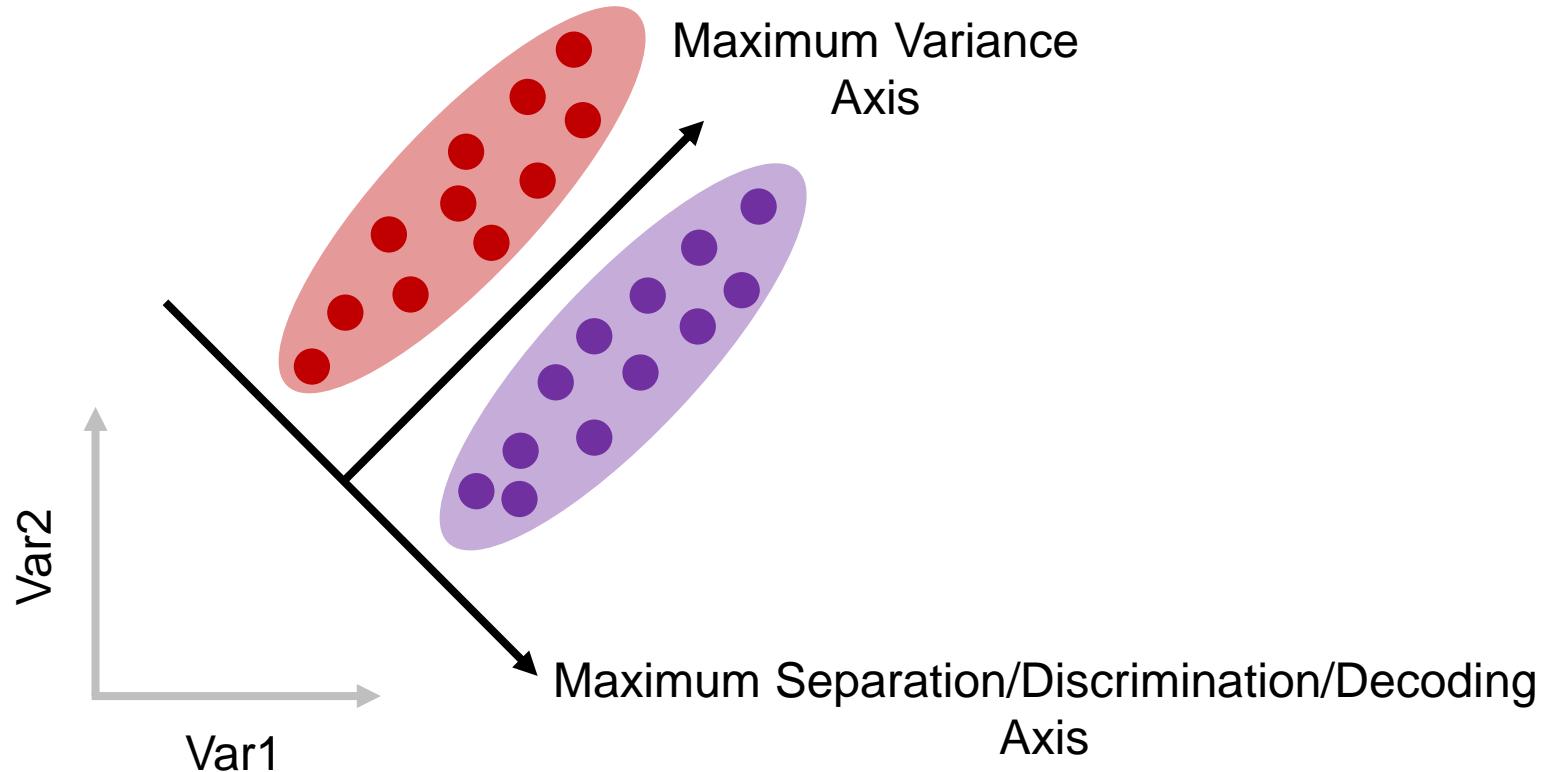
Covariance and Variance

$$cov_{x,y} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N - 1}$$

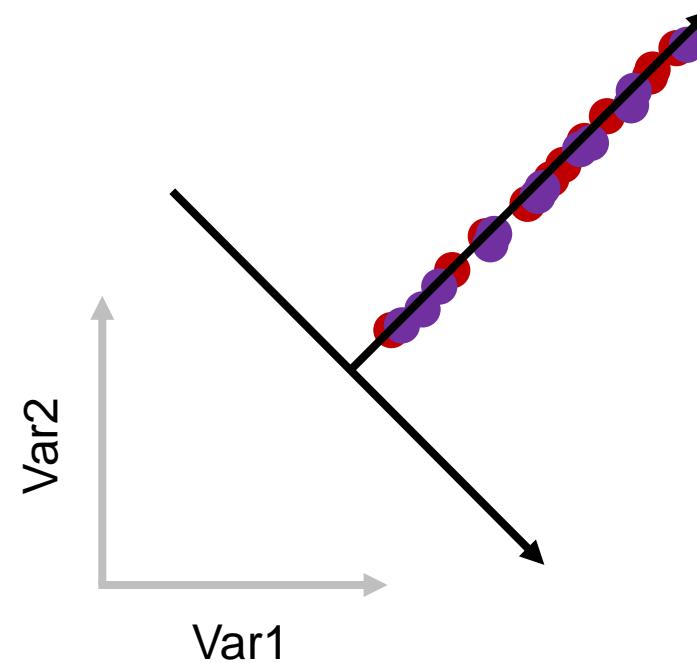
$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n - 1}$$



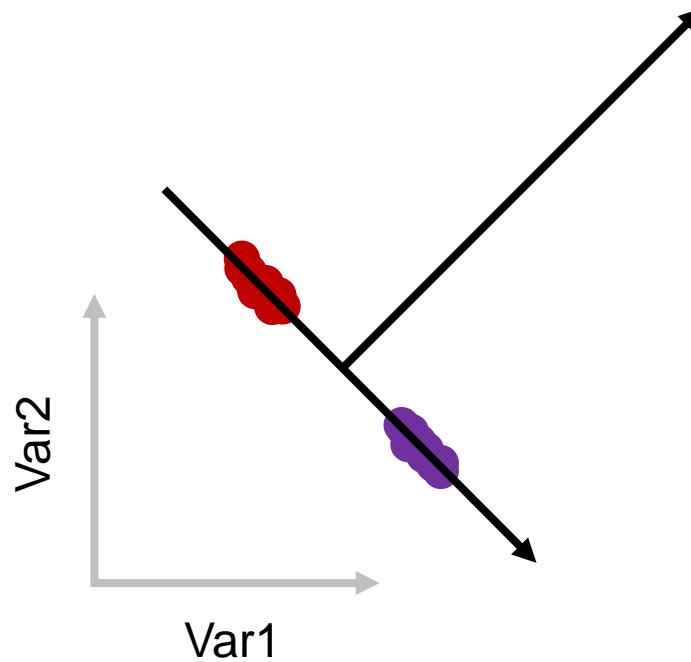
The structure of correlations and decodability



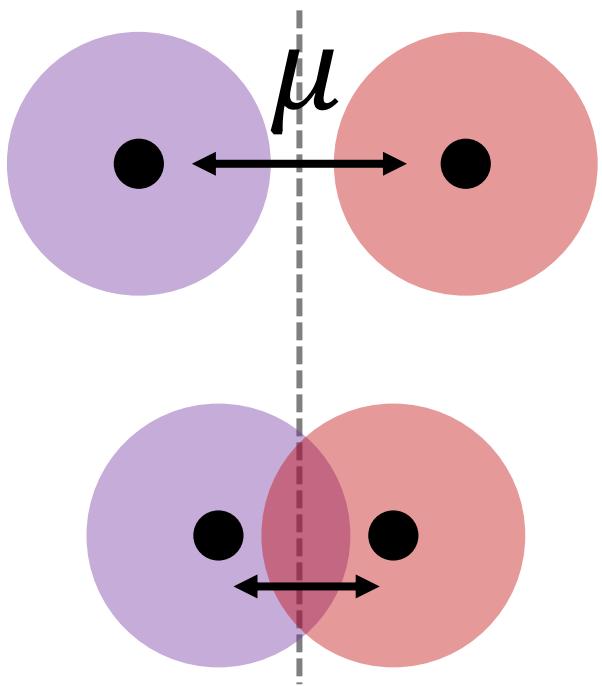
The structure of correlations and decodability



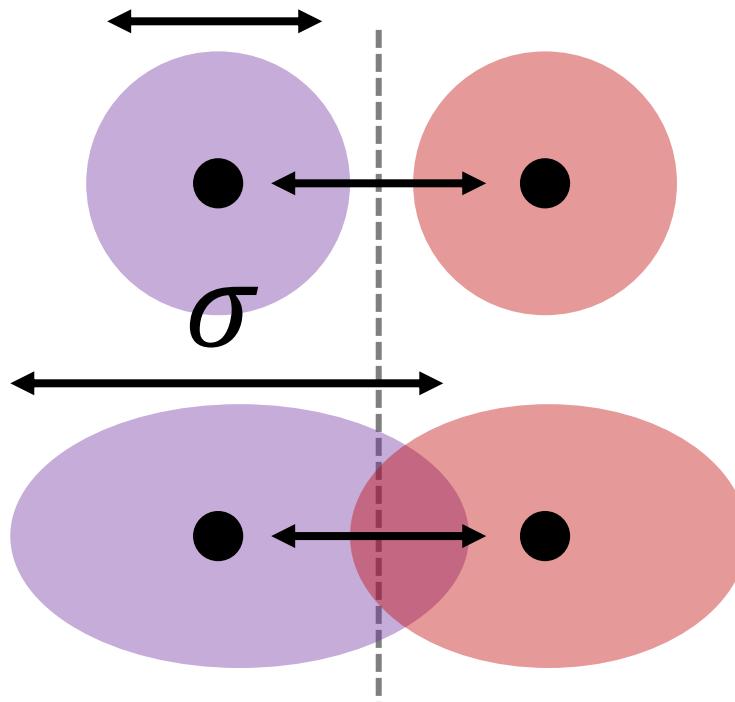
The structure of correlations and decodability



Variability and Separability



Between-class variability (S_B)



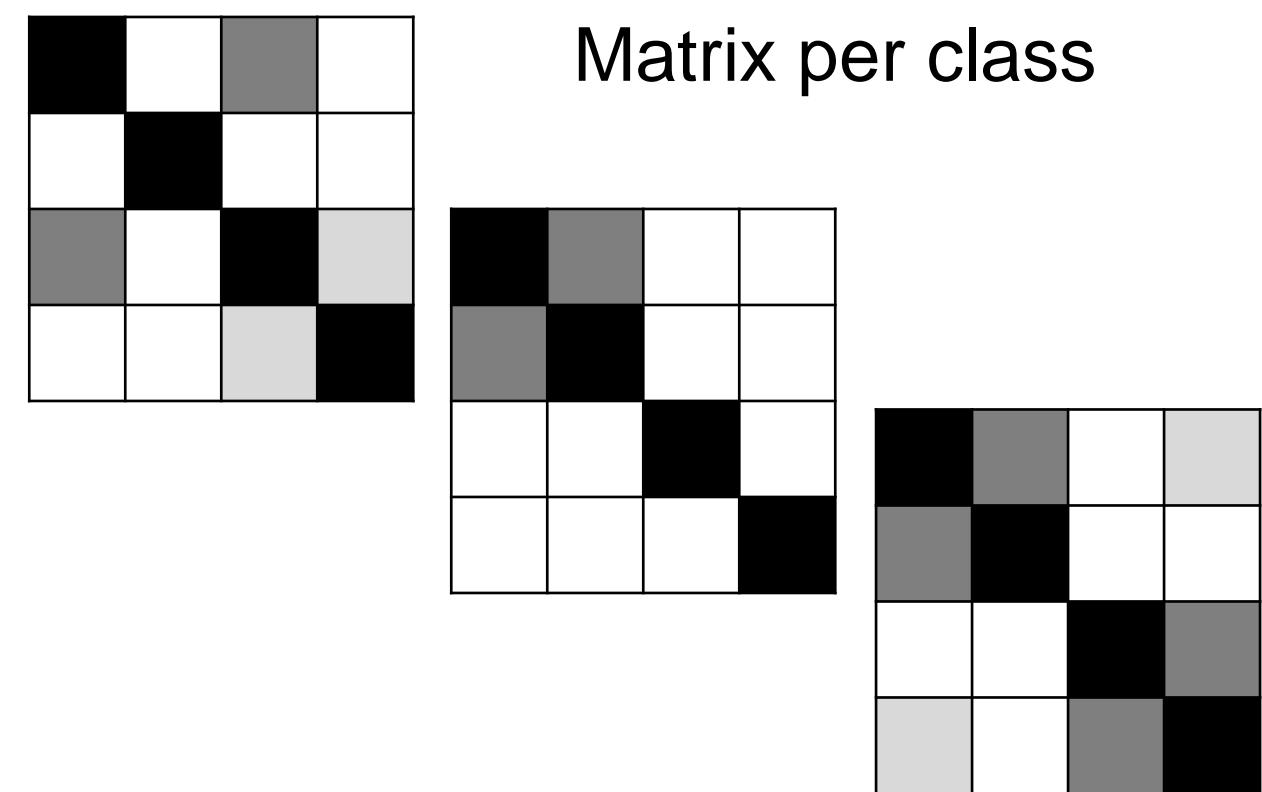
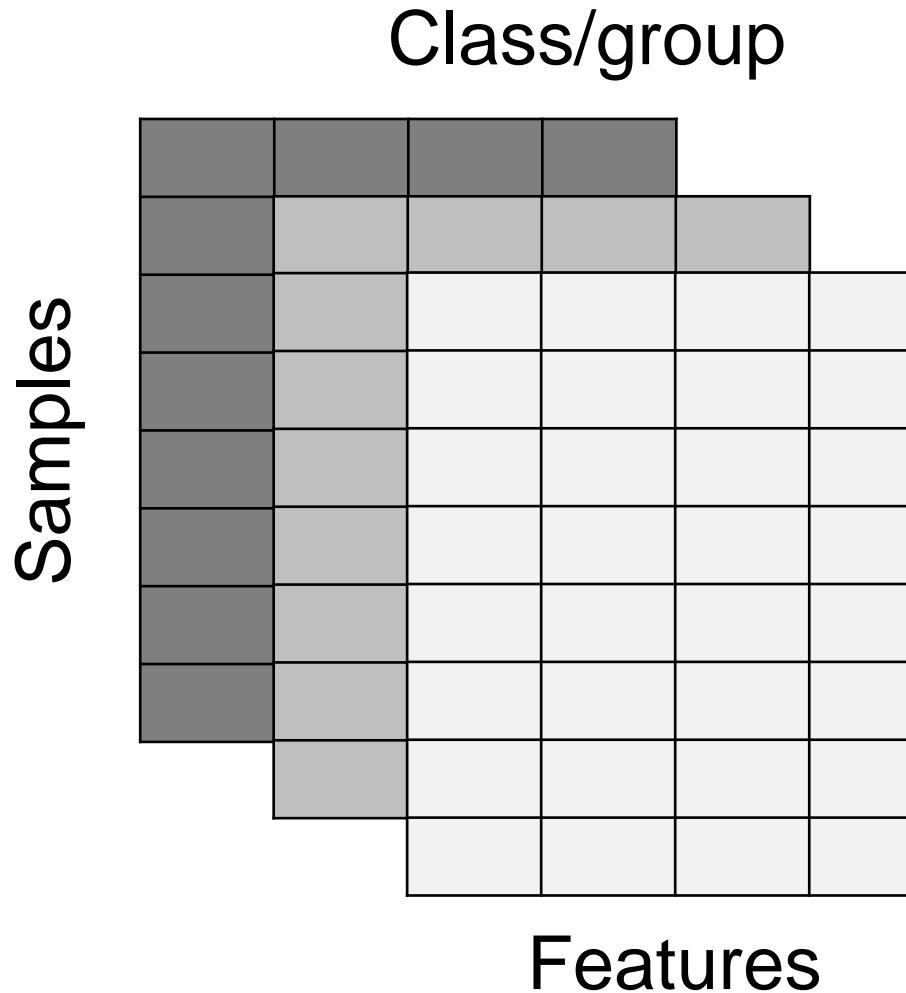
Within-class variability (S_w)

$$\text{Max}\left(\frac{S_B}{S_w}\right)$$

LDA

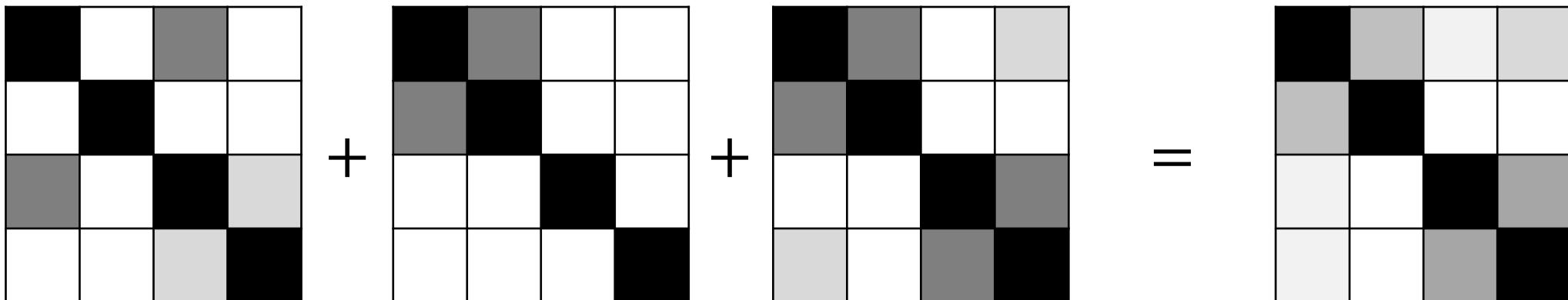
Implementation

The structure of the dataset for LDA



Within-class scatter matrix (S_W)

$$S_W = \sum_{c=1}^C S_c = \sum_{c=1}^C \sum_{i=1}^{n_c} (x_i - \mu_c)(x_i - \mu_c)^T$$



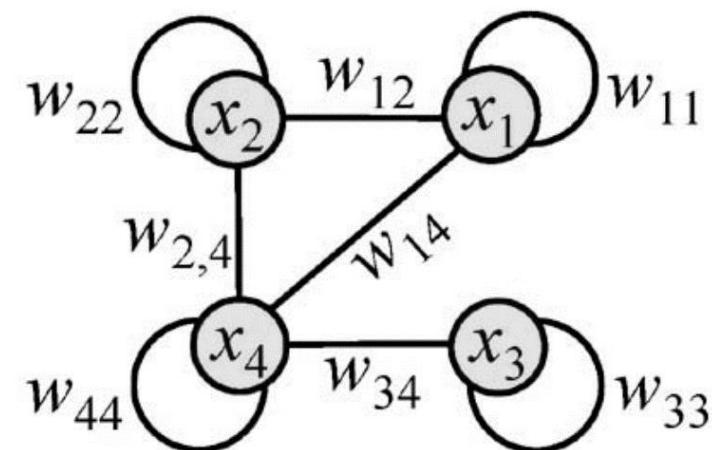
Covariance Matrix per class

Scatter Matrix

Within-class scatter matrix (S_W)

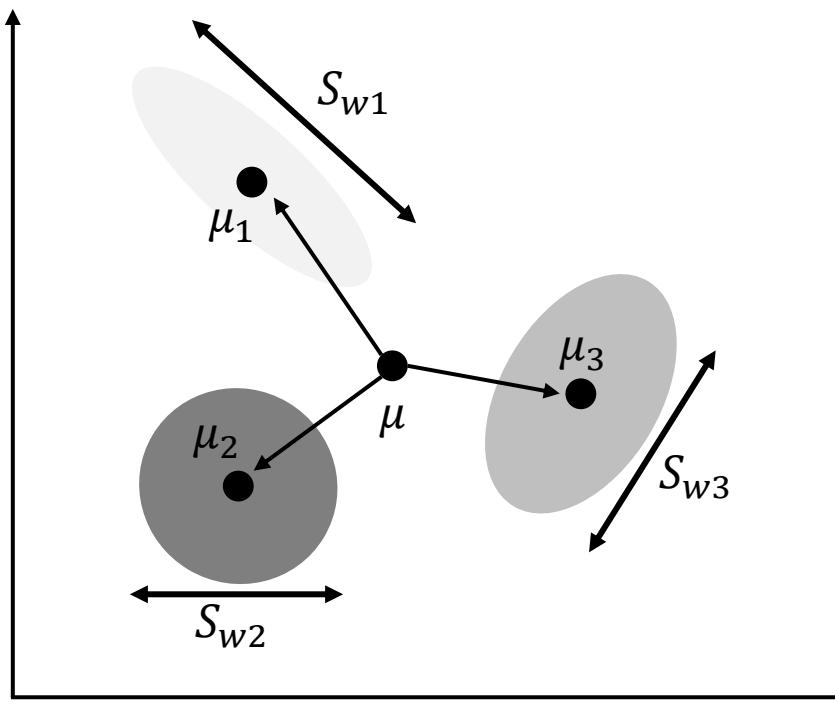
$$S_W = \sum_{c=1}^C S_c = \sum_{c=1}^C \sum_{i=1}^{n_c} (x_i - \mu_c)(x_i - \mu_c)^T$$

$$\mathbf{S}_W = \begin{bmatrix} \mathbf{S}_{W,1,1} & \mathbf{S}_{W,1,2} & 0 & \mathbf{S}_{W,1,4} \\ \mathbf{S}_{W,1,2} & \mathbf{S}_{W,2,2} & 0 & \mathbf{S}_{W,2,4} \\ 0 & 0 & \mathbf{S}_{W,3,3} & \mathbf{S}_{W,3,4} \\ \mathbf{S}_{W,1,4} & \mathbf{S}_{W,2,4} & \mathbf{S}_{W,3,4} & \mathbf{S}_{W,4,4} \end{bmatrix}$$



Between-class scatter matrix (S_B)

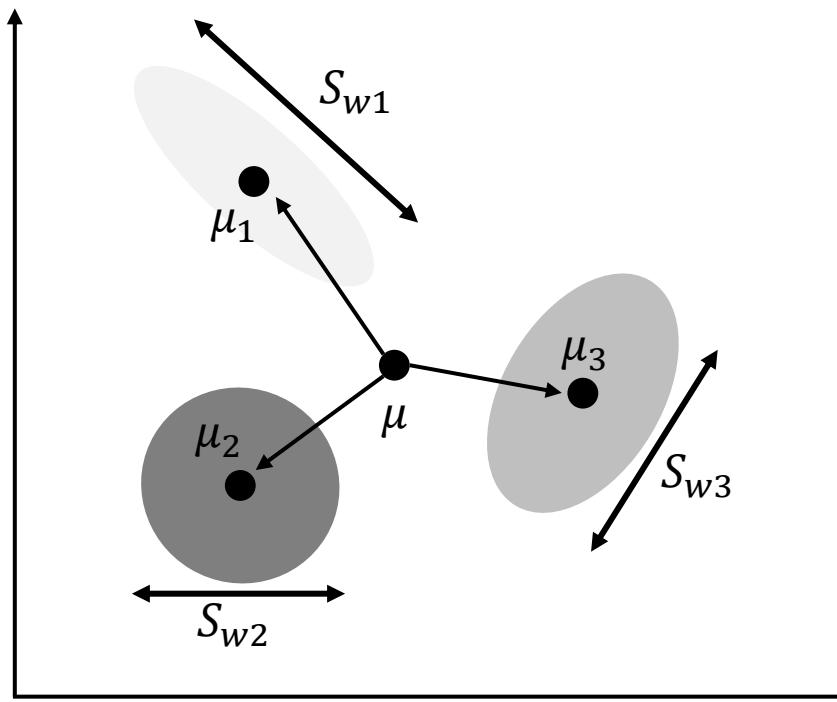
$$S_B = \sum_{c=1}^C n_c(\mu_c - \mu)(\mu_c - \mu)^T$$



LDA is the eigendecomposition
of scatter matrix: $Eig(S_W^{-1}S_B)$

Maximizing Class Separability

$$Eig(S_W^{-1}S_B)$$



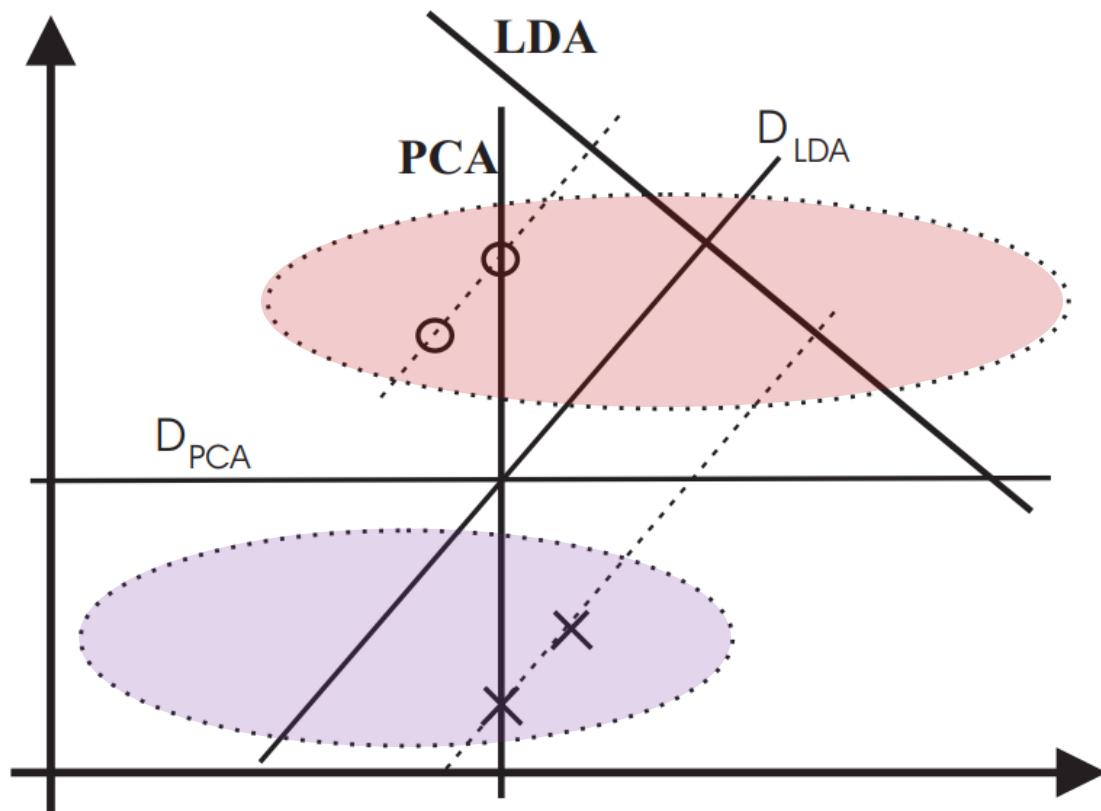
Summarizing the LDA approach in 5 steps

Listed below are the 5 general steps for performing a linear discriminant analysis; we will explore them in more detail in the following sections.

1. Compute the d -dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors ($\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_d$) and corresponding eigenvalues ($\lambda_1, \lambda_2, \dots, \lambda_d$) for the scatter matrices.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix \mathbf{W} (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $\mathbf{Y} = \mathbf{X} \times \mathbf{W}$ (where \mathbf{X} is a $n \times d$ -dimensional matrix representing the n samples, and \mathbf{y} are the transformed $n \times k$ -dimensional samples in the new subspace).



PCA vs LDA

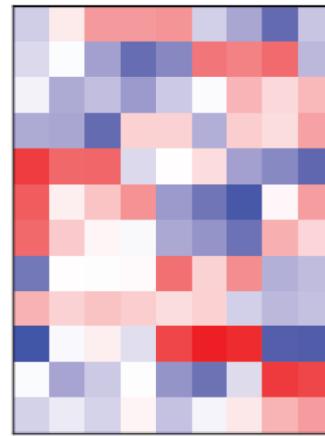


Assumptions

It should be mentioned that LDA assumes normal distributed data, features that are statistically independent, and identical covariance matrices for every class. However, this only applies for LDA as classifier and LDA for dimensionality reduction can also work reasonably well if those assumptions are violated. And even for classification tasks LDA seems can be quite robust to the distribution of the data:

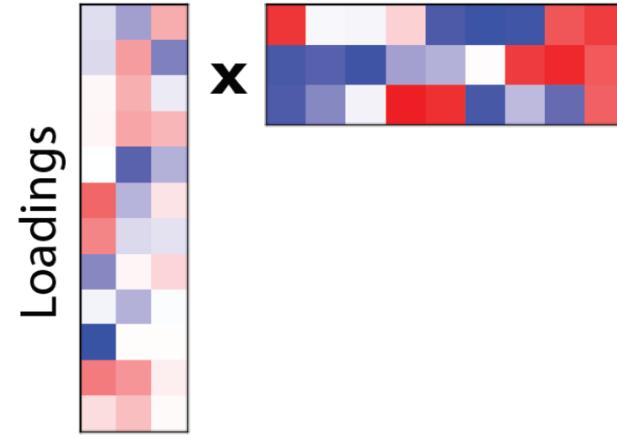
“Linear discriminant analysis frequently achieves good performances in the tasks of face and object recognition, even though the assumptions of common covariance matrix among groups and normality are often violated (Duda, et al., 2001)” (Tao Li, et al., 2006).

Original Data

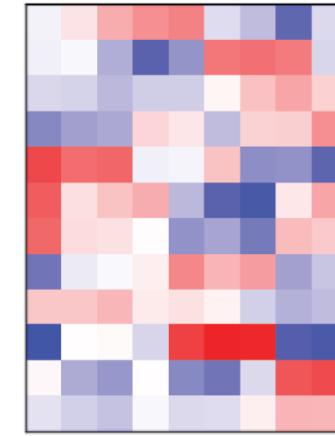


\approx

Components

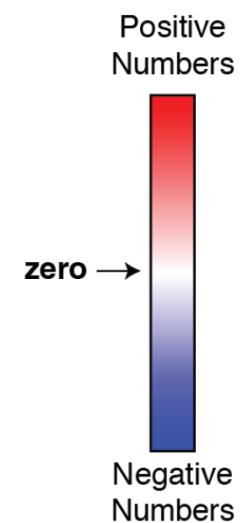
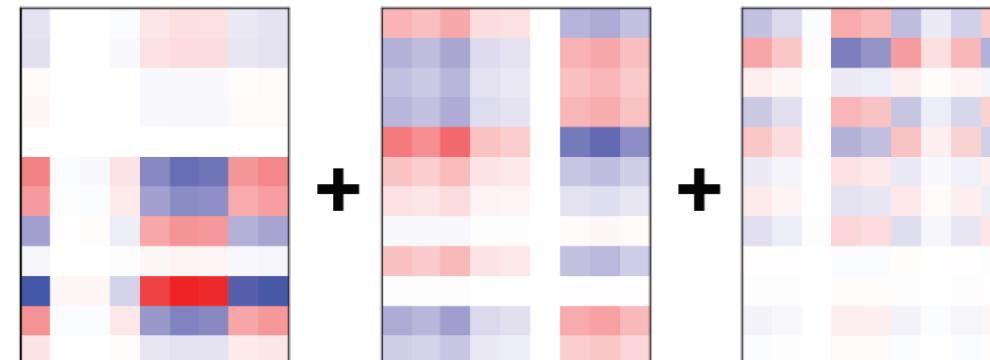


Reconstruction



Matrix decomposition

Sum of
Rank-1
Matrices



Neurosyntax Academy



@neurosyntaxacademy



@Neuro_Syntax



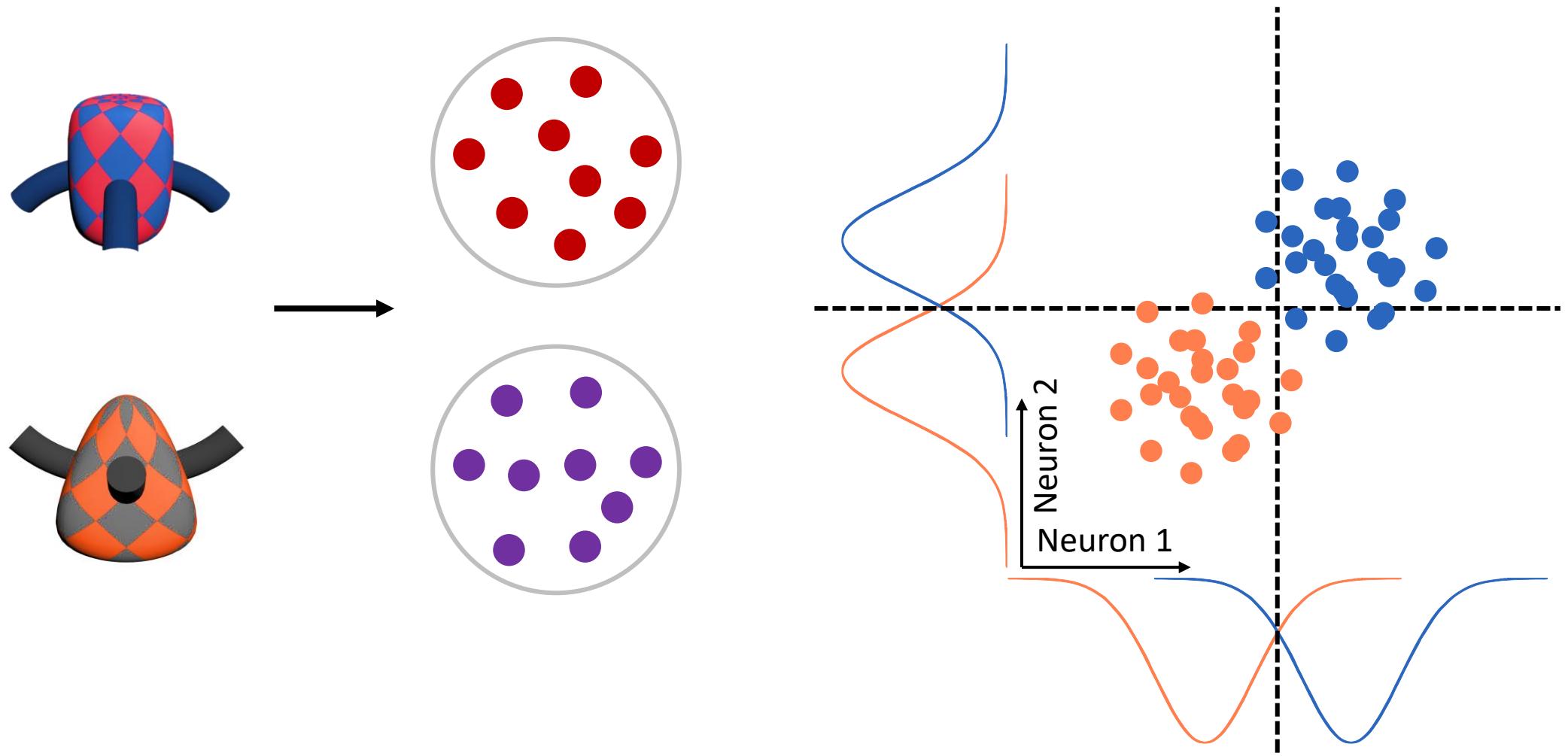
neurosyntax

Session 10

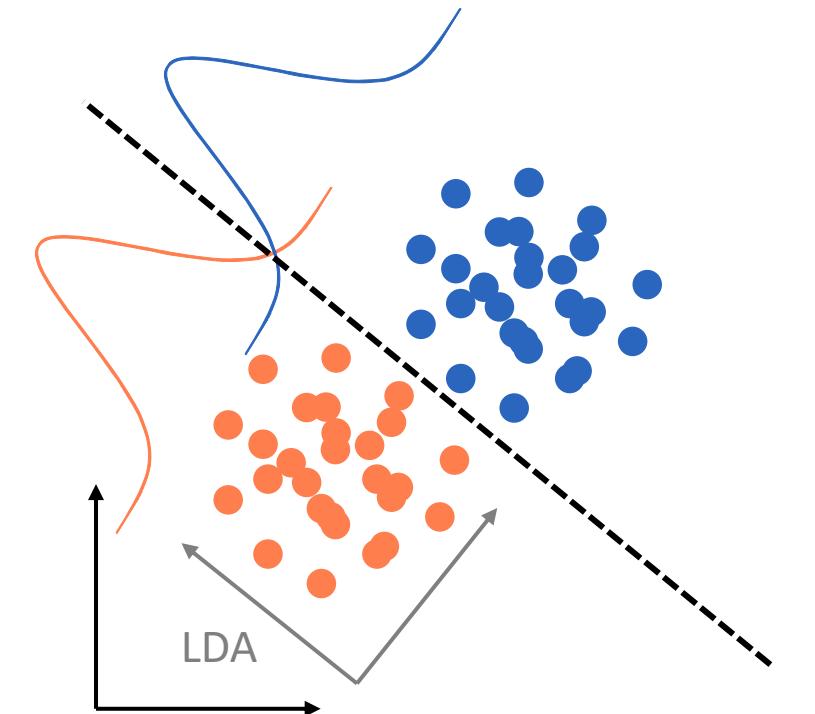
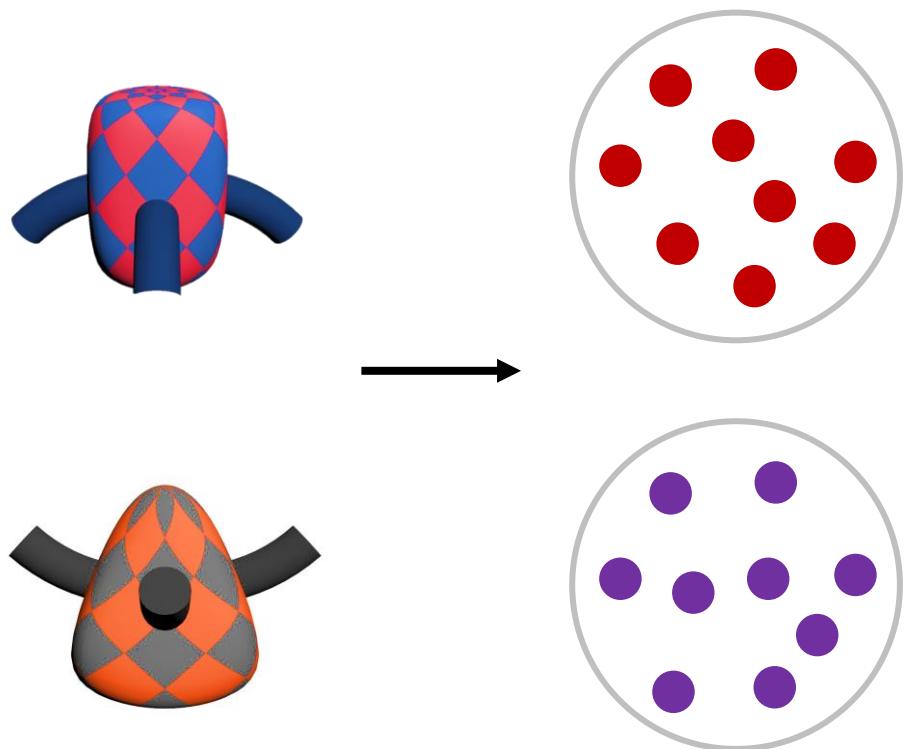
Lecturer: Saman Abbaspoor

The structure of correlations,
latent space, and
information processing

Neural Decoding and Optimal Decoding Axis

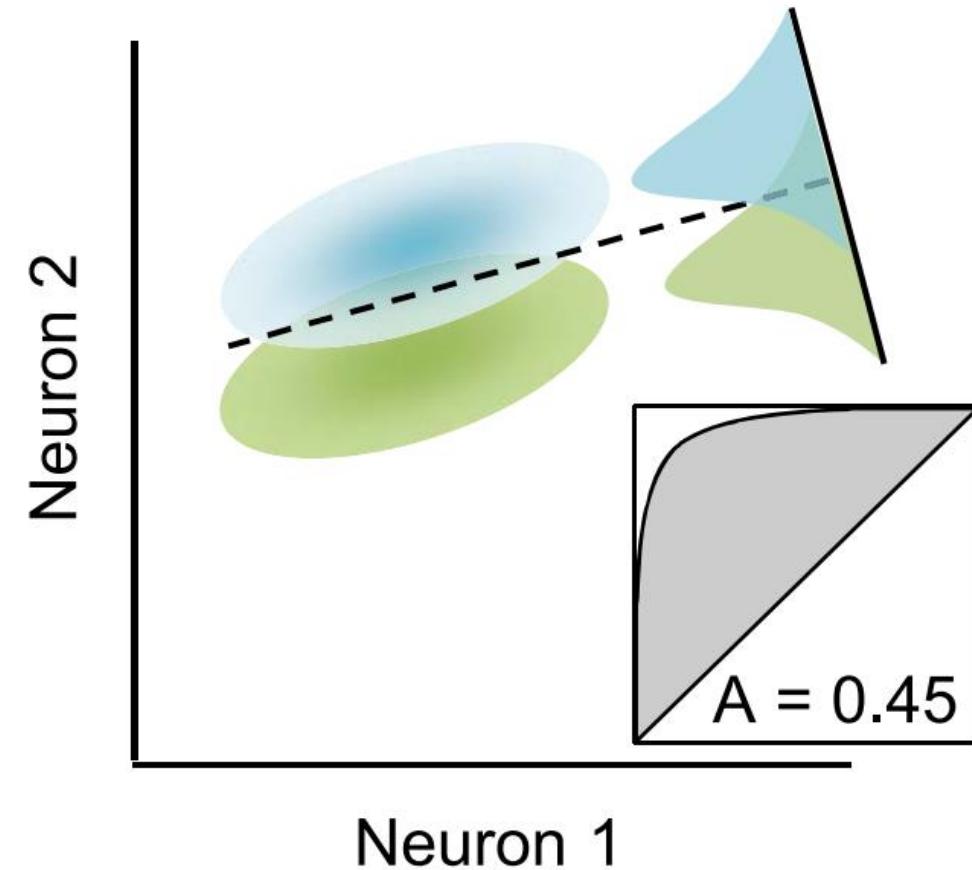
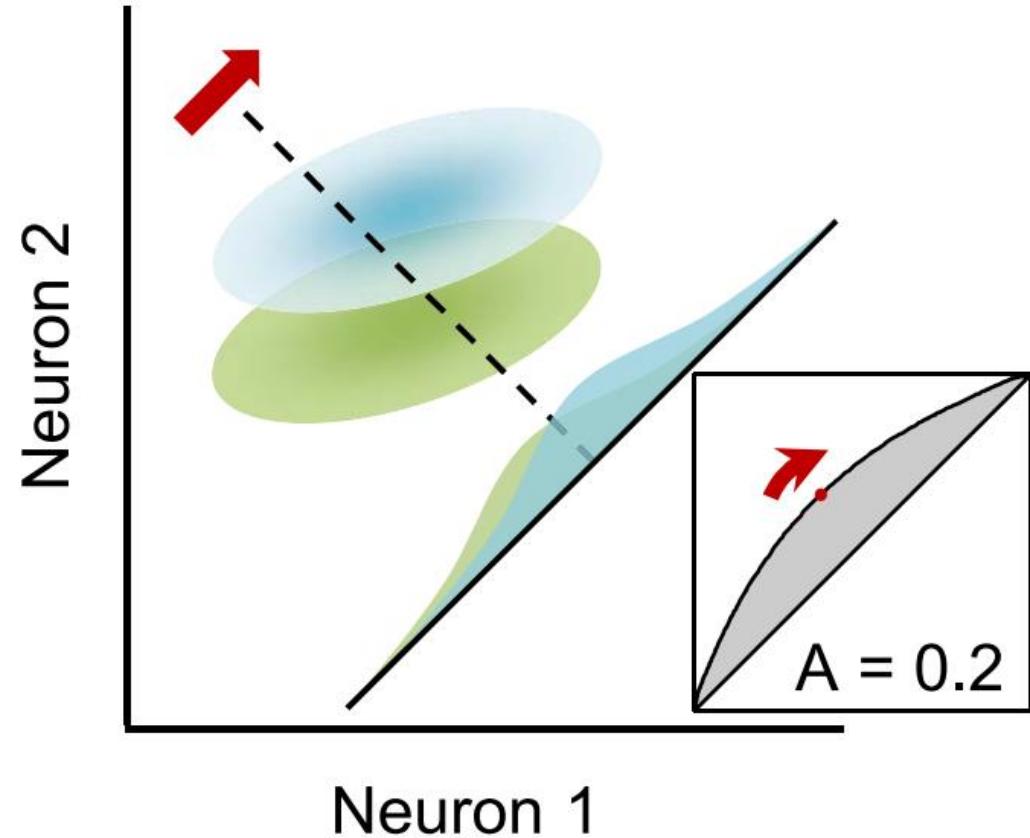


Neural Decoding and Optimal Decoding Axis



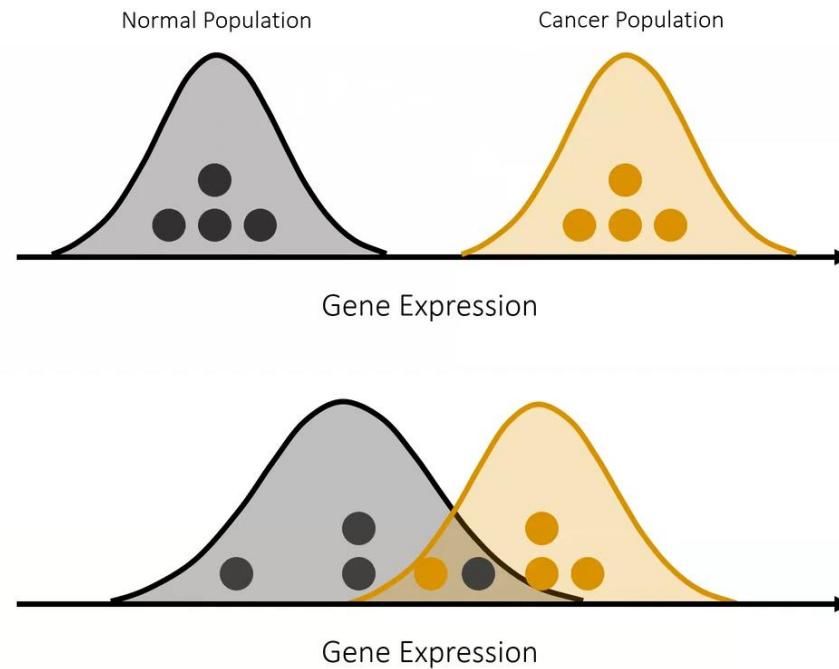
Optimal Decoding Subspace

Neural Decoding and Optimal Decoding Axis

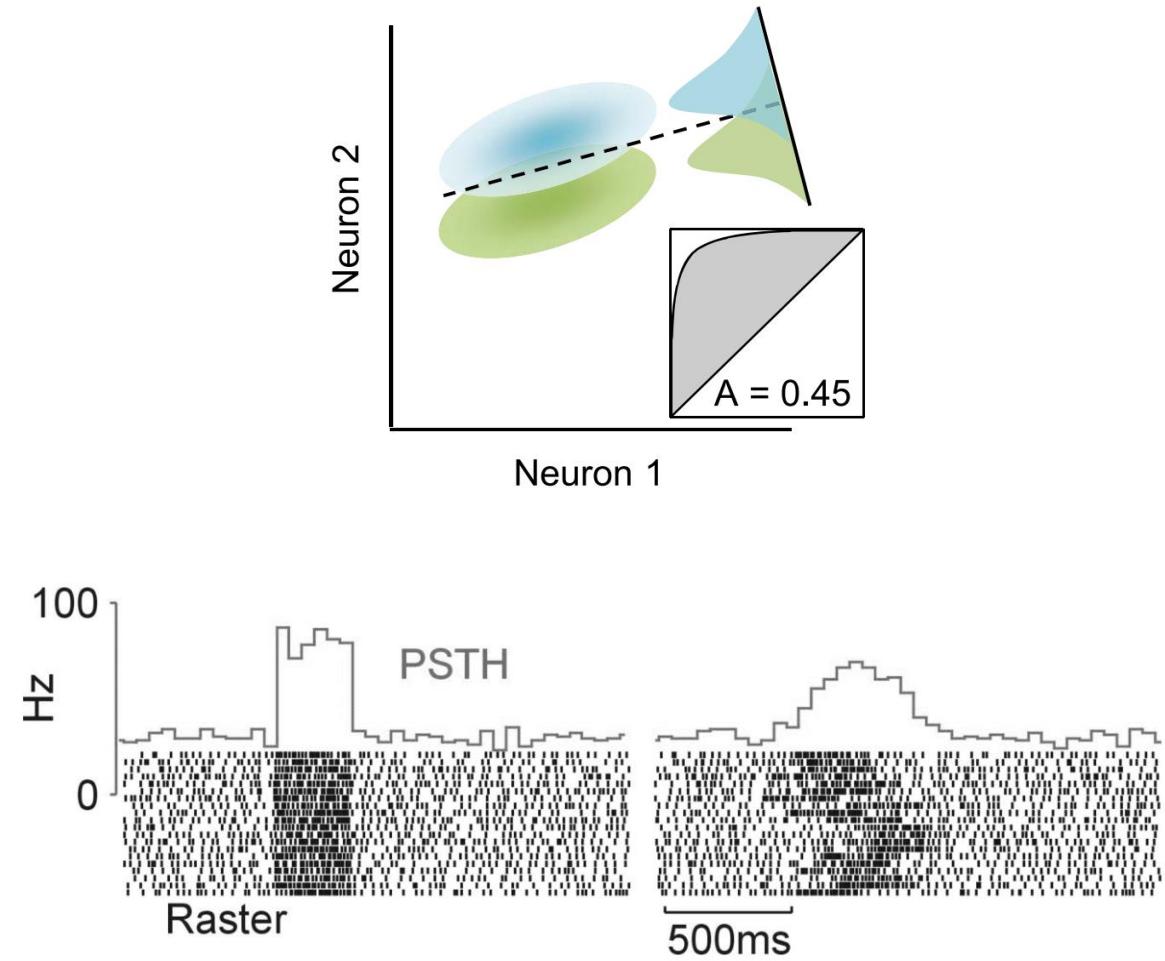
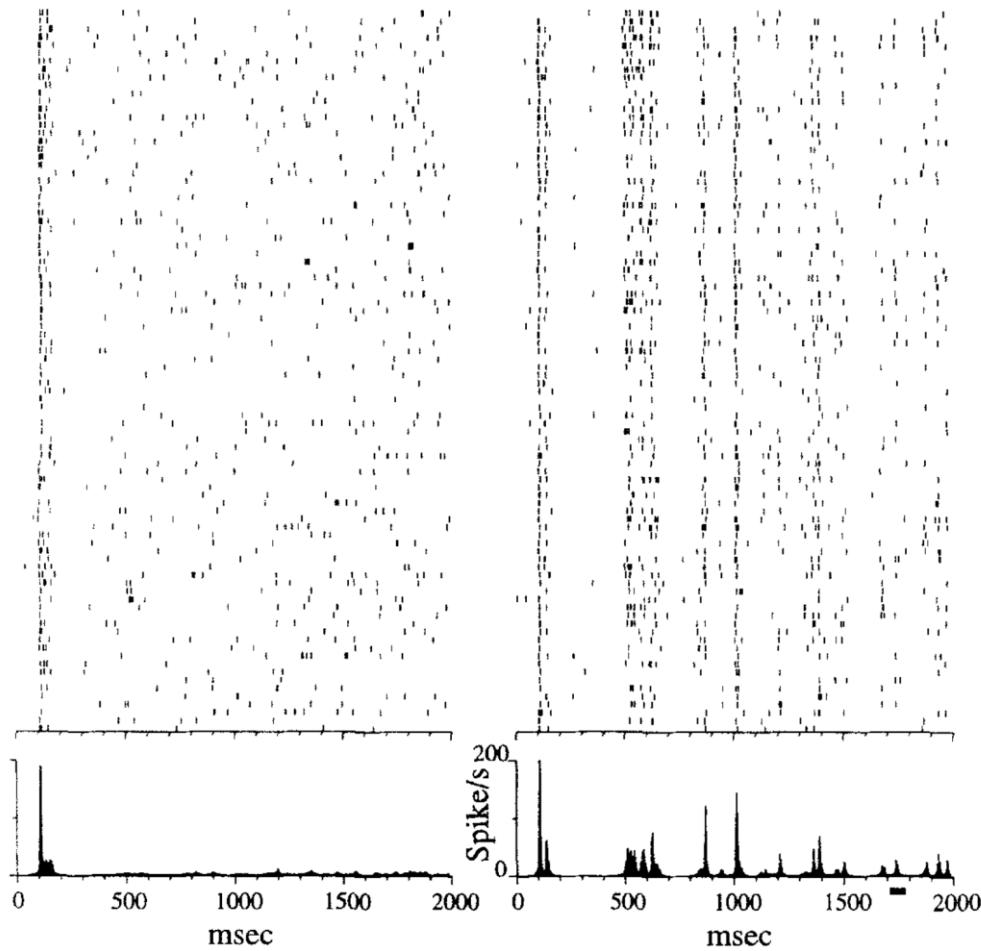


Signal Detection Theory and Decision Making

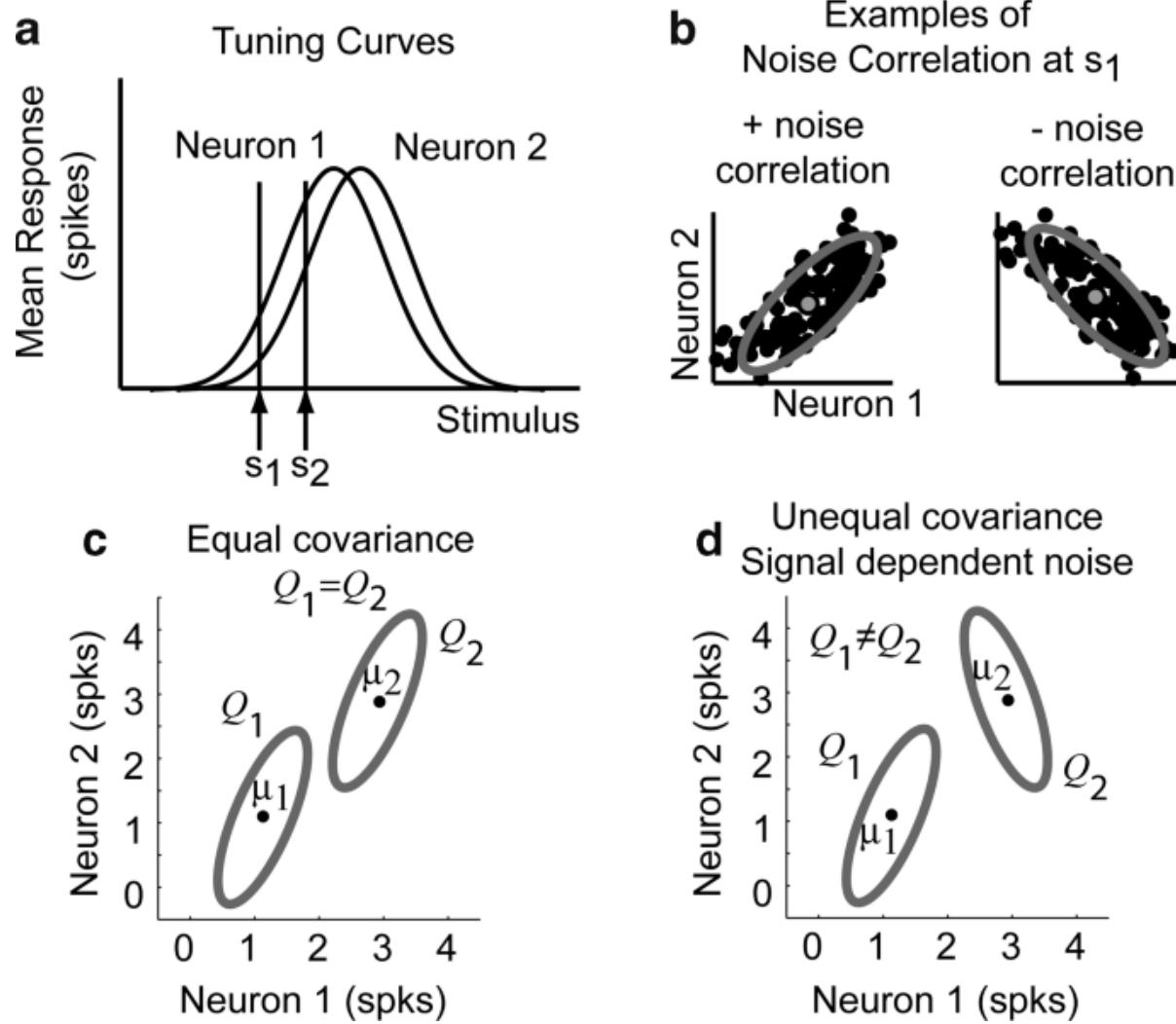
**Classification
Problem**



Trial to trial variability



Noise Correlations



Signal Correlations

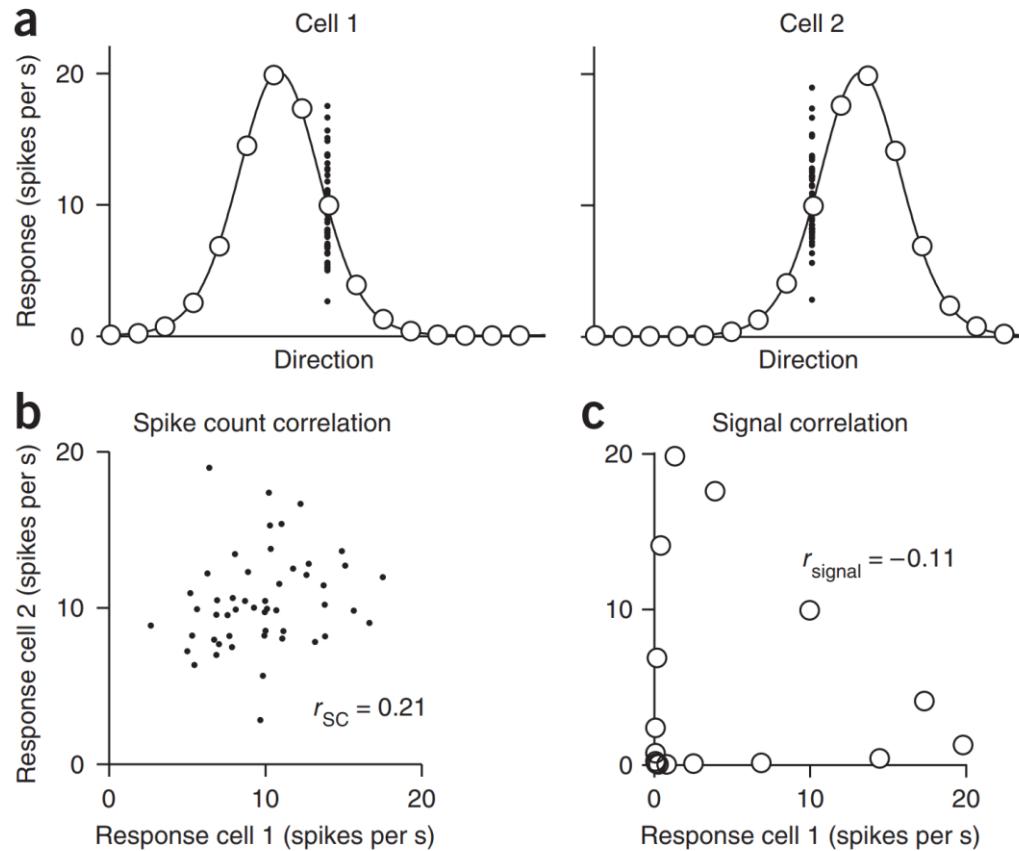
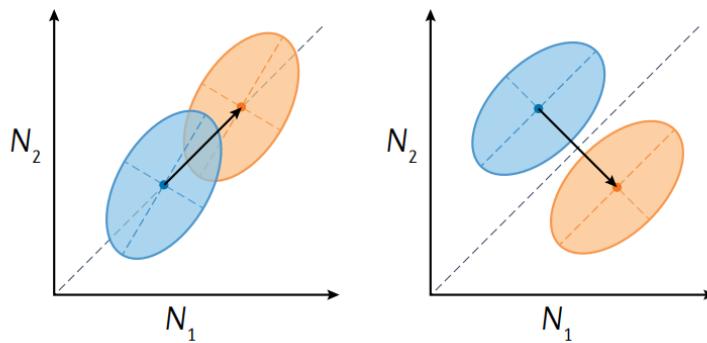


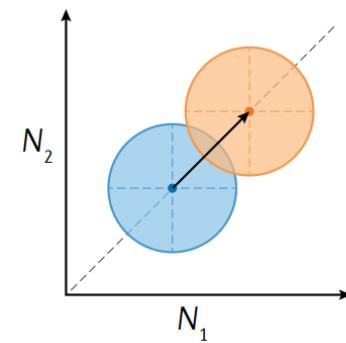
Figure 1 Types of pair-wise neuronal correlations. (a) Tuning curves for two hypothetical direction-selective neurons. Open circles show mean responses to different directions of motion and small points show responses to individual presentations of a stimulus at a particular direction. (b) Spike count or ‘noise’ correlation (r_{SC}) measures the correlation between fluctuations in responses to the same stimulus. Here, each point represents the response of the two neurons on one presentation of an individual stimulus. (c) Signal correlation (r_{signal}) measures the correlation between the two cells’ mean responses to different stimuli. Each point represents the mean response to a given direction of motion. Because the responses of cell 2 increase in a range of motion directions in which the responses of cell 1 decline, signal correlation is negative.

The structure of correlations and linear decodability

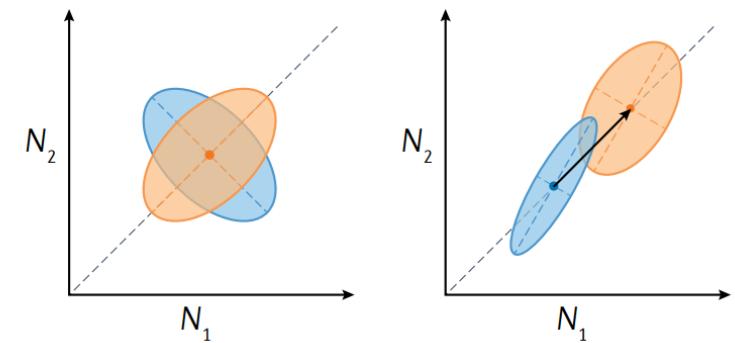
a Stimulus-independent noise correlations



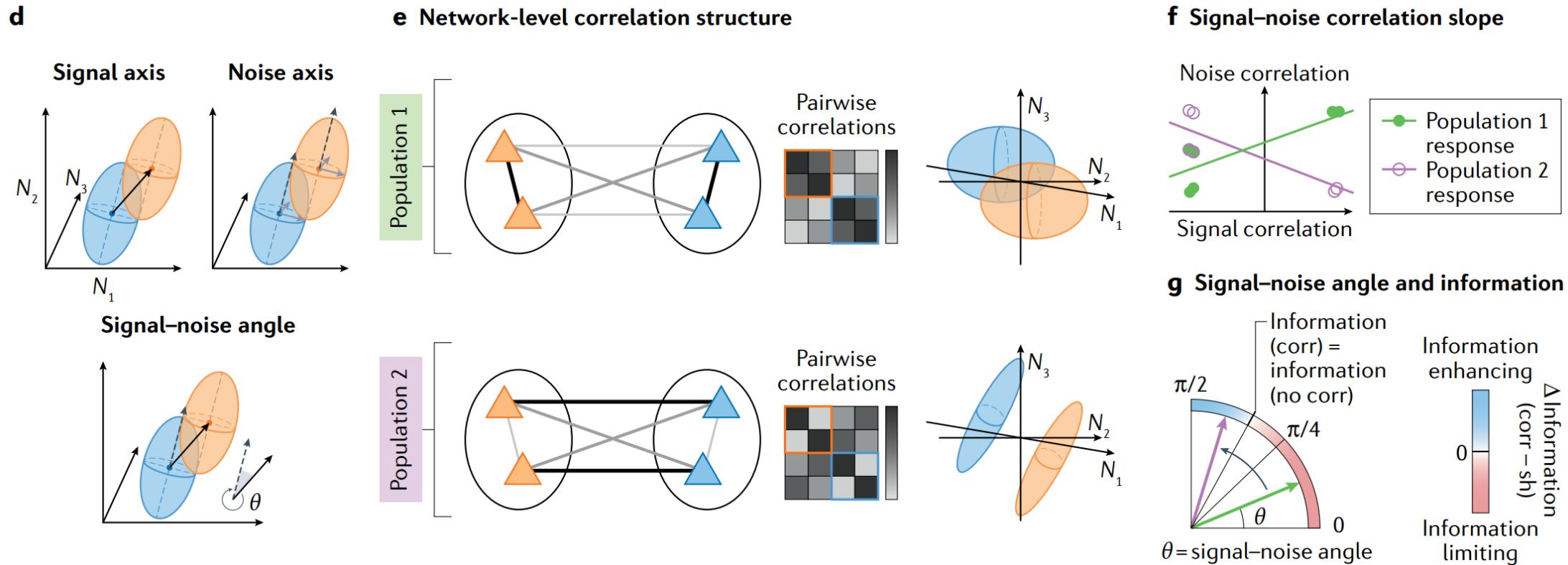
b Without noise correlations



c Stimulus-dependent noise correlations



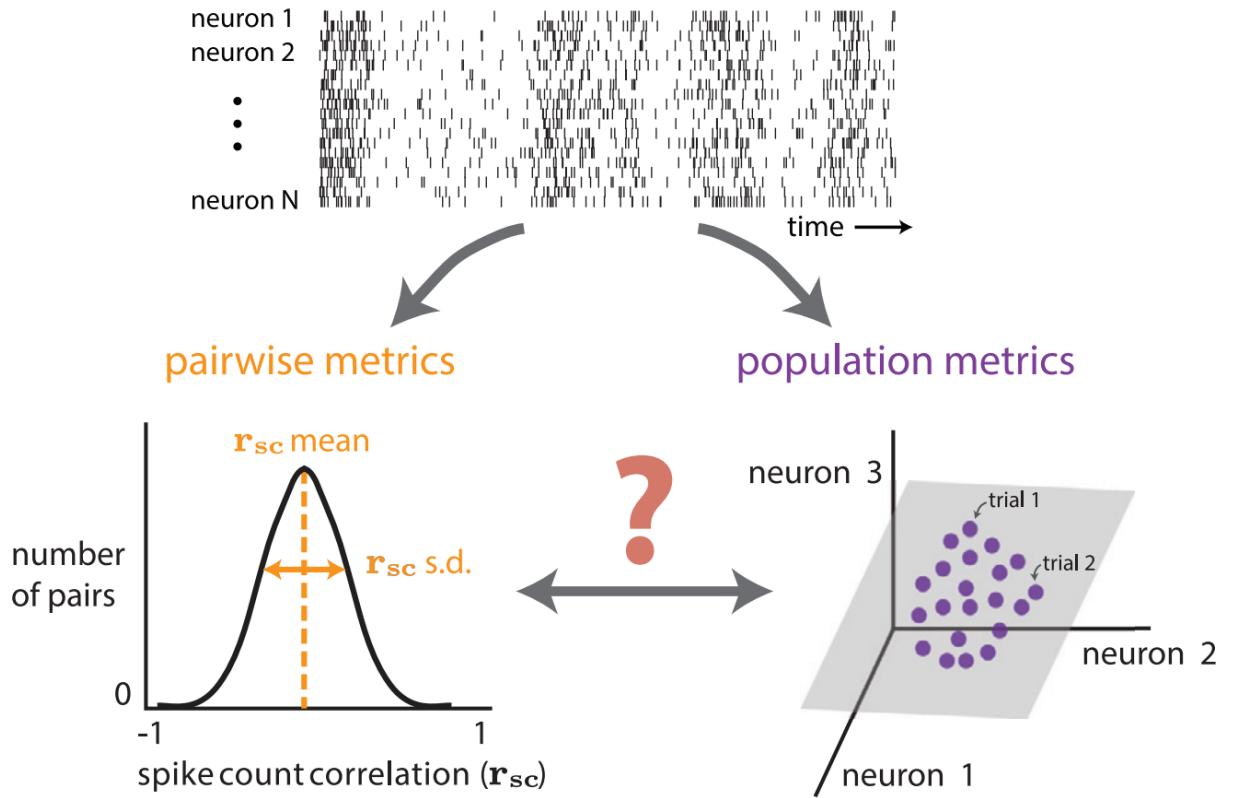
The structure of correlations and linear decodability



The structures and functions of correlations in neural population codes

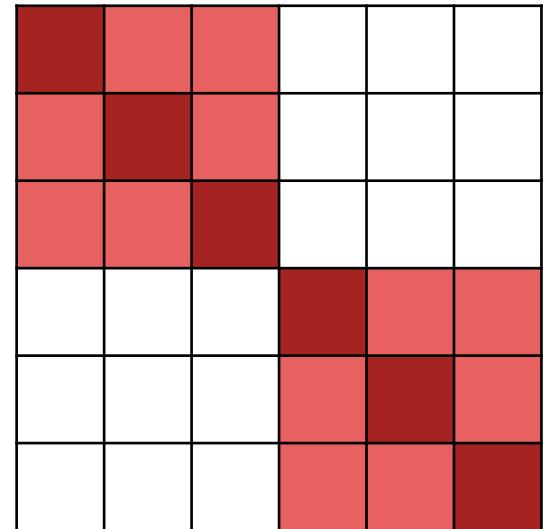
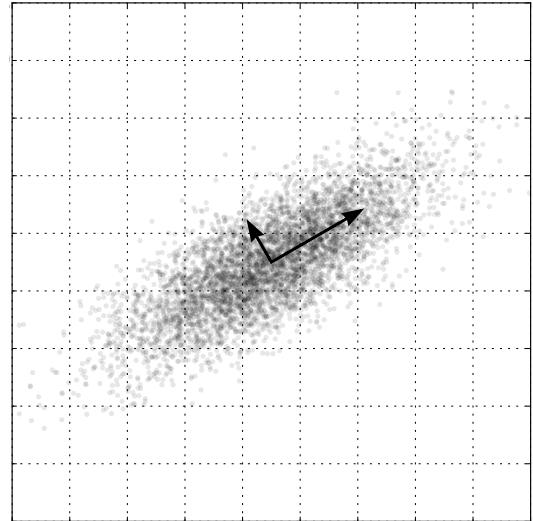
Panzeri et al., Nat Rev Neuro 2022

Neuronal correlations and dimensionality reduction



Principal component analysis (PCA)

Eigendecomposition of the
data covariance matrix



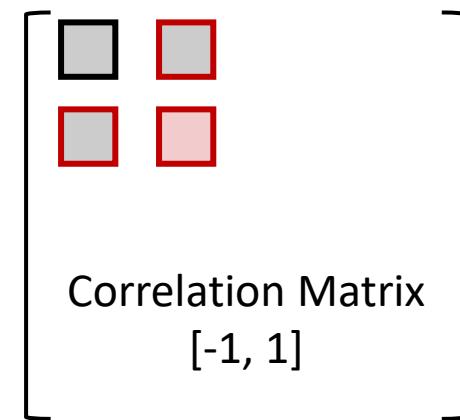
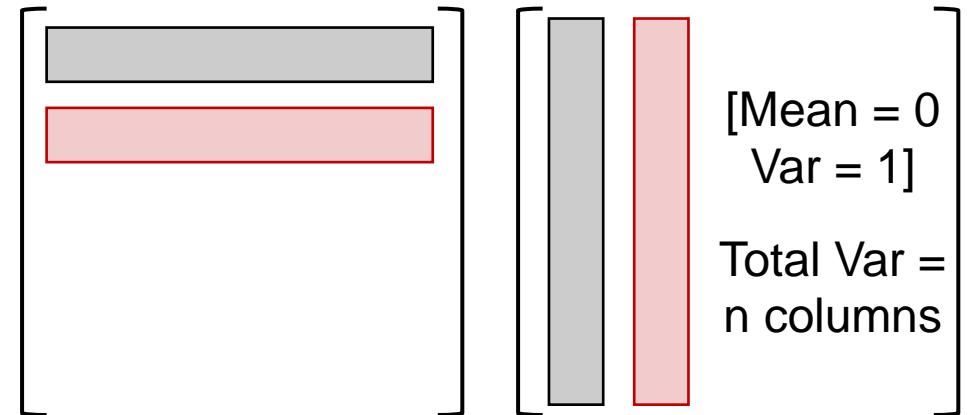
Correlation Matrix

$$Corr_{x,y} = \frac{\sum(x_j - \bar{x})(y_j - \bar{y})}{\sqrt{\sum(x_j - \bar{x})^2 \sum(y_j - \bar{y})^2}}$$

$$\bar{X} = \frac{1}{n} \sum_{j=1}^n X_j$$

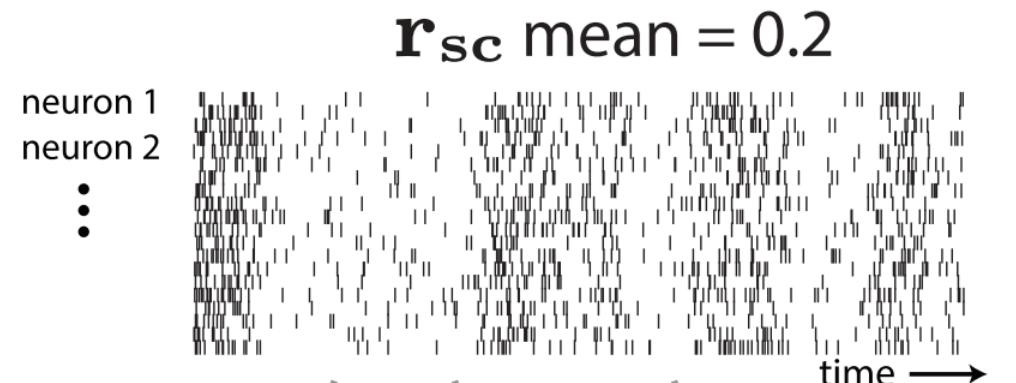
$$B = zscore(X) = \frac{x - \bar{x}}{\sigma}$$

$$C = B^T B$$

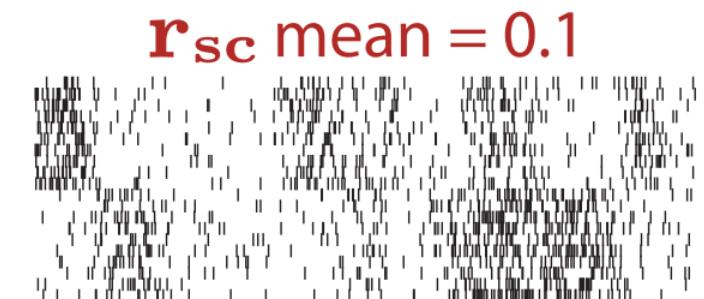
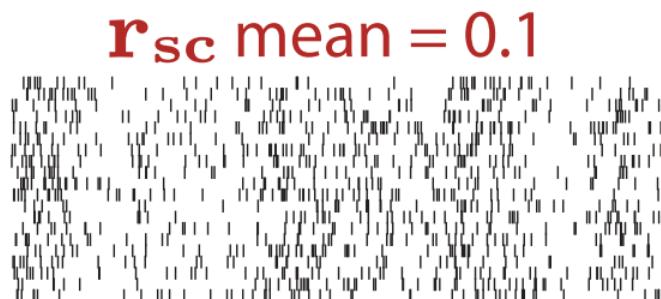
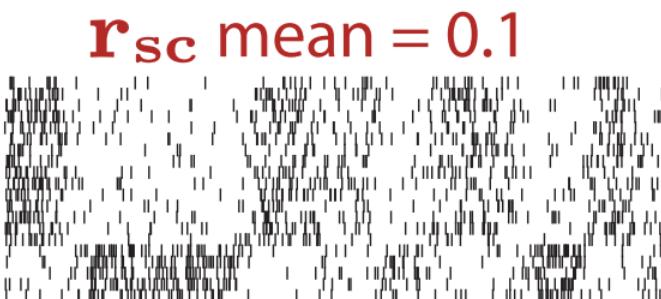


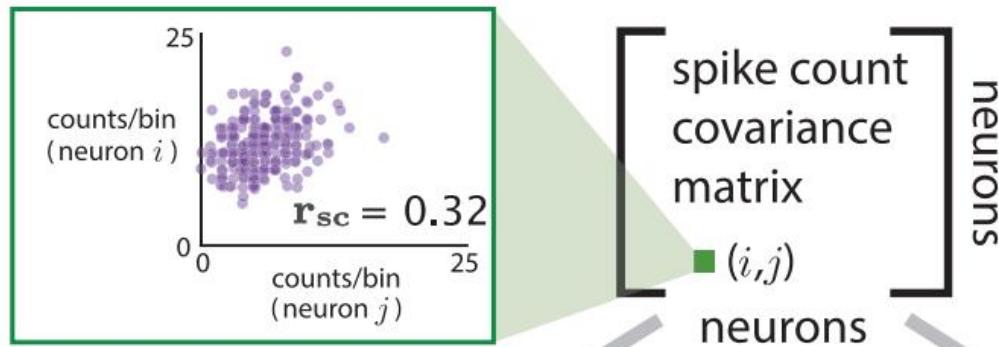
$$C = B^T B$$

condition 1:



condition 2:

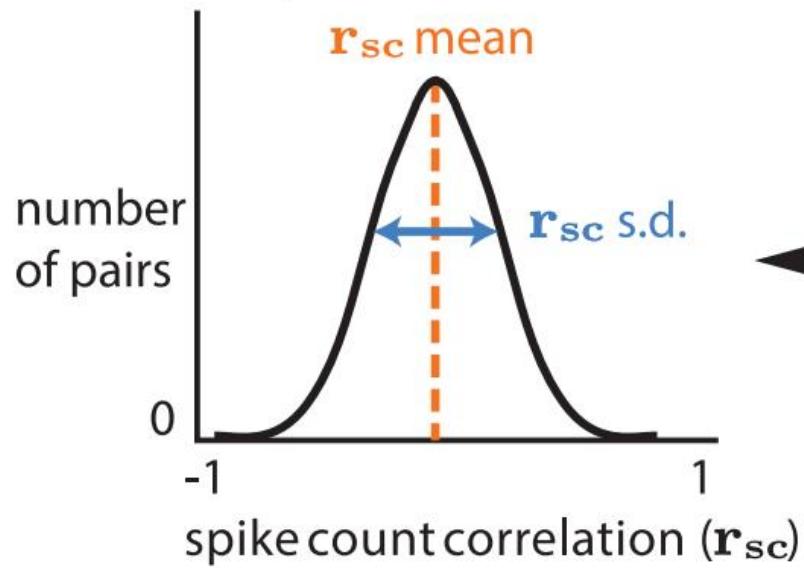




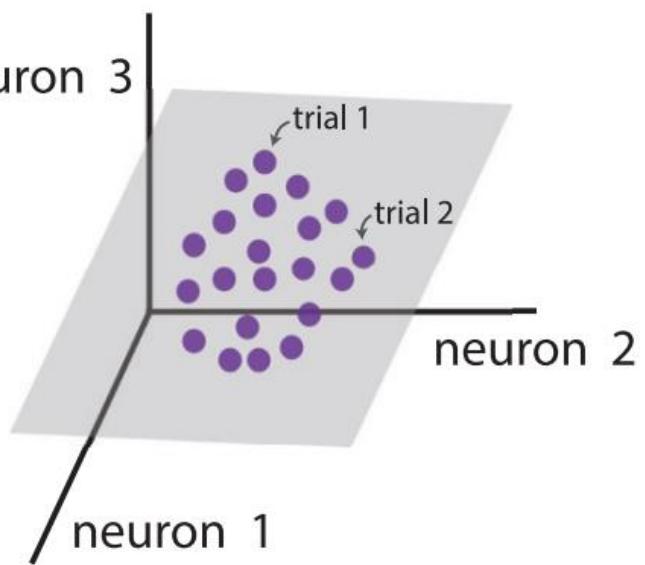
spike count covariance matrix
 $\boxed{\text{neurons}}$
 $\boxed{\text{neurons}}$

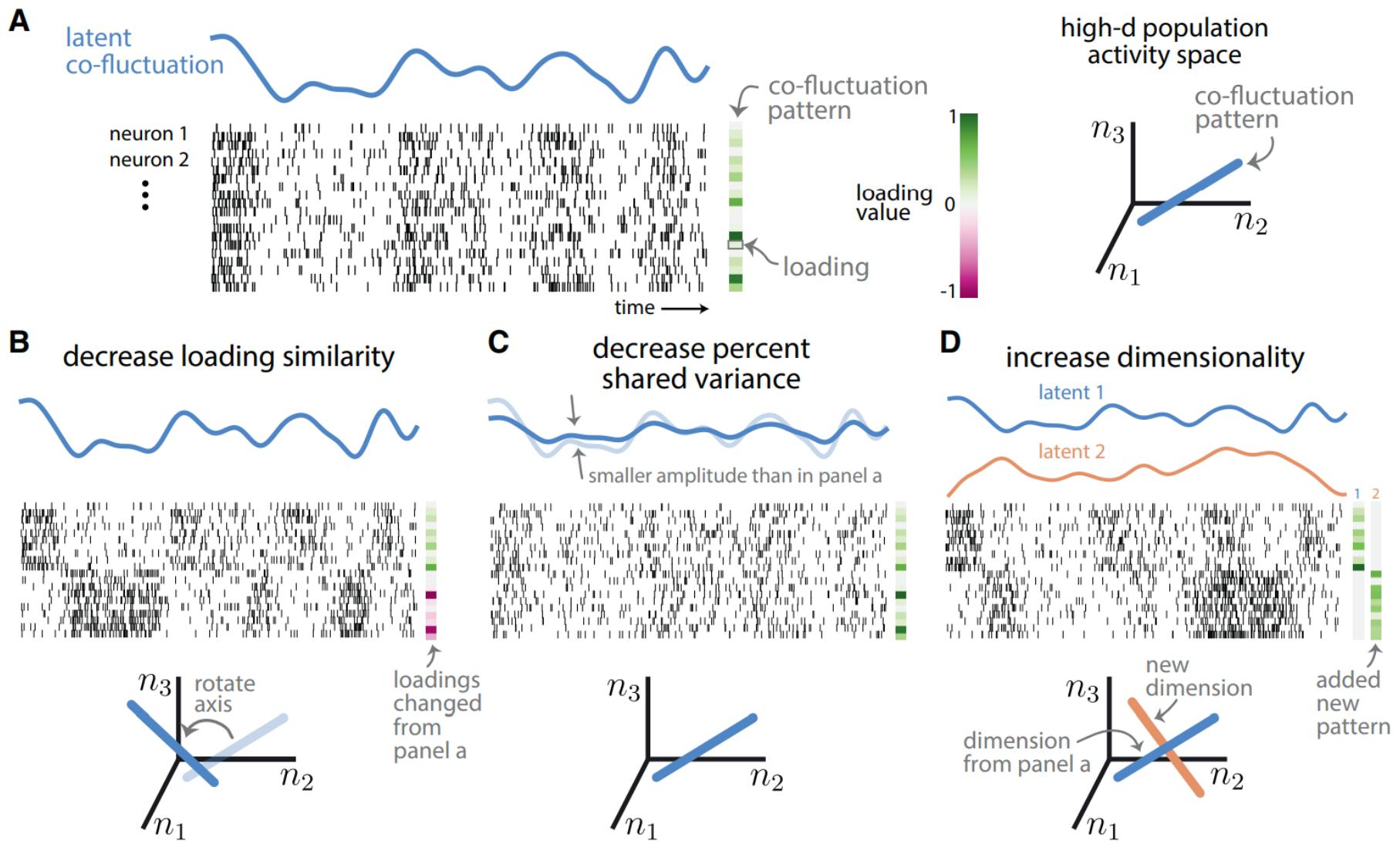
$\blacksquare (i,j)$

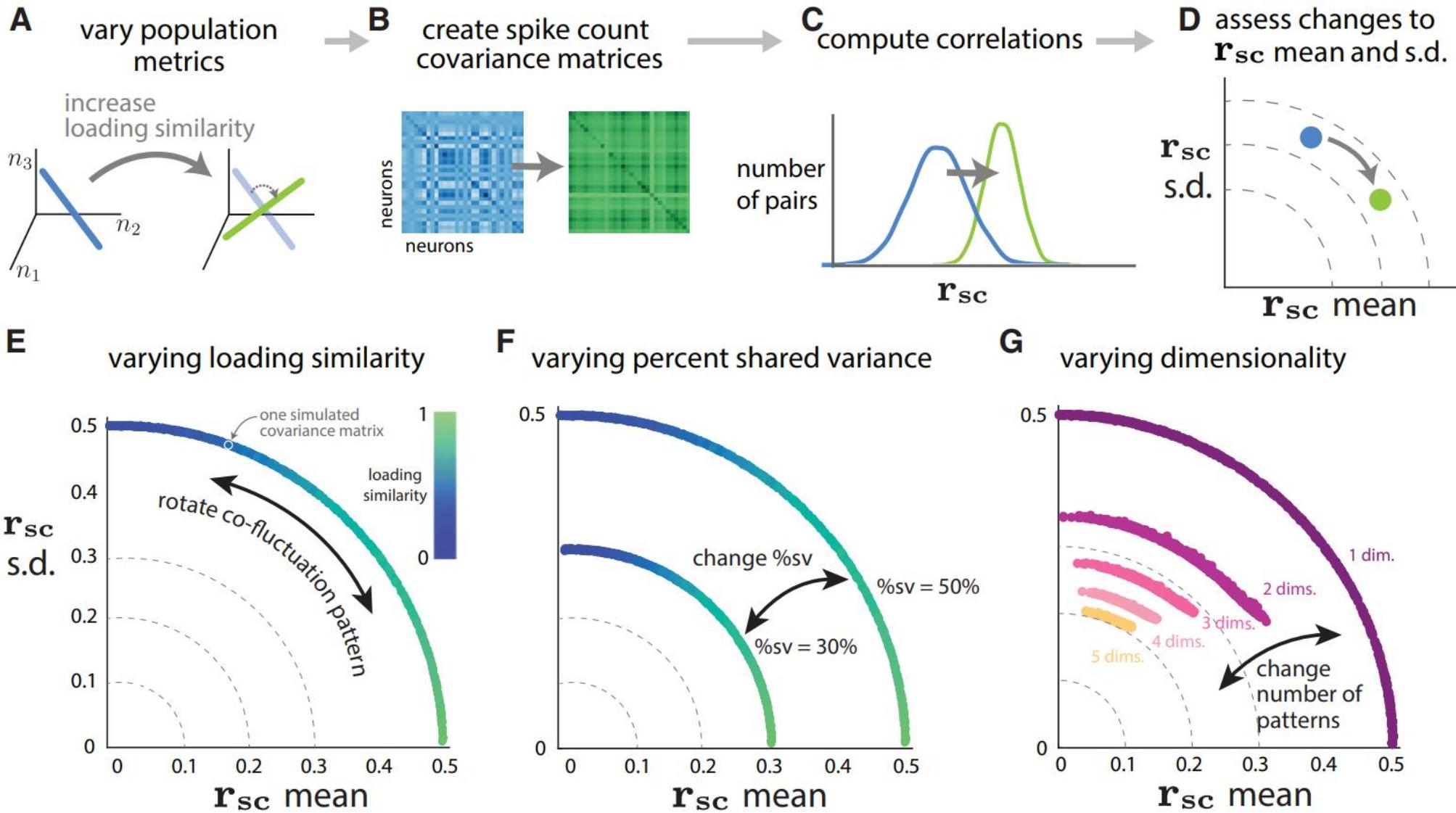
pairwise metrics



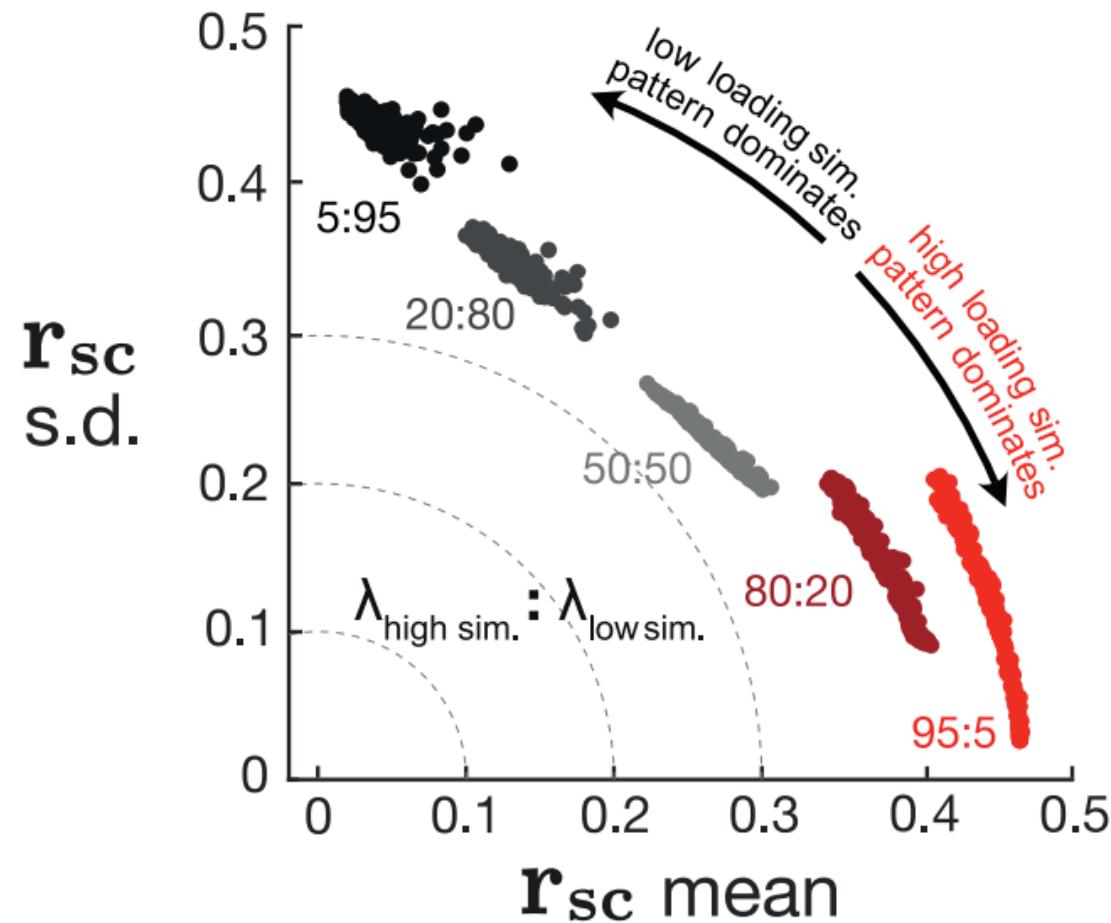
population metrics



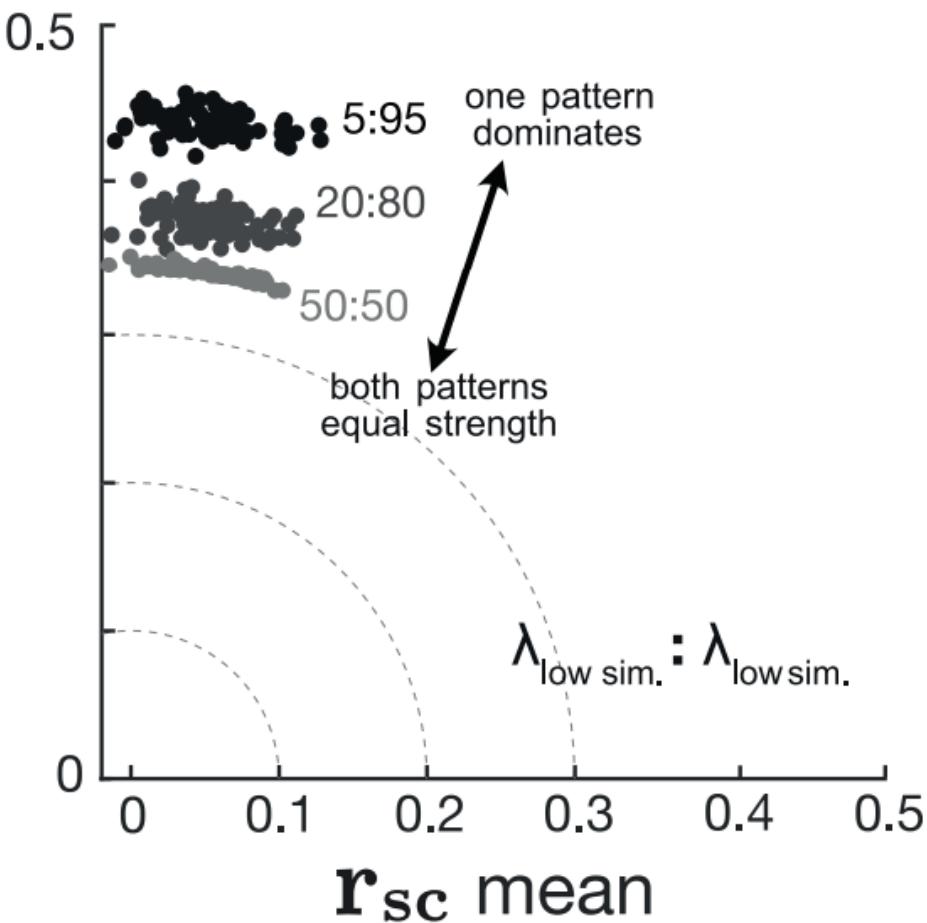




A pattern 1 has high loading sim.
pattern 2 has low loading sim.



B both patterns have low loading similarity



Summary of relationship between pairwise and population metrics

