

# Advances in Dimensionality Reduction

Nathan Mankovich



**ISP · Image & Signal Processing**  
*Universitat de València*

## DR Reduces the Feature Space

## DR Reduces the Feature Space

- ▶  $p$  samples

## DR Reduces the Feature Space

- ▶  $p$  samples
- ▶  $d$  dimensional ambient space

## DR Reduces the Feature Space

- ▶  $p$  samples
- ▶  $d$  dimensional ambient space
- ▶ Input dataset  $\mathcal{D}$  ( $d$  features)

## DR Reduces the Feature Space

- ▶  $p$  samples
- ▶  $d$  dimensional ambient space
- ▶ Input dataset  $\mathcal{D}$  ( $d$  features)
- ▶ Output: reduced dataset  $\mathcal{R}$  ( $k$  features)

## DR Reduces the Feature Space

- ▶  $p$  samples
- ▶  $d$  dimensional ambient space
- ▶ Input dataset  $\mathcal{D}$  ( $d$  features)
- ▶ Output: reduced dataset  $\mathcal{R}$  ( $k$  features)

**Dimensionality Reduction (DR)** maps  $d$ -dimensional features to  $k$ -dimensional features ( $k < d$ ).

## DR Reduces the Feature Space

- ▶  $p$  samples
- ▶  $d$  dimensional ambient space
- ▶ Input dataset  $\mathcal{D}$  ( $d$  features)
- ▶ Output: reduced dataset  $\mathcal{R}$  ( $k$  features)

**Dimensionality Reduction (DR)** maps  $d$ -dimensional features to  $k$ -dimensional features ( $k < d$ ).

**Question:** Why do we want to use dimensionality reduction?

# Dimensionality Reduction in Remote Sensing

# Dimensionality Reduction in Remote Sensing

Input a data cube with spectral, spatial, and temporal dimensions.

# Dimensionality Reduction in Remote Sensing

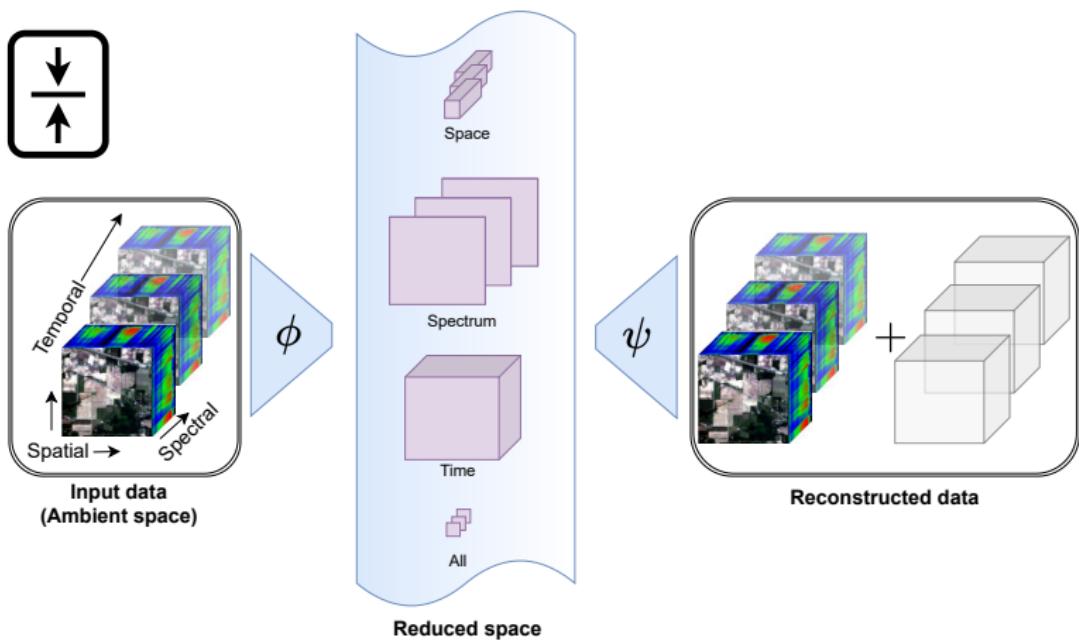
Input a data cube with spectral, spatial, and temporal dimensions.

# Dimensionality Reduction in Remote Sensing

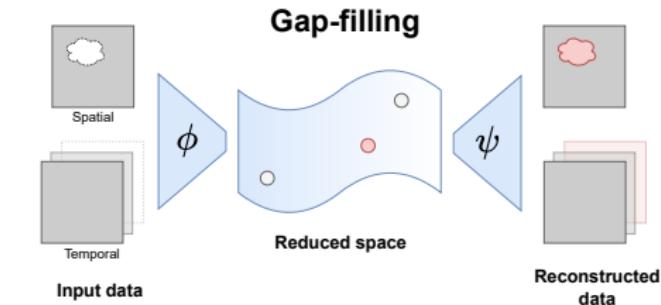
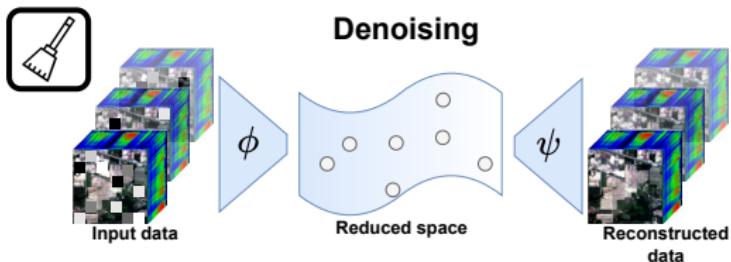
Input a data cube with spectral, spatial, and temporal dimensions.

- ▶ Compression
- ▶ Denoising
- ▶ Fusion
- ▶ Visualization
- ▶ Anomaly detection
- ▶ Improving predictions

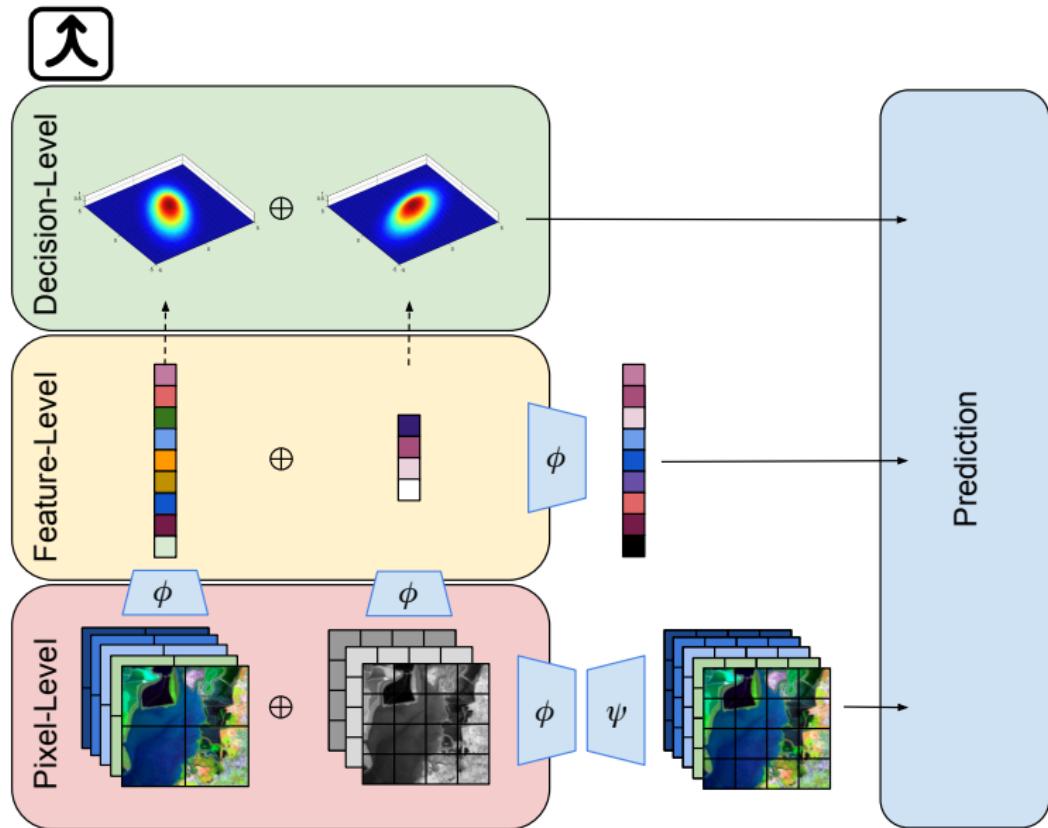
# Compression



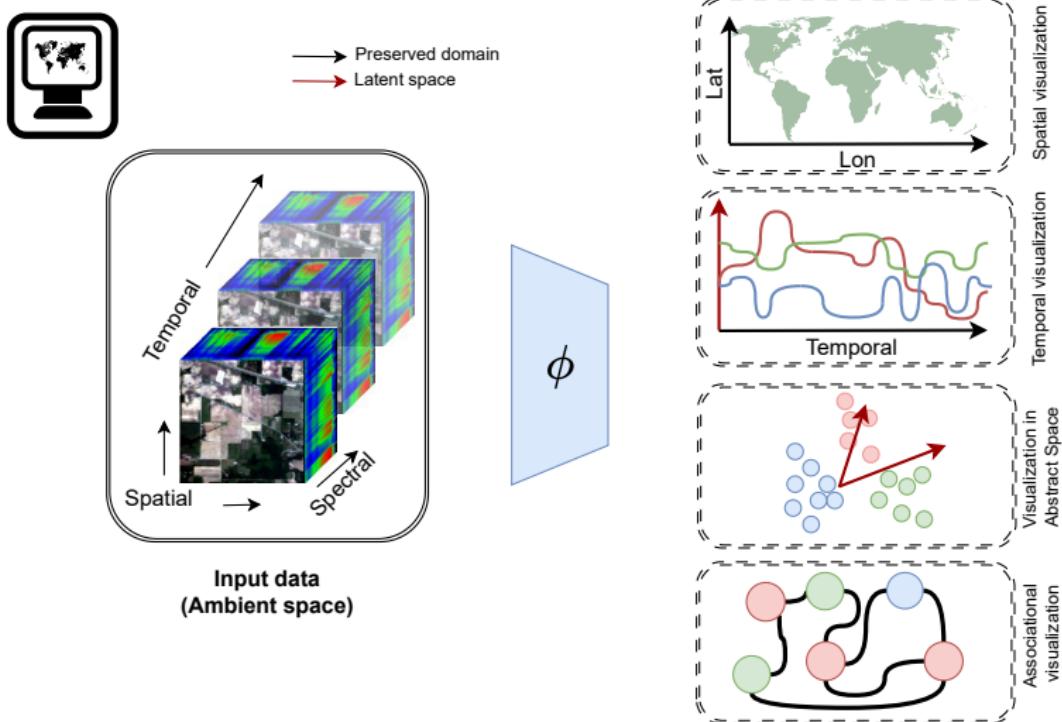
# Denoising



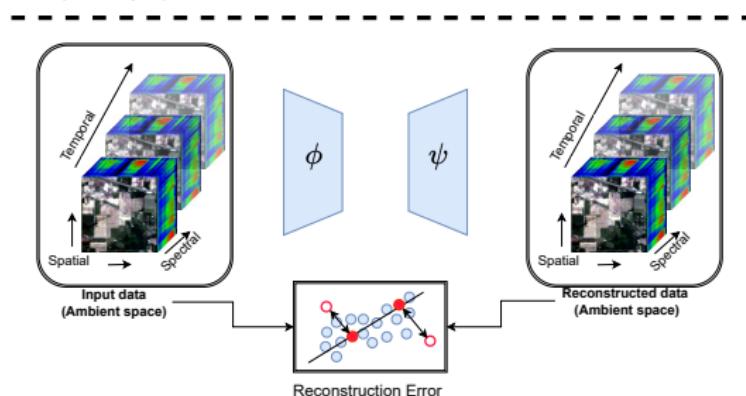
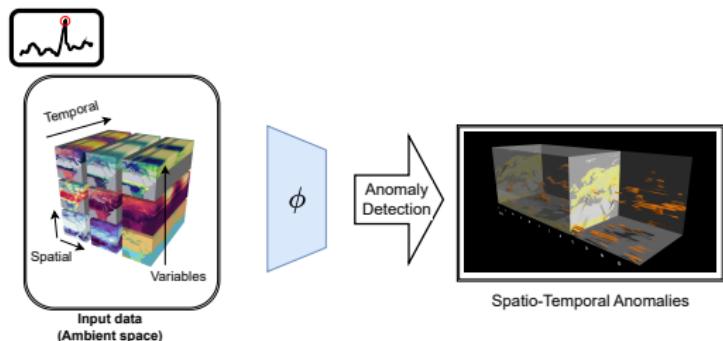
# Fusion



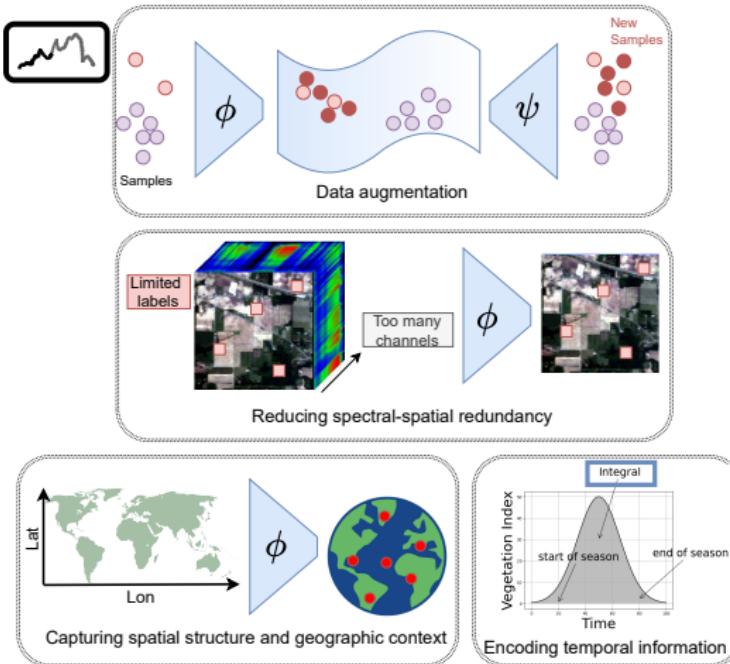
## Visualization



# Anomaly Detection



# Improved Predictions

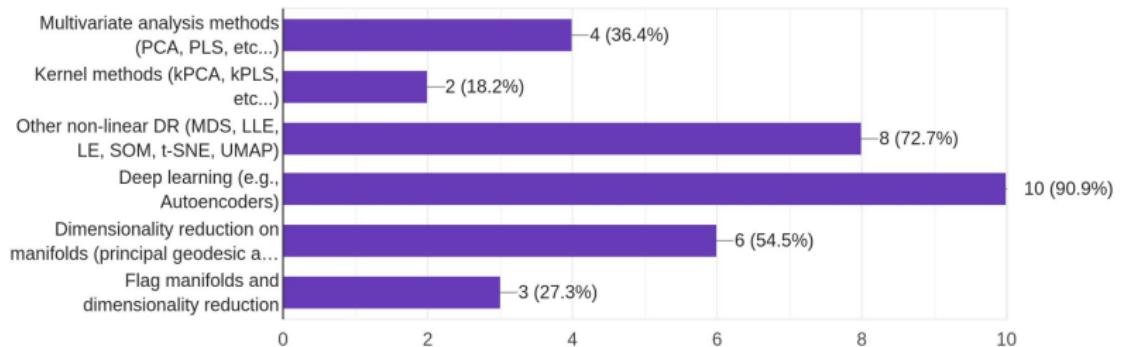


# Course Survey Results

# Survey Results

## Topics covered (choose up to 3)

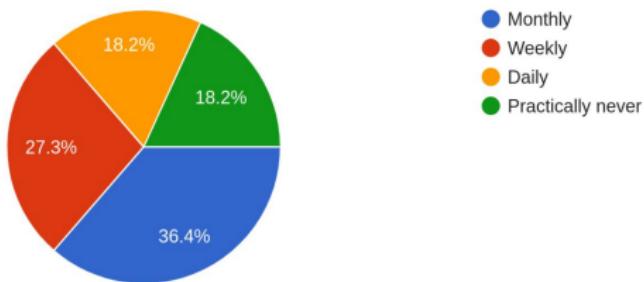
11 responses



# Survey Results

## How often do you use linear algebra

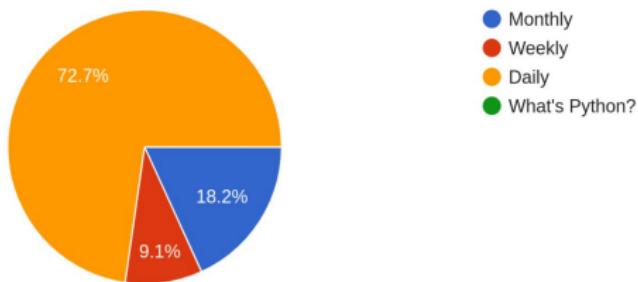
11 responses



# Survey Results

How often do you code in python?

11 responses



# Course Outline

1. Linear Algebra Review
2. Intro to DR
3. Linear Dimensionality Reduction
4. Nonlinear Dimensionality Reduction
5. Autoencoders

# Linear Algebra Review

*Further reading* [Strang, 2000]

# Outline

1. Vectors & Inner Products
2. Subspaces, Orthogonality, Projections
3. Matrix Decompositions
4. Gradient Descent

# Vectors & Inner Products

## Field Axioms

A field  $F$  is a set equipped with two operations (addition and multiplication) satisfying the following axioms for all  $a, b, c \in F$ :

# Field Axioms

A field  $F$  is a set equipped with two operations (addition and multiplication) satisfying the following axioms for all  $a, b, c \in F$ :

► **Associativity:**

- Addition:  $a + (b + c) = (a + b) + c$
- Multiplication:  $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

# Field Axioms

A field  $F$  is a set equipped with two operations (addition and multiplication) satisfying the following axioms for all  $a, b, c \in F$ :

► **Associativity:**

- Addition:  $a + (b + c) = (a + b) + c$
- Multiplication:  $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

► **Commutativity:**

- Addition:  $a + b = b + a$
- Multiplication:  $a \cdot b = b \cdot a$

# Field Axioms

A field  $F$  is a set equipped with two operations (addition and multiplication) satisfying the following axioms for all  $a, b, c \in F$ :

► **Associativity:**

- Addition:  $a + (b + c) = (a + b) + c$
- Multiplication:  $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

► **Commutativity:**

- Addition:  $a + b = b + a$
- Multiplication:  $a \cdot b = b \cdot a$

► **Identities:**

- Additive identity: there exists  $0 \in F$  such that  $a + 0 = a$
- Multiplicative identity: there exists  $1 \in F$ ,  $1 \neq 0$ , such that  $a \cdot 1 = a$

# Field Axioms

A field  $F$  is a set equipped with two operations (addition and multiplication) satisfying the following axioms for all  $a, b, c \in F$ :

► **Associativity:**

- Addition:  $a + (b + c) = (a + b) + c$
- Multiplication:  $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

► **Commutativity:**

- Addition:  $a + b = b + a$
- Multiplication:  $a \cdot b = b \cdot a$

► **Identities:**

- Additive identity: there exists  $0 \in F$  such that  $a + 0 = a$
- Multiplicative identity: there exists  $1 \in F$ ,  $1 \neq 0$ , such that  $a \cdot 1 = a$

► **Inverses:**

- Additive inverse: for every  $a \in F$ , there exists  $-a \in F$  such that  $a + (-a) = 0$
- Multiplicative inverse: for every  $a \neq 0$  in  $F$ , there exists  $a^{-1} \in F$  such that  $a \cdot a^{-1} = 1$

# Field Axioms

A field  $F$  is a set equipped with two operations (addition and multiplication) satisfying the following axioms for all  $a, b, c \in F$ :

► **Associativity:**

- Addition:  $a + (b + c) = (a + b) + c$
- Multiplication:  $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

► **Commutativity:**

- Addition:  $a + b = b + a$
- Multiplication:  $a \cdot b = b \cdot a$

► **Identities:**

- Additive identity: there exists  $0 \in F$  such that  $a + 0 = a$
- Multiplicative identity: there exists  $1 \in F$ ,  $1 \neq 0$ , such that  $a \cdot 1 = a$

► **Inverses:**

- Additive inverse: for every  $a \in F$ , there exists  $-a \in F$  such that  $a + (-a) = 0$
- Multiplicative inverse: for every  $a \neq 0$  in  $F$ , there exists  $a^{-1} \in F$  such that  $a \cdot a^{-1} = 1$

► **Distributivity:**  $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

The set of all real numbers (denoted  $\mathbb{R}$ ) is a Field

- ▶ Addition and multiplication operations
- ▶ Additive identity is 0
- ▶ Multiplicative identity is 1

Exercise: Show that  $\mathbb{R}$  satisfies the axioms of a field, whereas  $\mathbb{Z}$  (the set of all integers) does not.

# Vector Space Axioms

A vector space over a field  $F$  satisfies the following axioms for all  $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$  and all scalars  $a, b \in F$ :

# Vector Space Axioms

A vector space over a field  $F$  satisfies the following axioms for all  $\mathbf{u}, \mathbf{v}, \mathbf{w} \in V$  and all scalars  $a, b \in F$ :

- ▶ **Associativity of addition:**  $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$
- ▶ **Commutativity of addition:**  $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$
- ▶ **Additive identity:** There exists  $\mathbf{0} \in V$  such that  $\mathbf{v} + \mathbf{0} = \mathbf{v}$
- ▶ **Additive inverse:** For each  $\mathbf{v} \in V$ , there exists  $-\mathbf{v} \in V$  such that  $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
- ▶ **Compatibility with scalar multiplication:**  $a(b\mathbf{v}) = (ab)\mathbf{v}$
- ▶ **Identity element of scalar multiplication:**  $1\mathbf{v} = \mathbf{v}$ , where 1 is the multiplicative identity in  $F$
- ▶ **Distributivity over vector addition:**  $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$
- ▶ **Distributivity over field addition:**  $(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$

## $\mathbb{R}^2$ is a Vector Space

$\mathbb{R}^2$  is a Vector Space over  $\mathbb{R}$ .

$$\mathbb{R}^2 = \left\{ \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} : a_1, a_2 \in \mathbb{R} \right\}$$

Exercise: Show that  $\mathbb{R}^2$  satisfies the axioms of a vector space.

# Dot Product & Friends

# Dot Product & Friends

## ► Transpose

$$\mathbf{a}^\top = [a_1 \quad a_2]$$

# Dot Product & Friends

## ► Transpose

$$\mathbf{a}^\top = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

## ► Dot product

$$\mathbf{a}^\top \mathbf{b} = a_1 b_1 + a_2 b_2$$

# Dot Product & Friends

## ► Transpose

$$\mathbf{a}^\top = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

## ► Dot product

$$\mathbf{a}^\top \mathbf{b} = a_1 b_1 + a_2 b_2$$

*Two vectors are called “orthogonal” if their dot product is zero.*

# Dot Product & Friends

- ▶ **Transpose**

$$\mathbf{a}^\top = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

- ▶ **Dot product**

$$\mathbf{a}^\top \mathbf{b} = a_1 b_1 + a_2 b_2$$

*Two vectors are called “orthogonal” if their dot product is zero.*

- ▶ **2-Norm**

$$\|\mathbf{a}\|_2 = \sqrt{\mathbf{a}^\top \mathbf{a}}$$

# Dot Product & Friends

- ▶ **Transpose**

$$\mathbf{a}^\top = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

- ▶ **Dot product**

$$\mathbf{a}^\top \mathbf{b} = a_1 b_1 + a_2 b_2$$

*Two vectors are called “orthogonal” if their dot product is zero.*

- ▶ **2-Norm**

$$\|\mathbf{a}\|_2 = \sqrt{\mathbf{a}^\top \mathbf{a}}$$

- ▶ **Distance**

$$\|\mathbf{a} - \mathbf{b}\|_2$$

# Dot Product & Friends

- ▶ **Transpose**

$$\mathbf{a}^\top = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

- ▶ **Dot product**

$$\mathbf{a}^\top \mathbf{b} = a_1 b_1 + a_2 b_2$$

*Two vectors are called “orthogonal” if their dot product is zero.*

- ▶ **2-Norm**

$$\|\mathbf{a}\|_2 = \sqrt{\mathbf{a}^\top \mathbf{a}}$$

- ▶ **Distance**

$$\|\mathbf{a} - \mathbf{b}\|_2$$

- ▶ **Angle between vectors**

$$\cos(\theta) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2}$$

Span

# Span

The *span* of two vectors  $\mathbf{a}, \mathbf{b}$  is the set of all vectors that can be written as a linear combination of  $\mathbf{a}$  and  $\mathbf{b}$

# Span

The *span of two vectors*  $\mathbf{a}, \mathbf{b}$  is the set of all vectors that can be written as a linear combination of  $\mathbf{a}$  and  $\mathbf{b}$

$$\text{span}(\mathbf{a}, \mathbf{b}) = \alpha\mathbf{a} + \beta\mathbf{b} : \alpha, \beta \in \mathbb{R}$$

# Linear Independence

A set of vectors  $\{v_1, v_2, \dots, v_k\}$  in a vector space  $V$  is **linearly independent** if:

$$c_1v_1 + c_2v_2 + \dots + c_kv_k = 0 \quad \Rightarrow \quad c_1 = c_2 = \dots = c_k = 0$$

# Linear Independence

A set of vectors  $\{v_1, v_2, \dots, v_k\}$  in a vector space  $V$  is **linearly independent** if:

$$c_1v_1 + c_2v_2 + \dots + c_kv_k = 0 \quad \Rightarrow \quad c_1 = c_2 = \dots = c_k = 0$$

In other words, the only solution to the linear combination equaling zero is the **trivial solution**.

# Linear Independence

A set of vectors  $\{v_1, v_2, \dots, v_k\}$  in a vector space  $V$  is **linearly independent** if:

$$c_1v_1 + c_2v_2 + \dots + c_kv_k = 0 \quad \Rightarrow \quad c_1 = c_2 = \dots = c_k = 0$$

In other words, the only solution to the linear combination equaling zero is the **trivial solution**.

# Linear Independence

A set of vectors  $\{v_1, v_2, \dots, v_k\}$  in a vector space  $V$  is **linearly independent** if:

$$c_1v_1 + c_2v_2 + \dots + c_kv_k = 0 \quad \Rightarrow \quad c_1 = c_2 = \dots = c_k = 0$$

In other words, the only solution to the linear combination equaling zero is the **trivial solution**.

**Example:**

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

are linearly independent in  $\mathbb{R}^2$ , since neither can be written as a multiple of the other.

# What is a Basis?

# What is a Basis?

A **basis** of a vector space  $V$  is a set of vectors  $\{v_1, v_2, \dots, v_n\} \subset V$  such that:

1. The vectors are **linearly independent**.

# What is a Basis?

A **basis** of a vector space  $V$  is a set of vectors  $\{v_1, v_2, \dots, v_n\} \subset V$  such that:

1. The vectors are **linearly independent**.
2. The vectors **span**  $V$ , i.e., every vector in  $V$  can be written as a linear combination of  $v_1, \dots, v_n$ .

# What is a Basis?

A **basis** of a vector space  $V$  is a set of vectors  $\{v_1, v_2, \dots, v_n\} \subset V$  such that:

1. The vectors are **linearly independent**.
2. The vectors **span**  $V$ , i.e., every vector in  $V$  can be written as a linear combination of  $v_1, \dots, v_n$ .

# What is a Basis?

A **basis** of a vector space  $V$  is a set of vectors  $\{v_1, v_2, \dots, v_n\} \subset V$  such that:

1. The vectors are **linearly independent**.
2. The vectors **span**  $V$ , i.e., every vector in  $V$  can be written as a linear combination of  $v_1, \dots, v_n$ .

**Example:** The standard basis for  $\mathbb{R}^3$  is:

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

# What is a Basis?

A **basis** of a vector space  $V$  is a set of vectors  $\{v_1, v_2, \dots, v_n\} \subset V$  such that:

1. The vectors are **linearly independent**.
2. The vectors **span**  $V$ , i.e., every vector in  $V$  can be written as a linear combination of  $v_1, \dots, v_n$ .

**Example:** The standard basis for  $\mathbb{R}^3$  is:

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

# What is a Basis?

A **basis** of a vector space  $V$  is a set of vectors  $\{v_1, v_2, \dots, v_n\} \subset V$  such that:

1. The vectors are **linearly independent**.
2. The vectors **span**  $V$ , i.e., every vector in  $V$  can be written as a linear combination of  $v_1, \dots, v_n$ .

**Example:** The standard basis for  $\mathbb{R}^3$  is:

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

The **dimension** of a vector space  $V$  is the number of elements in a basis for  $V$ .

# Subspaces, Orthogonality, Projections

# Matrices

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

# Matrices

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

- ▶ The **row space** of  $\mathbf{A}$  (denoted  $\text{row}(\mathbf{A})$ ) is the set of linear combinations of rows.

# Matrices

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

- ▶ The **row space** of  $\mathbf{A}$  (denoted  $\text{row}(\mathbf{A})$ ) is the set of linear combinations of rows.
- ▶ The **column space** of  $\mathbf{A}$  (denoted  $\text{col}(\mathbf{A})$ ) is the set of linear combinations of columns.

# Matrices

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

- ▶ The **row space** of  $\mathbf{A}$  (denoted  $\text{row}(\mathbf{A})$ ) is the set of linear combinations of rows.
- ▶ The **column space** of  $\mathbf{A}$  (denoted  $\text{col}(\mathbf{A})$ ) is the set of linear combinations of columns.
- ▶ The **rank** of  $\mathbf{A}$  is the dimension of its column space.

# Matrices

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

- ▶ The **row space** of  $\mathbf{A}$  (denoted  $\text{row}(\mathbf{A})$ ) is the set of linear combinations of rows.
- ▶ The **column space** of  $\mathbf{A}$  (denoted  $\text{col}(\mathbf{A})$ ) is the set of linear combinations of columns.
- ▶ The **rank** of  $\mathbf{A}$  is the dimension of its column space.
- ▶ The **trace** of  $\mathbf{A}$  is the sum of the diagonal entries

$$\text{tr}(\mathbf{A}) = a_{11} + a_{22}$$

# Matrices

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

- ▶ The **row space** of  $\mathbf{A}$  (denoted  $\text{row}(\mathbf{A})$ ) is the set of linear combinations of rows.
- ▶ The **column space** of  $\mathbf{A}$  (denoted  $\text{col}(\mathbf{A})$ ) is the set of linear combinations of columns.
- ▶ The **rank** of  $\mathbf{A}$  is the dimension of its column space.
- ▶ The **trace** of  $\mathbf{A}$  is the sum of the diagonal entries

$$\text{tr}(\mathbf{A}) = a_{11} + a_{22}$$

- ▶ The Frobenius norm of  $\mathbf{A}$  is

$$\|\mathbf{A}\|_F = \sqrt{\text{tr}(\mathbf{A}^\top \mathbf{A})}$$

# Matrix multiplication

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

# Matrix multiplication

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ b_{11} \\ b_{21} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \\ b_{12} \\ b_{22} \end{bmatrix}$$

# Projections

# Projections

Projection of  $\mathbf{b}$  onto  $\text{col}(\mathbf{A})$ :

$$\Pi_{\mathbf{A}}(\mathbf{b}) = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

# Projections

Projection of  $\mathbf{b}$  onto  $\text{col}(\mathbf{A})$ :

$$\Pi_{\mathbf{A}}(\mathbf{b}) = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

*Exercise:* If the columns of  $\mathbf{A}$  are orthogonal, show that the projection onto the column space of  $\mathbf{A}$  is  $\mathbf{A}\mathbf{A}^T$ .

# Lab 1: Vectors & Matrices

*Go to Lab1\_VectorsMatrices.ipynb*

# Matrix Decompositions

# Singular Value Decomposition (SVD)

# Singular Value Decomposition (SVD)

SVD of matrix  $\mathbf{A} \in \mathbb{R}^{p \times d}$  of rank  $r$ :

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$$

# Singular Value Decomposition (SVD)

SVD of matrix  $\mathbf{A} \in \mathbb{R}^{p \times d}$  of rank  $r$ :

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$$

- ▶  $\mathbf{U} \in \mathbb{R}^{p \times r}$ : left singular vectors,  $\mathbf{U}^\top \mathbf{U} = \mathbf{I}$
- ▶  $\Sigma \in \mathbb{R}^{r \times r}$ : diagonal matrix of singular values
- ▶  $\mathbf{V}^\top \in \mathbb{R}^{r \times d}$ : right singular vectors,  $\mathbf{V}^\top \mathbf{V} = \mathbf{I}$

# Singular Value Decomposition (SVD)

SVD of matrix  $\mathbf{A} \in \mathbb{R}^{p \times d}$  of rank  $r$ :

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$$

- ▶  $\mathbf{U} \in \mathbb{R}^{p \times r}$ : left singular vectors,  $\mathbf{U}^\top \mathbf{U} = \mathbf{I}$
- ▶  $\Sigma \in \mathbb{R}^{r \times r}$ : diagonal matrix of singular values
- ▶  $\mathbf{V}^\top \in \mathbb{R}^{r \times d}$ : right singular vectors,  $\mathbf{V}^\top \mathbf{V} = \mathbf{I}$

**Applications:** dimensionality reduction, image compression, linear systems.

# Eigenvalue Decomposition from SVD

Given the SVD

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$$

# Eigenvalue Decomposition from SVD

Given the SVD

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$$

The Eigenvalue Decomposition of  $\mathbf{A}^\top \mathbf{A}$  is

$$\mathbf{C} = \mathbf{A}^\top \mathbf{A} = \mathbf{V}\Sigma^\top \Sigma\mathbf{V}^\top = \mathbf{V}\Lambda\mathbf{V}^\top$$

# Eigenvalue Decomposition from SVD

Given the SVD

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$$

The Eigenvalue Decomposition of  $\mathbf{A}^\top \mathbf{A}$  is

$$\mathbf{C} = \mathbf{A}^\top \mathbf{A} = \mathbf{V}\Sigma^\top \Sigma\mathbf{V}^\top = \mathbf{V}\Lambda\mathbf{V}^\top$$

- ▶ **Eigenvectors**: columns of  $\mathbf{V}$ ,  $\mathbf{V}^\top \mathbf{V} = \mathbf{I}$
- ▶ **Eigenvalues**: diagonal of  $\Sigma$

# Eigenvalue Decomposition from SVD

Given the SVD

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$$

The Eigenvalue Decomposition of  $\mathbf{A}^\top \mathbf{A}$  is

$$\mathbf{C} = \mathbf{A}^\top \mathbf{A} = \mathbf{V}\Sigma^\top \Sigma\mathbf{V}^\top = \mathbf{V}\Lambda\mathbf{V}^\top$$

- ▶ **Eigenvectors:** columns of  $\mathbf{V}$ ,  $\mathbf{V}^\top \mathbf{V} = \mathbf{I}$
- ▶ **Eigenvalues:** diagonal of  $\Sigma$

*Exercises:*

- ▶ Show the eigenvalue decomposition of  $\mathbf{A}\mathbf{A}^\top$  is  $\mathbf{U}\Sigma\Sigma^\top\mathbf{U}^\top$
- ▶ Show the trace of  $\mathbf{C}$  is the sum of its eigenvalues

# Eigenvalue Optimization Formulation

# Eigenvalue Optimization Formulation

## 1. Raleigh quotient:

$$\lambda_i = \max_{\mathbf{v}^\top \mathbf{v}_j = 0 \ \forall j < i} \frac{\mathbf{v}^\top \mathbf{C} \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}$$

# Eigenvalue Optimization Formulation

## 1. Raleigh quotient:

$$\lambda_i = \max_{\mathbf{v}^\top \mathbf{v} = 0 \ \forall j < i} \frac{\mathbf{v}^\top \mathbf{C} \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}$$

## 2. Via trace:

$$\mathbf{V} = \arg \max_{\mathbf{W}^\top \mathbf{W} = \mathbf{I}} \text{tr}(\mathbf{W}^\top \mathbf{C} \mathbf{W})$$

# Eigenvalue Optimization Formulation

## 1. Raleigh quotient:

$$\lambda_i = \max_{\mathbf{v}^\top \mathbf{v}_j = 0 \ \forall j < i} \frac{\mathbf{v}^\top \mathbf{C} \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}$$

## 2. Via trace:

$$\mathbf{V} = \arg \max_{\mathbf{W}^\top \mathbf{W} = \mathbf{I}} \text{tr}(\mathbf{W}^\top \mathbf{C} \mathbf{W})$$

## 3. Reconstruction error:

$$\mathbf{V} = \arg \min_{\mathbf{W}^\top \mathbf{W} = \mathbf{I}} \|\mathbf{A} - \mathbf{A} \mathbf{W} \mathbf{W}^\top\|_F^2$$

# Eigenvalue Optimization Formulation

## 1. Raleigh quotient:

$$\lambda_i = \max_{\mathbf{v}^\top \mathbf{v} = 0 \ \forall j < i} \frac{\mathbf{v}^\top \mathbf{C} \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}$$

## 2. Via trace:

$$\mathbf{V} = \arg \max_{\mathbf{W}^\top \mathbf{W} = \mathbf{I}} \text{tr}(\mathbf{W}^\top \mathbf{C} \mathbf{W})$$

## 3. Reconstruction error:

$$\mathbf{V} = \arg \min_{\mathbf{W}^\top \mathbf{W} = \mathbf{I}} \|\mathbf{A} - \mathbf{A} \mathbf{W} \mathbf{W}^\top\|_F^2$$

Exercise: Show that these optimizations are equivalent.

# Generalized Eigenvalue Decomposition

Solve:

$$\mathbf{C}_A \mathbf{w} = \lambda \mathbf{C}_B \mathbf{w}$$

Optimization form:

$$\lambda_i = \max_{\mathbf{w}^\top \mathbf{w}_j = 0 \ \forall j < i} \frac{\mathbf{w}^\top \mathbf{C}_A \mathbf{w}}{\mathbf{w}^\top \mathbf{C}_B \mathbf{w}}$$

Approximate with:

$$\mathbf{C}_B^{-1} \mathbf{C}_A$$

Use SVD for pseudo-inverse:

- ▶  $\mathbf{C}_B = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top$
- ▶  $\mathbf{C}_B^\dagger = \mathbf{V} \boldsymbol{\Sigma}^{-1} \mathbf{U}^\top$

Condition number:

$$\kappa(\mathbf{C}_B) = \frac{\sigma_{\max}}{\sigma_{\min}}$$

Big  $\kappa$  bad, Small  $\kappa$  good

# Lab 2: Matrix Decompositions

*Go to `Lab2_MatrixDecompositions.ipynb`*

# Gradient Descent

## Gradient Descent: Overview

- ▶ Gradient Descent is an optimization algorithm to minimize a function
- ▶ It iteratively moves in the direction of steepest descent (negative gradient)

# Objective

Minimize a loss function  $f(\theta)$ , where:

- ▶  $\theta$  are the model parameters

# Update Rule

# Update Rule

At each iteration, the parameters are updated as:

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

- ▶  $\theta_t$ : Current parameters at iteration  $t$
- ▶  $\eta$ : Learning rate (step size)
- ▶  $\nabla f(\theta_t)$ : Gradient of the loss function

# The Landscape of Dimensionality Reduction

*Inspired by Lee and Verleysen [2007]*

## DR Reduces the Feature Space

# DR Reduces the Feature Space

## **Data**

# DR Reduces the Feature Space

## **Data**

- ▶  $p$  samples

# DR Reduces the Feature Space

## Data

- ▶  $p$  samples
- ▶  $d$  dimensional ambient space

# DR Reduces the Feature Space

## Data

- ▶  $p$  samples
- ▶  $d$  dimensional ambient space
- ▶ Dataset  $\mathcal{D}$  ( $d$  features)

# DR Reduces the Feature Space

## Data

- ▶  $p$  samples
- ▶  $d$  dimensional ambient space
- ▶ Dataset  $\mathcal{D}$  ( $d$  features)
- ▶ Reduced dataset  $\mathcal{R}$  ( $k$  features)

**Dimensionality Reduction (DR)** maps  $d$ -dimensional features to  $k$ -dimensional features ( $k < d$ ).

# Why DR?

# Why DR?

give me 5 fake and funny quotes from researchers about how DR changed their life



Sure! Here are 5 fake and funny quotes from fictional researchers on how dimensionality reduction (DR) changed their lives:

---

1. *"Before PCA, my life was just noise. Now it's all principal components — including my dating life!"*

— **Dr. Ada Scatter**, Romantic Data Scientist

---

2. *"I applied t-SNE to my thoughts and finally found out why I kept forgetting my keys."*

— **Prof. Moe Mertz**, Cognitive Chaos Lab

---

3. *"After discovering manifold learning, I realized my career path wasn't linear either — just locally Euclidean."*

— **Dr. Lin E. Ar**, Nonlinear Trajectory Researcher

---

4. *"With UMAP, I reduced my midlife crisis to just two dimensions: wine and existential dread."*

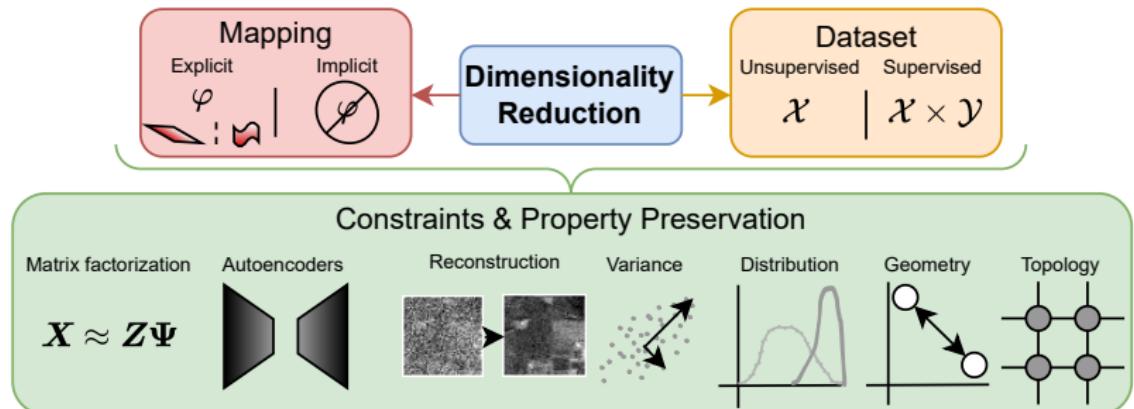
— **Dr. Max E. Stress**, Professor of Reduced Expectations

---

5. *"Autoencoders helped me compress my emotions. Now I only cry in low resolution."*

— **Dr. Dee Pression**, Deep Learning Enthusiast

# DR Summary



# Dataset

## Unsupervised

$$\mathcal{D} = \mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\} \subset \mathbb{R}^d$$

e.g., Visualize data in 2D to see if there are any patterns

# Dataset

## Unsupervised

$$\mathcal{D} = \mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\} \subset \mathbb{R}^d$$

e.g., Visualize data in 2D to see if there are any patterns

## Supervised

$$\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}, \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_p\}$$

e.g., Find a low-dimensional, discriminatory feature space w.r.t.  $\mathcal{Y}$

# DR Mapping

# DR Mapping

**Explicit** DR mapping outputs  $\phi$  where

$$\phi(\mathbf{x}_i) = \mathbf{z}_i$$

- ▶ Approximate inverse  $\psi \approx \phi^{-1}$
- ▶ Reconstructions:  $\mathbf{x}_i \approx \hat{\mathbf{x}}_i = \psi(\phi(\mathbf{x}_i))$

e.g., Want to fit a model on some data, then apply it to “unseen” data.

**Implicit:**  $\phi, \psi$  are not explicitly defined but inferred

e.g., Model fit on all the data to be reduced

# Matrix Factorization

# Matrix Factorization

**Data matrix:**  $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_p]^\top \in \mathbb{R}^{p \times d}$

# Matrix Factorization

**Data matrix:**  $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_p]^\top \in \mathbb{R}^{p \times d}$

**Reduced data matrix:**  $\mathbf{Z} = [\mathbf{z}_1 \cdots \mathbf{z}_p]^\top \in \mathbb{R}^{p \times k}$

# Matrix Factorization

**Data matrix:**  $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_p]^\top \in \mathbb{R}^{p \times d}$

**Reduced data matrix:**  $\mathbf{Z} = [\mathbf{z}_1 \cdots \mathbf{z}_p]^\top \in \mathbb{R}^{p \times k}$

**The model:**

$$\mathbf{X} \approx \mathbf{Z}\Psi$$

# Autoencoders

# Autoencoders

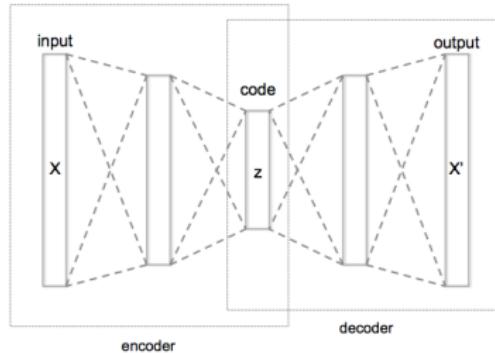


Figure: Image from  
[https://commons.wikimedia.org/wiki/File:Autoencoder\\_structure.png](https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png)

# Autoencoders

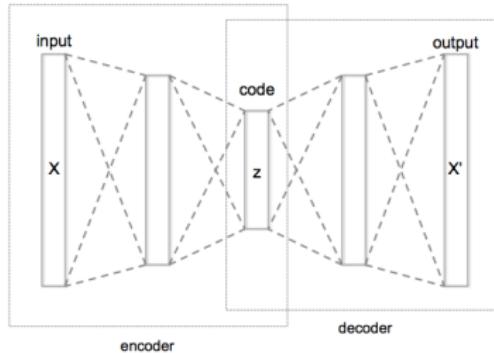


Figure: Image from  
[https://commons.wikimedia.org/wiki/File:Autoencoder\\_structure.png](https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png)

- Neural networks trained to reconstruct input

# Autoencoders

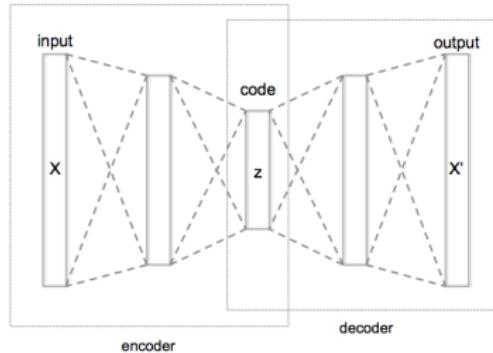


Figure: Image from  
[https://commons.wikimedia.org/wiki/File:Autoencoder\\_structure.png](https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png)

- ▶ Neural networks trained to reconstruct input
- ▶ Bottleneck structure forces compression

# Autoencoders

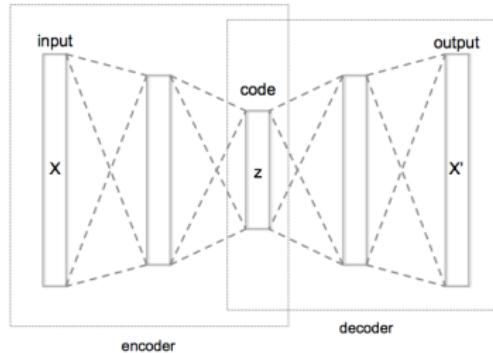


Figure: Image from  
[https://commons.wikimedia.org/wiki/File:Autoencoder\\_structure.png](https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png)

- ▶ Neural networks trained to reconstruct input
- ▶ Bottleneck structure forces compression
- ▶ Encoder:  $\phi(x) = z$ , Decoder:  $\psi(z) = \hat{x}$

# Autoencoders

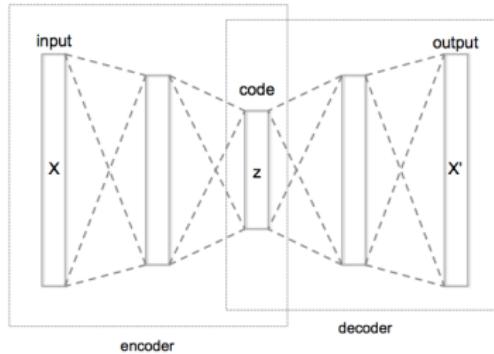


Figure: Image from  
[https://commons.wikimedia.org/wiki/File:Autoencoder\\_structure.png](https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png)

- ▶ Neural networks trained to reconstruct input
- ▶ Bottleneck structure forces compression
- ▶ Encoder:  $\phi(x) = z$ , Decoder:  $\psi(z) = \hat{x}$
- ▶ Loss: e.g., MSE  $\sum_i \|x_i - \psi \circ \phi(x_i)\|^2$

# Autoencoders

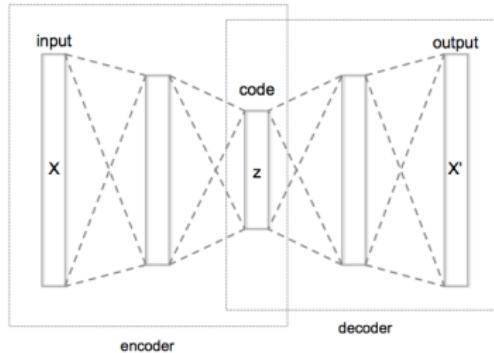


Figure: Image from

[https://commons.wikimedia.org/wiki/File:Autoencoder\\_structure.png](https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png)

- ▶ Neural networks trained to reconstruct input
- ▶ Bottleneck structure forces compression
- ▶ Encoder:  $\phi(\mathbf{x}) = \mathbf{z}$ , Decoder:  $\psi(\mathbf{z}) = \hat{\mathbf{x}}$
- ▶ Loss: e.g.,  $\text{MSE} \sum_i \|\mathbf{x}_i - \psi \circ \phi(\mathbf{x}_i)\|^2$
- ▶ Variants: Denoising, Variational (VAE), Convolutional

# DR as Optimization

$$\arg \min_{\mathcal{Z}} L(\mathcal{X}, \mathcal{Z})$$

# DR as Optimization

$$\arg \min_{\mathcal{Z}} L(\mathcal{X}, \mathcal{Z})$$

This general form preserves

- ▶ Reconstructions
- ▶ Variance
- ▶ Probability distributions
- ▶ Geometry
- ▶ Graph structures

# Reconstruction-preserving

# Reconstruction-preserving

$$\begin{aligned} \min_{\psi, \phi} \quad & \sum_{i=1}^p \|\mathbf{x}_i - \psi \circ \phi(\mathbf{x}_i)\|^2 \\ \text{s.t.} \quad & \phi : \mathbb{R}^d \rightarrow \mathbb{R}^k, \quad \psi : \mathbb{R}^k \rightarrow \mathbb{R}^d \end{aligned} \tag{1}$$

## Reconstruction-preserving

$$\begin{aligned} \min_{\psi, \phi} \quad & \sum_{i=1}^p \|\mathbf{x}_i - \psi \circ \phi(\mathbf{x}_i)\|^2 \\ \text{s.t.} \quad & \phi: \mathbb{R}^d \rightarrow \mathbb{R}^k, \quad \psi: \mathbb{R}^k \rightarrow \mathbb{R}^d \end{aligned} \tag{1}$$

Useful when interpretability or invertibility is important.

## Variance-preserving

Data  $\mathbf{x}$  and DR mapping  $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$

## Variance-preserving

Data  $\mathbf{x}$  and DR mapping  $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$

$$\phi = \arg \max \mathbb{E}[\phi(\mathbf{x})^\top \phi(\mathbf{x})] \quad (2)$$

# Distribution-preserving

## Distribution-preserving

- ▶  $P$  “true” (target) distribution with distribution function  $p$
- ▶  $Q$  “predicted” (modeled) distribution with distribution function  $q$

# Distribution-preserving

- ▶  $P$  “true” (target) distribution with distribution function  $p$
- ▶  $Q$  “predicted” (modeled) distribution with distribution function  $q$

## Entropy

$$H(p) = - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

# Distribution-preserving

- ▶  $P$  “true” (target) distribution with distribution function  $p$
- ▶  $Q$  “predicted” (modeled) distribution with distribution function  $q$

## Entropy

$$H(p) = - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

## Cross Entropy

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x) = - \sum_{x \in \mathcal{X}} p(x) \log p(x) + \text{KL}(p \parallel q)$$

# Distribution-preserving

- ▶  $P$  “true” (target) distribution with distribution function  $p$
- ▶  $Q$  “predicted” (modeled) distribution with distribution function  $q$

## Entropy

$$H(p) = - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

## Cross Entropy

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x) = - \sum_{x \in \mathcal{X}} p(x) \log p(x) + \text{KL}(p \parallel q)$$

## Kullback–Leibler (KL) Divergence

$$\text{KL}(p \parallel q) = H(p, q) - H(p) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$$

# Distribution-preserving

- ▶  $P$  “true” (target) distribution with distribution function  $p$
- ▶  $Q$  “predicted” (modeled) distribution with distribution function  $q$

## Entropy

$$H(p) = - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

## Cross Entropy

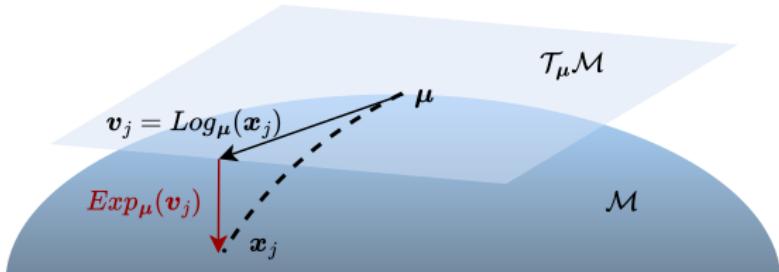
$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x) = - \sum_{x \in \mathcal{X}} p(x) \log p(x) + \text{KL}(p \parallel q)$$

## Kullback–Leibler (KL) Divergence

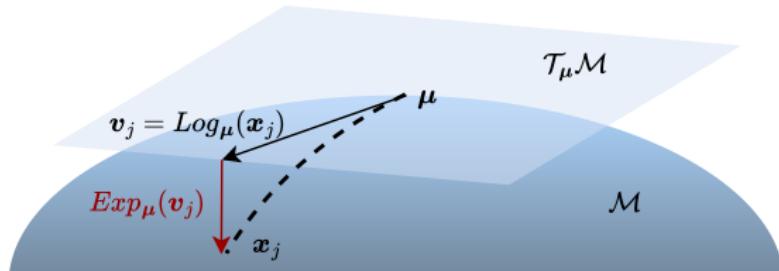
$$\text{KL}(p \parallel q) = H(p, q) - H(p) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$$

1. Determine two distributions to compare
2. Minimize KL divergence directly
3. With conditional distributions, max Evidence Lower BOund (ELBO)

# Geometry-preserving

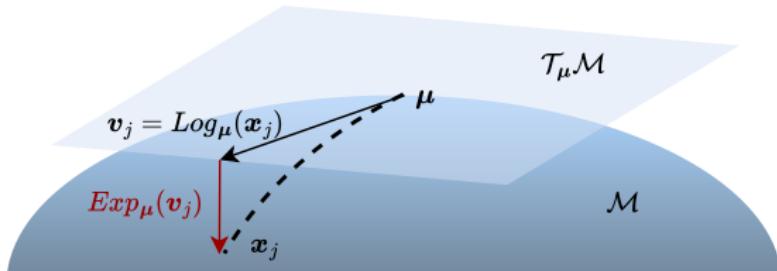


# Geometry-preserving



**Manifold Hypothesis:** High-dimensional data lies (approximately) on a low-dimensional manifold  $\mathcal{M} \subset \mathbb{R}^d$ .

# Geometry-preserving



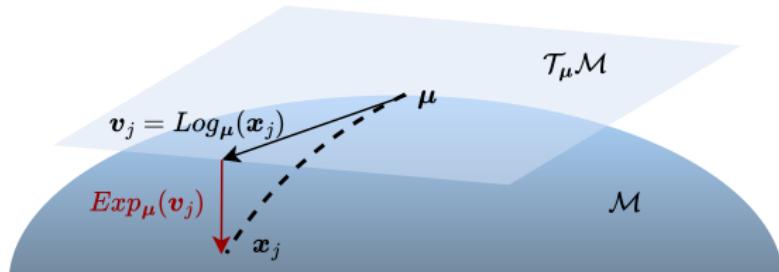
**Manifold Hypothesis:** High-dimensional data lies (approximately) on a low-dimensional manifold  $\mathcal{M} \subset \mathbb{R}^d$ .

**Local Linearity:** Around each  $\mathbf{x}_i \in \mathcal{M}$ , there exists a neighborhood  $U$  such that:

$$\mathcal{M} \cap U \approx T_{\mathbf{x}_i} \mathcal{M}$$

where  $T_{\mathbf{x}_i} \mathcal{M}$  is the tangent space at  $\mathbf{x}_i$ .

# Geometry-preserving



**Manifold Hypothesis:** High-dimensional data lies (approximately) on a low-dimensional manifold  $\mathcal{M} \subset \mathbb{R}^d$ .

**Local Linearity:** Around each  $\mathbf{x}_i \in \mathcal{M}$ , there exists a neighborhood  $U$  such that:

$$\mathcal{M} \cap U \approx T_{\mathbf{x}_i} \mathcal{M}$$

where  $T_{\mathbf{x}_i} \mathcal{M}$  is the tangent space at  $\mathbf{x}_i$ .

**Geodesic Distances:** For  $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M}$ , define

$$d_{\mathcal{M}}(\mathbf{x}_i, \mathbf{x}_j) = \text{length of shortest path on } \mathcal{M}$$

Improves global geometry preservation.

## Topology-preserving

**Goal:** Dimensionality reduction often assumes a topology that determines local neighborhoods and connectivity.

# Topology-preserving

**Goal:** Dimensionality reduction often assumes a topology that determines local neighborhoods and connectivity.

## Data-driven topology (learned):

- ▶ Construct graph  $G = (\mathcal{X}, E)$  from data.
- ▶ Edge weights:  $w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right)$  or  $k$ -nearest neighbors.
- ▶ Preserve graph in low-dimensional representation

# Topology-preserving

**Goal:** Dimensionality reduction often assumes a topology that determines local neighborhoods and connectivity.

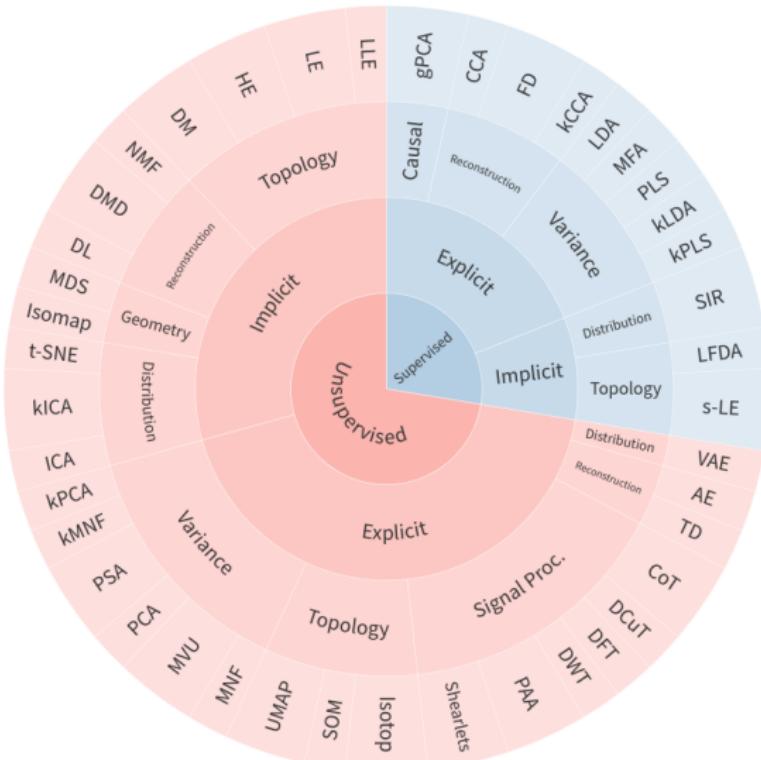
## Data-driven topology (learned):

- ▶ Construct graph  $G = (\mathcal{X}, E)$  from data.
- ▶ Edge weights:  $w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right)$  or  $k$ -nearest neighbors.
- ▶ Preserve graph in low-dimensional representation

## Predefined topology (fixed):

- ▶ Impose grid or lattice structure:  $\mathcal{G}$  = 1D or 2D lattice.
- ▶ Each reduce-space representation  $\mathbf{z}_i$  corresponds to a fixed node in  $\mathcal{G}$ .

## Too Many DR Methods..



# Linear Dimensionality Reduction

# Outline

1. Principal Component Analysis [Hotelling, 1933; Shlens, 2014]
2. Linear Discriminant Analysis [Tharwat et al., 2017]
3. Dynamic Mode Decomposition [Schmid, 2010]

# Principal Component Analysis

# Dataset

We consider a dataset of  $p$  samples with  $d$  features:

$$\{\mathbf{x}_i\}_{i=1}^p \subset \mathbb{R}^d$$

We collect the dataset into a matrix:

$$\mathbf{X} \in \mathbb{R}^{p \times d}$$

**Important:** The data must be **mean-centered**:

$$\frac{1}{p} \sum_{i=1}^p \mathbf{x}_i = \mathbf{0}$$

# Optimization Goal

We aim to extract  $k < p$  features that describe the directions of maximum variance.

We find the  $j$ th direction of maximum variance as

$$\mathbf{v}_j = \arg \max_{\substack{\mathbf{v}^\top \mathbf{v} = 1 \\ \mathbf{v}^\top \mathbf{v}_\ell = 0 \forall \ell < i}} \sum_i (\mathbf{v} \mathbf{x}_i)^\top (\mathbf{v} \mathbf{x}_i) \quad (3)$$

$$= \arg \max_{\substack{\mathbf{v}^\top \mathbf{v} = 1 \\ \mathbf{v}^\top \mathbf{v}_\ell = 0 \forall \ell < i}} \sum_i \|\mathbf{x}_i\|_2^2 \cos \theta(\mathbf{v}, \mathbf{x}_i) \quad (4)$$

$$= \arg \max_{\substack{\mathbf{v}^\top \mathbf{v} = 1 \\ \mathbf{v}^\top \mathbf{v}_\ell = 0 \forall \ell < i}} \mathbf{v}^\top \mathbf{X}^\top \mathbf{X} \mathbf{v} \quad (5)$$

## Alternative Formulation: Reconstruction Error

This is equivalent to finding the rank- $k$  projection:

$$\Pi_{\mathbf{W}}(\mathbf{x}) := \mathbf{W}\mathbf{W}^\top \mathbf{x}$$

We solve:

$$\mathbf{V} = \arg \min_{\mathbf{W}^\top \mathbf{W} = \mathbf{I}} \mathbb{E} [\|\mathbf{x}_i - \mathbf{W}\mathbf{W}^\top \mathbf{x}_i\|_2^2]$$

*Exercise* Show that we can find the EOFs (columns of  $\mathbf{V}$ ) via eigendecomposition of  $\mathbf{X}^\top \mathbf{X}$  or the SVD of  $\mathbf{X}$ .

# PCA Objective Summary

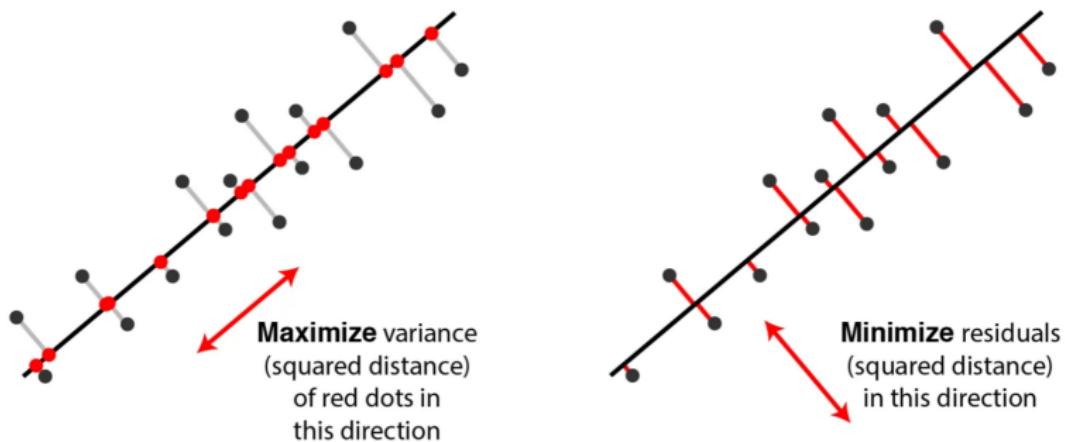


Figure: Image borrowed from <https://medium.com/@fraidoonmarzai99/principal-component-analysis-pca-in-depth-93c871f25dfa>.

## Transformation

We call  $\mathbf{V}$  the matrix of **PCA weights** or **Empirical Orthogonal Functions (EOFs)**.

## Transformation

We call **V** the matrix of **PCA weights** or **Empirical Orthogonal Functions (EOFs)**.

1st EOF is direction of maximum variance, 2nd EOF in direction of maximum variance that is orthogonal to 1st EOF, and so on ...

## Transformation

We call  $\mathbf{V}$  the matrix of **PCA weights** or **Empirical Orthogonal Functions (EOFs)**.

1st EOF is direction of maximum variance, 2nd EOF in direction of maximum variance that is orthogonal to 1st EOF, and so on ...

The first  $k$  principal components of  $\mathbf{X}$ :

# Transformation

We call  $\mathbf{V}$  the matrix of **PCA weights** or **Empirical Orthogonal Functions (EOFs)**.

1st EOF is direction of maximum variance, 2nd EOF in direction of maximum variance that is orthogonal to 1st EOF, and so on ...

The first  $k$  principal components of  $\mathbf{X}$ :

$$\mathbf{Z} = \mathbf{X}\mathbf{V} \in \mathbb{R}^{p \times k}$$

# Transformation

We call  $\mathbf{V}$  the matrix of **PCA weights** or **Empirical Orthogonal Functions (EOFs)**.

1st EOF is direction of maximum variance, 2nd EOF in direction of maximum variance that is orthogonal to 1st EOF, and so on ...

The first  $k$  principal components of  $\mathbf{X}$ :

$$\mathbf{Z} = \mathbf{X}\mathbf{V} \in \mathbb{R}^{p \times k}$$

PCA map of an individual sample:

# Transformation

We call  $\mathbf{V}$  the matrix of **PCA weights** or **Empirical Orthogonal Functions (EOFs)**.

1st EOF is direction of maximum variance, 2nd EOF in direction of maximum variance that is orthogonal to 1st EOF, and so on ...

The first  $k$  principal components of  $\mathbf{X}$ :

$$\mathbf{Z} = \mathbf{X}\mathbf{V} \in \mathbb{R}^{p \times k}$$

PCA map of an individual sample:

$$\mathbf{z} = \phi(\mathbf{x}) = \mathbf{V}^\top \mathbf{x} \in \mathbb{R}^k$$

# Transformation

We call  $\mathbf{V}$  the matrix of **PCA weights** or **Empirical Orthogonal Functions (EOFs)**.

1st EOF is direction of maximum variance, 2nd EOF in direction of maximum variance that is orthogonal to 1st EOF, and so on ...

The first  $k$  principal components of  $\mathbf{X}$ :

$$\mathbf{Z} = \mathbf{X}\mathbf{V} \in \mathbb{R}^{p \times k}$$

PCA map of an individual sample:

$$\mathbf{z} = \phi(\mathbf{x}) = \mathbf{V}^\top \mathbf{x} \in \mathbb{R}^k$$

PCA reconstruction of an individual sample:

# Transformation

We call  $\mathbf{V}$  the matrix of **PCA weights** or **Empirical Orthogonal Functions (EOFs)**.

1st EOF is direction of maximum variance, 2nd EOF in direction of maximum variance that is orthogonal to 1st EOF, and so on ...

The first  $k$  principal components of  $\mathbf{X}$ :

$$\mathbf{Z} = \mathbf{X}\mathbf{V} \in \mathbb{R}^{p \times k}$$

PCA map of an individual sample:

$$\mathbf{z} = \phi(\mathbf{x}) = \mathbf{V}^\top \mathbf{x} \in \mathbb{R}^k$$

PCA reconstruction of an individual sample:

$$\hat{\mathbf{x}} = \psi \circ \phi(\mathbf{x}) = \mathbf{V}\mathbf{V}^\top \mathbf{x} = \Pi_{\mathbf{V}}(\mathbf{x}) \in \mathbb{R}^d$$

## Explained Variance

The **explained variance** is how much variance each principal component captures:

## Explained Variance

The **explained variance** is how much variance each principal component captures:

Variance of component  $i$ :

# Explained Variance

The **explained variance** is how much variance each principal component captures:

Variance of component  $i$ :

$$\lambda_i = \mathbf{v}_i^\top \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$$

# Explained Variance

The **explained variance** is how much variance each principal component captures:

Variance of component  $i$ :

$$\lambda_i = \mathbf{v}_i^\top \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$$

Explained variance ratio of component  $i$ :

# Explained Variance

The **explained variance** is how much variance each principal component captures:

Variance of component  $i$ :

$$\lambda_i = \mathbf{v}_i^\top \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$$

Explained variance ratio of component  $n$ :

$$\frac{\lambda_n}{\sum_{j=1}^d \lambda_j} = \frac{\lambda_n}{\text{tr}(\mathbf{X}^\top \mathbf{X})}$$

# Explained Variance

The **explained variance** is how much variance each principal component captures:

Variance of component  $i$ :

$$\lambda_i = \mathbf{v}_i^\top \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$$

Explained variance ratio of component  $n$ :

$$\frac{\lambda_n}{\sum_{j=1}^d \lambda_j} = \frac{\lambda_n}{\text{tr}(\mathbf{X}^\top \mathbf{X})}$$

This helps select the number of components to keep...

# Explained Variance

The **explained variance** is how much variance each principal component captures:

Variance of component  $i$ :

$$\lambda_i = \mathbf{v}_i^\top \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$$

Explained variance ratio of component  $n$ :

$$\frac{\lambda_n}{\sum_{j=1}^d \lambda_j} = \frac{\lambda_n}{\text{tr}(\mathbf{X}^\top \mathbf{X})}$$

This helps select the number of components to keep...

*Rule of thumb is select components that explain > 90% of variance.*

# Explained Variance

The **explained variance** is how much variance each principal component captures:

Variance of component  $i$ :

$$\lambda_i = \mathbf{v}_i^\top \mathbf{X}^\top \mathbf{X} \mathbf{v}_i$$

Explained variance ratio of component  $n$ :

$$\frac{\lambda_n}{\sum_{j=1}^d \lambda_j} = \frac{\lambda_n}{\text{tr}(\mathbf{X}^\top \mathbf{X})}$$

This helps select the number of components to keep...

*Rule of thumb is select components that explain  $> 90\%$  of variance. (but there's other methods [Gavish and Donoho, 2014])*

# When to Use PCA

# When to Use PCA

- ▶ Have unlabeled data

## When to Use PCA

- ▶ Have unlabeled data
- ▶ Want to reduce dimensionality while preserving variance

# When to Use PCA

- ▶ Have unlabeled data
- ▶ Want to reduce dimensionality while preserving variance
- ▶ Want a fast, interpretable linear projection

# When to Use PCA

- ▶ Have unlabeled data
- ▶ Want to reduce dimensionality while preserving variance
- ▶ Want a fast, interpretable linear projection
- ▶ Want to decorrelate features

# When to Use PCA

- ▶ Have unlabeled data
- ▶ Want to reduce dimensionality while preserving variance
- ▶ Want a fast, interpretable linear projection
- ▶ Want to decorrelate features
- ▶ Few outliers

# When to Use PCA

- ▶ Have unlabeled data
- ▶ Want to reduce dimensionality while preserving variance
- ▶ Want a fast, interpretable linear projection
- ▶ Want to decorrelate features
- ▶ Few outliers

**Beyond PCA...** robust subspace recovery [Lerman and Maunu, 2018] & flag manifolds [Mankovich et al., 2024; Szwagier and Pennec, 2024, 2025]

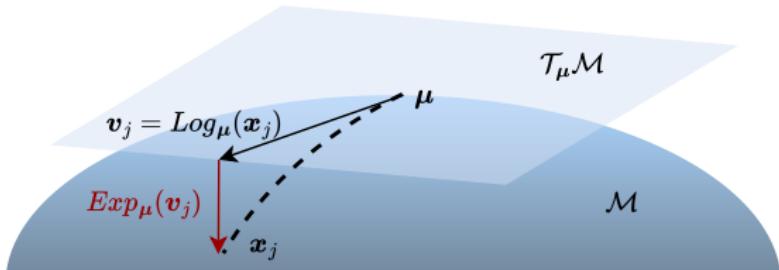
## Aside- PCA on Manifolds (Tangent-PCA)

## Aside- PCA on Manifolds (Tangent-PCA)

Assume data samples from known, Riemannian manifold...

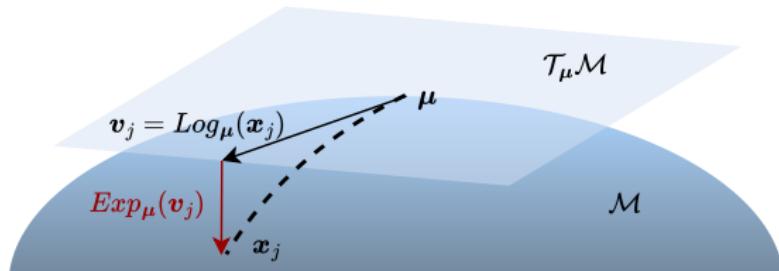
# Aside- PCA on Manifolds (Tangent-PCA)

Assume data samples from known, Riemannian manifold...



## Aside- PCA on Manifolds (Tangent-PCA)

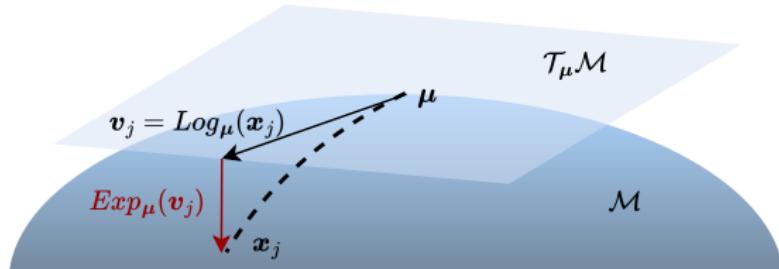
Assume data samples from known, Riemannian manifold...



- ▶ Map data to tangent space

## Aside- PCA on Manifolds (Tangent-PCA)

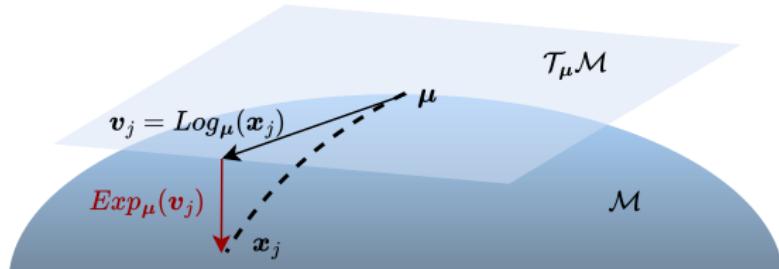
Assume data samples from known, Riemannian manifold...



- ▶ Map data to tangent space
- ▶ Run PCA in tangent space

# Aside- PCA on Manifolds (Tangent-PCA)

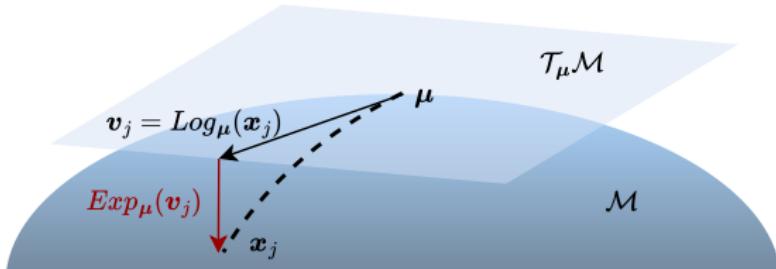
Assume data samples from known, Riemannian manifold...



- ▶ Map data to tangent space
- ▶ Run PCA in tangent space
- ▶ Map back to manifold

# Aside- PCA on Manifolds (Tangent-PCA)

Assume data samples from known, Riemannian manifold...



- ▶ Map data to tangent space
- ▶ Run PCA in tangent space
- ▶ Map back to manifold

Read more here: [Fletcher et al., 2004; Pennec, 2018]

# Lab 3: PCA

*Go to Lab3\_PCA.ipynb*

# Linear Discriminant Analysis

# What is LDA?

# What is LDA?

**Linear Discriminant Analysis (LDA)** is a supervised dimensionality reduction technique.

# What is LDA?

**Linear Discriminant Analysis (LDA)** is a supervised dimensionality reduction technique.

**Goal:** Project high-dimensional data onto a lower-dimensional space that best separates multiple classes.

# What is LDA?

**Linear Discriminant Analysis (LDA)** is a supervised dimensionality reduction technique.

**Goal:** Project high-dimensional data onto a lower-dimensional space that best separates multiple classes.

- ▶ Maximizes **between-class variance**
- ▶ Minimizes **within-class variance**

# What is LDA?

**Linear Discriminant Analysis (LDA)** is a supervised dimensionality reduction technique.

**Goal:** Project high-dimensional data onto a lower-dimensional space that best separates multiple classes.

- ▶ Maximizes **between-class variance**
- ▶ Minimizes **within-class variance**

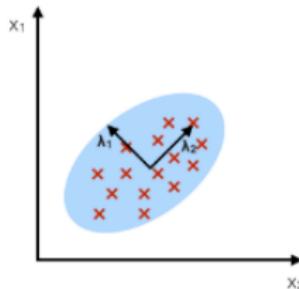
*"PCA with class information!"*

# LDA Intuition

# LDA Intuition

## PCA:

component axes that maximize the variance



## LDA:

maximizing the component axes for class-separation

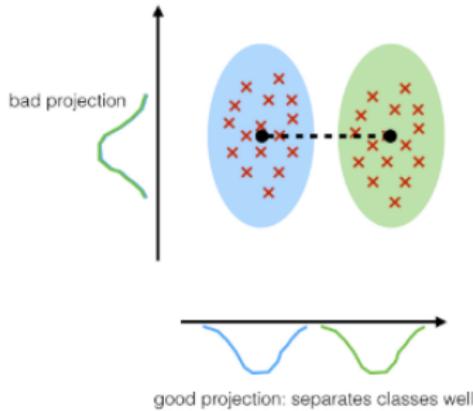


Figure: Image borrowed from

[https://sebastianraschka.com/Articles/2014\\_python\\_lda.html](https://sebastianraschka.com/Articles/2014_python_lda.html)

# Mathematical Formulation

## Mathematical Formulation

Given labeled dataset:

# Mathematical Formulation

Given labeled dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^p, \quad \mathbf{x}_i \in \mathbb{R}^d$$

Number of classes:  $C$ .

# Mathematical Formulation

Given labeled dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^p, \quad \mathbf{x}_i \in \mathbb{R}^d$$

Number of classes:  $C$ .

**Within-class scatter matrix:**

$$\mathbf{S}_W = \sum_{c=1}^C \sum_{\mathbf{x}_i \in c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^\top$$

# Mathematical Formulation

Given labeled dataset:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^p, \quad \mathbf{x}_i \in \mathbb{R}^d$$

Number of classes:  $C$ .

**Within-class scatter matrix:**

$$\mathbf{S}_W = \sum_{c=1}^C \sum_{\mathbf{x}_i \in c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^\top$$

**Between-class scatter matrix:**

$$\mathbf{S}_B = \sum_{c=1}^C (\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^\top$$

# Optimization Problem

# Optimization Problem

We solve the following generalized eigenvalue problem:

# Optimization Problem

We solve the following generalized eigenvalue problem:

$$\mathbf{W} = \arg \max_{\mathbf{W}} \frac{\text{tr}(\mathbf{W}^\top S_B \mathbf{W})}{\text{tr}(\mathbf{W}^\top S_W \mathbf{W})}$$

# Optimization Problem

We solve the following generalized eigenvalue problem:

$$\mathbf{W} = \arg \max_{\mathbf{W}} \frac{\text{tr}(\mathbf{W}^\top S_B \mathbf{W})}{\text{tr}(\mathbf{W}^\top S_W \mathbf{W})}$$

**Output:**

# Optimization Problem

We solve the following generalized eigenvalue problem:

$$\mathbf{W} = \arg \max_{\mathbf{W}} \frac{\text{tr}(\mathbf{W}^\top S_B \mathbf{W})}{\text{tr}(\mathbf{W}^\top S_W \mathbf{W})}$$

## Output:

- ▶  $\mathbf{W} \in \mathbb{R}^{d \times k}$ : DR mapping matrix

# Optimization Problem

We solve the following generalized eigenvalue problem:

$$\mathbf{W} = \arg \max_{\mathbf{W}} \frac{\text{tr}(\mathbf{W}^\top S_B \mathbf{W})}{\text{tr}(\mathbf{W}^\top S_W \mathbf{W})}$$

## Output:

- ▶  $\mathbf{W} \in \mathbb{R}^{d \times k}$ : DR mapping matrix
- ▶  $k \leq C - 1$ : max number of discriminative components

# Optimization Problem

We solve the following generalized eigenvalue problem:

$$\mathbf{W} = \arg \max_{\mathbf{W}} \frac{\text{tr}(\mathbf{W}^\top S_B \mathbf{W})}{\text{tr}(\mathbf{W}^\top S_W \mathbf{W})}$$

## Output:

- ▶  $\mathbf{W} \in \mathbb{R}^{d \times k}$ : DR mapping matrix
- ▶  $k \leq C - 1$ : max number of discriminative components

## Reduced data:

$$\mathbf{z}_i = \mathbf{W}^\top \mathbf{x}_i$$

# When to Use LDA

# When to Use LDA

- ▶ Labeled data

## When to Use LDA

- ▶ Labeled data
- ▶ Want low-dimensional features that separate classes well

# When to Use LDA

- ▶ Labeled data
- ▶ Want low-dimensional features that separate classes well
- ▶ Want a fast, interpretable linear projection

## When to Use LDA

- ▶ Labeled data
- ▶ Want low-dimensional features that separate classes well
- ▶ Want a fast, interpretable linear projection
- ▶ Linearly separable classes

# When to Use LDA

- ▶ Labeled data
- ▶ Want low-dimensional features that separate classes well
- ▶ Want a fast, interpretable linear projection
- ▶ Linearly separable classes
- ▶ Each class follows a multivariate normal distribution with equal covariances

# LDA vs PCA

# LDA vs PCA

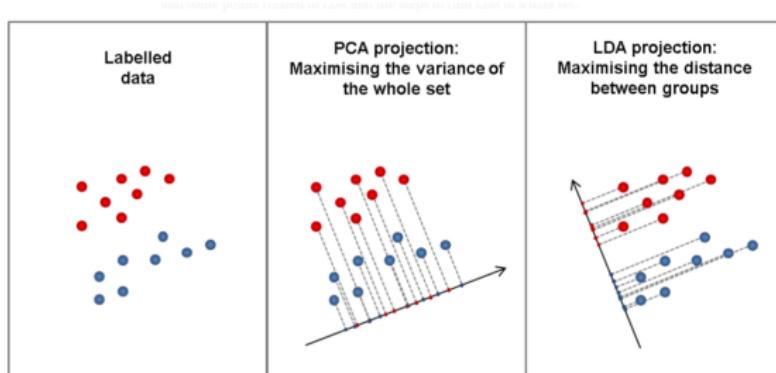


Figure: Image borrowed from <https://vivekmuraleedharan73.medium.com/what-is-linear-discriminant-analysis-lda-7e33ff59020a>.

Aspect	PCA [Shlens, 2014]	LDA [Tharwat et al., 2017]
Supervised?	No	Yes
Objective	Maximize variance	Maximize class separation
Axes chosen	max variance	best separation
Max dimensions	$\leq$ input dimension	$\leq$ number of classes - 1

# Lab 3: LDA

*Go to Lab4\_LDA.ipynb*

# Dynamic Mode Decomposition

# Dynamical Systems

# Dynamical Systems

A **dynamical system** describes how a state evolves over time according to a fixed rule.

# Dynamical Systems

A **dynamical system** describes how a state evolves over time according to a fixed rule.

Discrete-time:  $\mathbf{x}(t + \tau) = f(\mathbf{x}(t))$

Continuous-time:  $\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t))$

# Dynamical Systems

A **dynamical system** describes how a state evolves over time according to a fixed rule.

Discrete-time:  $\mathbf{x}(t + \tau) = f(\mathbf{x}(t))$

Continuous-time:  $\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t))$

- ▶  $\mathbf{x}(t) \in \mathbb{R}^n$ : the state at time  $t$

# Dynamical Systems

A **dynamical system** describes how a state evolves over time according to a fixed rule.

Discrete-time:  $\mathbf{x}(t + \tau) = f(\mathbf{x}(t))$

Continuous-time:  $\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t))$

- ▶  $\mathbf{x}(t) \in \mathbb{R}^n$ : the state at time  $t$
- ▶  $f, g$ : a (possibly nonlinear) update rule

# Dynamical Systems

A **dynamical system** describes how a state evolves over time according to a fixed rule.

Discrete-time:  $\mathbf{x}(t + \tau) = f(\mathbf{x}(t))$

Continuous-time:  $\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t))$

- ▶  $\mathbf{x}(t) \in \mathbb{R}^n$ : the state at time  $t$
- ▶  $f, g$ : a (possibly nonlinear) update rule
- ▶ Goal: Understand, predict, or control the evolution of  $\mathbf{x}(t)$

# Dynamical Systems

A **dynamical system** describes how a state evolves over time according to a fixed rule.

Discrete-time:  $\mathbf{x}(t + \tau) = f(\mathbf{x}(t))$       Continuous-time:  $\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t))$

- ▶  $\mathbf{x}(t) \in \mathbb{R}^n$ : the state at time  $t$
- ▶  $f, g$ : a (possibly nonlinear) update rule
- ▶ Goal: Understand, predict, or control the evolution of  $\mathbf{x}(t)$

## Linear Systems

If  $f(\mathbf{x}) = A\mathbf{x}$ , the system is linear:

$$\mathbf{x}(t + \tau) = A\mathbf{x}(t)$$

# Dynamical Systems

A **dynamical system** describes how a state evolves over time according to a fixed rule.

Discrete-time:  $\mathbf{x}(t + \tau) = f(\mathbf{x}(t))$       Continuous-time:  $\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t))$

- ▶  $\mathbf{x}(t) \in \mathbb{R}^n$ : the state at time  $t$
- ▶  $f, g$ : a (possibly nonlinear) update rule
- ▶ Goal: Understand, predict, or control the evolution of  $\mathbf{x}(t)$

## Linear Systems

If  $f(\mathbf{x}) = A\mathbf{x}$ , the system is linear:

$$\mathbf{x}(t + \tau) = A\mathbf{x}(t)$$

⇒ Solutions evolve through powers of  $A$ : eigenvalues/eigenvectors govern behavior.

# Dynamic Mode Decomposition

# Dynamic Mode Decomposition

Assume that the data is sampled from the timeseries

$$\mathbf{x}(t + \tau) \approx \mathbf{A} \mathbf{x}(t).$$

# Dynamic Mode Decomposition

Assume that the data is sampled from the timeseries

$$\mathbf{x}(t + \tau) \approx \mathbf{A}\mathbf{x}(t).$$

**Decompose the system** into spatial and temporal patterns.

# Dynamic Mode Decomposition

Assume that the data is sampled from the timeseries

$$\mathbf{x}(t + \tau) \approx \mathbf{A}\mathbf{x}(t).$$

**Decompose the system** into spatial and temporal patterns.

Analyzing  $\mathbf{A}$  results in the DMD [Schmid, 2010]

$$\mathbf{x}(t) = \sum_{j=1}^k \phi_j e^{\omega_j t} b_j.$$

# Dynamic Mode Decomposition

Assume that the data is sampled from the timeseries

$$\mathbf{x}(t + \tau) \approx \mathbf{A}\mathbf{x}(t).$$

**Decompose the system** into spatial and temporal patterns.

Analyzing  $\mathbf{A}$  results in the DMD [Schmid, 2010]

$$\mathbf{x}(t) = \sum_{j=1}^k \phi_j e^{\omega_j t} b_j.$$

- ▶  $\phi_j \in \mathbb{C}^n$  **feature patterns** (dynamic modes)
- ▶  $\omega_j \in \mathbb{C}$  **temporal characteristics** (continuous time eigenvalues)
- ▶  $b_j \in \mathbb{R}$  scalar loadings (a.k.a. amplitudes)

# Eigenvalue Interpretation

# Eigenvalue Interpretation

$$\lambda_j = e^{\omega_j \tau}$$

- ▶ Discrete time  $\lambda_j$
- ▶ Continuous time  $\omega_j$

# Eigenvalue Interpretation

$$\lambda_j = e^{\omega_j \tau}$$

- ▶ Discrete time  $\lambda_j$
- ▶ Continuous time  $\omega_j$

Write in polar form

$$\lambda_j = r e^{i\theta}$$

# Eigenvalue Interpretation

$$\lambda_j = e^{\omega_j \tau}$$

- ▶ Discrete time  $\lambda_j$
- ▶ Continuous time  $\omega_j$

Write in polar form

$$\lambda_j = r e^{i\theta}$$

- ▶ Trend:  $r = |\lambda_j|$

# Eigenvalue Interpretation

$$\lambda_j = e^{\omega_j \tau}$$

- Discrete time  $\lambda_j$
- Continuous time  $\omega_j$

Write in polar form

$$\lambda_j = r e^{i\theta}$$

- Trend:  $r = |\lambda_j|$
- Oscillation frequency (per  $\tau$ ):  $\theta = -i \log(\lambda_j/r)$

# Eigenvalue Interpretation

$$\lambda_j = e^{\omega_j \tau}$$

- Discrete time  $\lambda_j$
- Continuous time  $\omega_j$

Write in polar form

$$\lambda_j = r e^{i\theta}$$

- Trend:  $r = |\lambda_j|$
- Oscillation frequency (per  $\tau$ ):  $\theta = -i \log(\lambda_j/r)$

# Exact Dynamic Mode Decomposition [Dawson et al., 2016]

# Exact Dynamic Mode Decomposition [Dawson et al., 2016]

1. Stack the data

$$\mathbf{X} = [\mathbf{x}(1) | \cdots | \mathbf{x}(T - \tau)] \in \mathbb{R}^{n \times p}, \quad \mathbf{X}' = [\mathbf{x}(1 + \tau) | \cdots | \mathbf{x}(T)] \in \mathbb{R}^{n \times p},$$

# Exact Dynamic Mode Decomposition [Dawson et al., 2016]

1. Stack the data

$$\mathbf{X} = [\mathbf{x}(1) | \cdots | \mathbf{x}(T - \tau)] \in \mathbb{R}^{n \times p}, \quad \mathbf{X}' = [\mathbf{x}(1 + \tau) | \cdots | \mathbf{x}(T)] \in \mathbb{R}^{n \times p},$$

2. Want to solve

$$\min_{\mathbf{A}} \|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F$$

# Exact Dynamic Mode Decomposition [Dawson et al., 2016]

1. Stack the data

$$\mathbf{X} = [\mathbf{x}(1) | \cdots | \mathbf{x}(T-\tau)] \in \mathbb{R}^{n \times p}, \quad \mathbf{X}' = [\mathbf{x}(1+\tau) | \cdots | \mathbf{x}(T)] \in \mathbb{R}^{n \times p},$$

2. Want to solve

$$\min_{\mathbf{A}} \|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F$$

3. Rank- $r$  truncated SVD  $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$  leads to

$$\tilde{\mathbf{A}} = \mathbf{U}^\top \mathbf{X}' \mathbf{V} \Sigma^{-1} \in \mathbb{R}^{k \times k}, \quad \tilde{\mathbf{A}} \mathbf{W} = \mathbf{W} \Lambda, \quad (\text{eigendecomposition})$$

# Exact Dynamic Mode Decomposition [Dawson et al., 2016]

1. Stack the data

$$\mathbf{X} = [\mathbf{x}(1) | \cdots | \mathbf{x}(T-\tau)] \in \mathbb{R}^{n \times p}, \quad \mathbf{X}' = [\mathbf{x}(1+\tau) | \cdots | \mathbf{x}(T)] \in \mathbb{R}^{n \times p},$$

2. Want to solve

$$\min_{\mathbf{A}} \|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F$$

3. Rank- $r$  truncated SVD  $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$  leads to

$$\tilde{\mathbf{A}} = \mathbf{U}^\top \mathbf{X}' \mathbf{V} \Sigma^{-1} \in \mathbb{R}^{k \times k}, \quad \tilde{\mathbf{A}} \mathbf{W} = \mathbf{W} \Lambda, \quad (\text{eigendecomposition})$$

4. Dynamic modes

$$\phi_j = \frac{1}{\lambda_j} \mathbf{X}' \mathbf{V} \Sigma^{-1} \mathbf{w}_j$$

# Exact Dynamic Mode Decomposition [Dawson et al., 2016]

1. Stack the data

$$\mathbf{X} = [\mathbf{x}(1) | \cdots | \mathbf{x}(T-\tau)] \in \mathbb{R}^{n \times p}, \quad \mathbf{X}' = [\mathbf{x}(1+\tau) | \cdots | \mathbf{x}(T)] \in \mathbb{R}^{n \times p},$$

2. Want to solve

$$\min_{\mathbf{A}} \|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F$$

3. Rank- $r$  truncated SVD  $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$  leads to

$$\tilde{\mathbf{A}} = \mathbf{U}^\top \mathbf{X}' \mathbf{V}\Sigma^{-1} \in \mathbb{R}^{k \times k}, \quad \tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\Lambda, \quad (\text{eigendecomposition})$$

4. Dynamic modes

$$\phi_j = \frac{1}{\lambda_j} \mathbf{X}' \mathbf{V}\Sigma^{-1} \mathbf{w}_j$$

5. Eigenvalues (discrete time)

$$\lambda_j = e^{\omega_j \tau}$$

# Exact Dynamic Mode Decomposition [Dawson et al., 2016]

1. Stack the data

$$\mathbf{X} = [\mathbf{x}(1) | \cdots | \mathbf{x}(T-\tau)] \in \mathbb{R}^{n \times p}, \quad \mathbf{X}' = [\mathbf{x}(1+\tau) | \cdots | \mathbf{x}(T)] \in \mathbb{R}^{n \times p},$$

2. Want to solve

$$\min_{\mathbf{A}} \|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F$$

3. Rank- $r$  truncated SVD  $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$  leads to

$$\tilde{\mathbf{A}} = \mathbf{U}^\top \mathbf{X}' \mathbf{V}\Sigma^{-1} \in \mathbb{R}^{k \times k}, \quad \tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\Lambda, \quad (\text{eigendecomposition})$$

4. Dynamic modes

$$\phi_j = \frac{1}{\lambda_j} \mathbf{X}' \mathbf{V}\Sigma^{-1} \mathbf{w}_j$$

5. Eigenvalues (discrete time)

$$\lambda_j = e^{\omega_j \tau}$$

6. Loadings found by solving

$$\Phi \mathbf{b} = \mathbf{x}(1).$$

# My Interpretation

## My Interpretation

The system evolves in the low-dimensional reduced space mapped to by  
 $\mathbf{U}^T \in \mathbb{R}^{k \times n}$

## My Interpretation

The system evolves in the low-dimensional reduced space mapped to by  $\mathbf{U}^T \in \mathbb{R}^{k \times n}$

$$\mathbf{U}^T \mathbf{x}(t + \tau) = \mathbf{z}(t + \tau) = \tilde{\mathbf{A}}\mathbf{z}(t) = \tilde{\mathbf{A}}\mathbf{U}^T \mathbf{x}(t).$$

## My Interpretation

The system evolves in the low-dimensional reduced space mapped to by  $\mathbf{U}^T \in \mathbb{R}^{k \times n}$

$$\mathbf{U}^T \mathbf{x}(t + \tau) = \mathbf{z}(t + \tau) = \tilde{\mathbf{A}}\mathbf{z}(t) = \tilde{\mathbf{A}}\mathbf{U}^T \mathbf{x}(t).$$

Projected dynamic modes:

$$\phi_j = \mathbf{U}\mathbf{w}_j$$

## My Interpretation

The system evolves in the low-dimensional reduced space mapped to by  $\mathbf{U}^T \in \mathbb{R}^{k \times n}$

$$\mathbf{U}^T \mathbf{x}(t + \tau) = \mathbf{z}(t + \tau) = \tilde{\mathbf{A}} \mathbf{z}(t) = \tilde{\mathbf{A}} \mathbf{U}^T \mathbf{x}(t).$$

Projected dynamic modes:

$$\phi_j = \mathbf{U} \mathbf{w}_j$$

DMD analyzes the stability of this system and generates spatial patterns in  $\mathbb{R}^n$  (ambient space)

# Using Dynamic Mode Decomposition

# Using Dynamic Mode Decomposition

- ▶ Spatial & temporal patterns of timeseries

# Using Dynamic Mode Decomposition

- ▶ Spatial & temporal patterns of timeseries
- ▶ Ill-conditioning of **X** and **A**

# Using Dynamic Mode Decomposition

- ▶ Spatial & temporal patterns of timeseries
- ▶ Ill-conditioning of  $\mathbf{X}$  and  $\mathbf{A}$
- ▶ Not robust to noise

# Optimized Dynamic Mode Decomposition [Askham and Kutz, 2018]

Stack all the data together

# Optimized Dynamic Mode Decomposition [Askham and Kutz, 2018]

Stack all the data together

$$\mathbf{X} = [\mathbf{x}(t_1) \mid \mathbf{x}(t_2) \cdots \mid \mathbf{x}(t_p)] \in \mathbb{R}^{n \times p}.$$

# Optimized Dynamic Mode Decomposition [Askham and Kutz, 2018]

Stack all the data together

$$\mathbf{X} = [\mathbf{x}(t_1) \mid \mathbf{x}(t_2) \cdots \mid \mathbf{x}(t_p)] \in \mathbb{R}^{n \times p}.$$

Optimize directly for DMD matrix decomposition

# Optimized Dynamic Mode Decomposition [Askham and Kutz, 2018]

Stack all the data together

$$\mathbf{X} = [\mathbf{x}(t_1) | \mathbf{x}(t_2) \cdots | \mathbf{x}(t_p)] \in \mathbb{R}^{n \times p}.$$

Optimize directly for DMD matrix decomposition

$$\min_{\mathbf{b}_j, \omega_j, b_j} \|\mathbf{X} - \sum_{j=1}^r \mathbf{b}_j e^{\omega_j t} b_j\|_F.$$

# Optimized Dynamic Mode Decomposition [Askham and Kutz, 2018]

Stack all the data together

$$\mathbf{X} = [\mathbf{x}(t_1) | \mathbf{x}(t_2) \cdots | \mathbf{x}(t_p)] \in \mathbb{R}^{n \times p}.$$

Optimize directly for DMD matrix decomposition

$$\min_{\mathbf{b}_j, \omega_j, b_j} \|\mathbf{X} - \sum_{j=1}^r \mathbf{b}_j e^{\omega_j t} b_j\|_F.$$

Translate to

$$\min_{\mathbf{A}} \|\mathbf{X}^\top - \Phi(\alpha) \mathbf{B}\|_F.$$

# Optimized Dynamic Mode Decomposition [Askham and Kutz, 2018]

Stack all the data together

$$\mathbf{X} = [\mathbf{x}(t_1) | \mathbf{x}(t_2) \cdots | \mathbf{x}(t_p)] \in \mathbb{R}^{n \times p}.$$

Optimize directly for DMD matrix decomposition

$$\min_{\mathbf{b}_j, \omega_j, b_j} \|\mathbf{X} - \sum_{j=1}^r \mathbf{b}_j e^{\omega_j t} b_j\|_F.$$

Translate to

$$\min_{\mathbf{A}} \|\mathbf{X}^\top - \Phi(\alpha) \mathbf{B}\|_F.$$

- $\Phi(\alpha)_{i,j} = \exp(\alpha_j t_i)$
- $b_j = \|\mathbf{B}^\top(:,j)\|_2$
- $\phi_j = \frac{\mathbf{B}^\top(:,j)}{b_j}$

# Optimized Dynamic Mode Decomposition [Askham and Kutz, 2018]

Stack all the data together

$$\mathbf{X} = [\mathbf{x}(t_1) | \mathbf{x}(t_2) \cdots | \mathbf{x}(t_p)] \in \mathbb{R}^{n \times p}.$$

Optimize directly for DMD matrix decomposition

$$\min_{\mathbf{b}_j, \omega_j, b_j} \|\mathbf{X} - \sum_{j=1}^r \mathbf{b}_j e^{\omega_j t} b_j\|_F.$$

Translate to

$$\min_{\mathbf{A}} \|\mathbf{X}^\top - \Phi(\alpha) \mathbf{B}\|_F.$$

- ▶  $\Phi(\alpha)_{i,j} = \exp(\alpha_j t_i)$
- ▶  $b_j = \|\mathbf{B}^\top(:,j)\|_2$
- ▶  $\phi_j = \frac{\mathbf{B}^\top(:,j)}{b_j}$

Solve via variable projection method.

## Further Reading

## Further Reading

- ▶ Multiverse of DMD [Colbrook, 2023]
- ▶ Generalizing DMD: Modern Koopman theory [Brunton et al., 2021]

# Bonus lab!

*Tutorial 1 here: <https://github.com/PyDMD/PyDMD/blob/master/tutorials/tutorial1/tutorial-1-dmd.ipynb>*

# Nonlinear Dimensionality Reduction

# Outline

1. Multidimensional Scaling (MDS) [Kruskal and Wish, 1978]
2. Isometric Mapping (Isomap) [Balasubramanian and Schwartz, 2002]
3. t-distributed Stochastic Neighborhood Embedding (t-SNE) [Van der Maaten and Hinton, 2008]
4. Uniform Manifold Approximation (UMAP) [McInnes et al., 2018]
5. Koopman Mode Decomposition [Brunton et al., 2021]

# Multidimensional Scaling

# What is MDS?

**Multidimensional Scaling (MDS)** recovers low-dimensional embeddings of abstract objects from a pairwise distance matrix.

# What is MDS?

**Multidimensional Scaling (MDS)** recovers low-dimensional embeddings of abstract objects from a pairwise distance matrix.

**Given:**

- ▶ Abstract points  $\{x_i\}_{i=1}^p \in \mathcal{M}$
- ▶ Distance function  $d : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$
- ▶ Distance matrix  $\mathbf{D} \in \mathbb{R}^{p \times p}$ ,  $d_{ij} = d(x_i, x_j)$

# What is MDS?

**Multidimensional Scaling (MDS)** recovers low-dimensional embeddings of abstract objects from a pairwise distance matrix.

**Given:**

- ▶ Abstract points  $\{x_i\}_{i=1}^p \in \mathcal{M}$
- ▶ Distance function  $d : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}$
- ▶ Distance matrix  $\mathbf{D} \in \mathbb{R}^{p \times p}$ ,  $d_{ij} = d(x_i, x_j)$

**Goal:** Find latent coordinates  $\{\mathbf{z}_i\}_{i=1}^p \subset \mathbb{R}^k$  such that

$$\|\mathbf{z}_i - \mathbf{z}_j\|_2 \approx d(x_i, x_j)$$

Useful when there's a non-Euclidean distance between points in the ambient space.

# Classical MDS Objective

We solve:

$$\{\mathbf{z}_i\}_{i=1}^P = \arg \min_{\{\mathbf{w}_i\} \subset \mathbb{R}^k} \sum_{i,j} (\|\mathbf{w}_i - \mathbf{w}_j\|_2 - d_{ij})^2$$

# Classical MDS Objective

We solve:

$$\{\mathbf{z}_i\}_{i=1}^p = \arg \min_{\{\mathbf{w}_i\} \subset \mathbb{R}^k} \sum_{i,j} (\|\mathbf{w}_i - \mathbf{w}_j\|_2 - d_{ij})^2$$

**Issue:**

# Classical MDS Objective

We solve:

$$\{\mathbf{z}_i\}_{i=1}^p = \arg \min_{\{\mathbf{w}_i\} \subset \mathbb{R}^k} \sum_{i,j} (\|\mathbf{w}_i - \mathbf{w}_j\|_2 - d_{ij})^2$$

**Issue:** The solution is not unique —

# Classical MDS Objective

We solve:

$$\{\mathbf{z}_i\}_{i=1}^p = \arg \min_{\{\mathbf{w}_i\} \subset \mathbb{R}^k} \sum_{i,j} (\|\mathbf{w}_i - \mathbf{w}_j\|_2 - d_{ij})^2$$

**Issue:** The solution is not unique — distances are translation invariant.

# Classical MDS Objective

We solve:

$$\{\mathbf{z}_i\}_{i=1}^p = \arg \min_{\{\mathbf{w}_i\} \subset \mathbb{R}^k} \sum_{i,j} (\|\mathbf{w}_i - \mathbf{w}_j\|_2 - d_{ij})^2$$

**Issue:** The solution is not unique — distances are translation invariant.

**Fix:** Require mean centering:

$$\sum_{i=1}^p \mathbf{x}_i = \mathbf{0}, \quad \sum_{i=1}^p \mathbf{z}_i = \mathbf{0}$$

# Eigenvalue Decomposition

Let  $\mathbf{D}^{(2)}$  be the matrix of squared distances:  $d_{ij}^2$ .

Mean-centering matrix:

$$\mathbf{J} = \mathbf{I} - \frac{1}{p} \mathbf{1} \mathbf{1}^\top$$

Define the Gram matrix:

$$\mathbf{B} = -\frac{1}{2} \mathbf{J} \mathbf{D}^{(2)} \mathbf{J}$$

Compute eigenvalue decomposition:

$$\mathbf{B} \mathbf{V} = \mathbf{V} \Lambda$$

# Low-Dimensional Embedding

Truncate to top- $k$  eigenvalues and eigenvectors:

$$\bar{\Lambda}, \bar{\mathbf{V}}$$

Construct the embedding:

$$\mathbf{Z} = \bar{\mathbf{V}} \bar{\Lambda}^{1/2}$$

$\mathbf{Z}^\top = [\mathbf{z}_1, \dots, \mathbf{z}_n]$  are the low-dimensional coordinates.

**Note:** When  $\mathcal{M} = \mathbb{R}^d$  and  $d(x_i, x_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ , MDS recovers PCA.  
(Details: <https://mlatcl.github.io/advds/notes/mds.pdf>)

# Isometric Mapping

# What is Isomap?

**Isomap (Isometric Mapping)** is a nonlinear dimensionality reduction method that:

- ▶ Preserves **geodesic distances** (not just Euclidean distances)
- ▶ Extends classical MDS to nonlinear manifolds

**Key idea:** Approximate manifold distances with shortest paths on a neighborhood graph, then apply MDS.

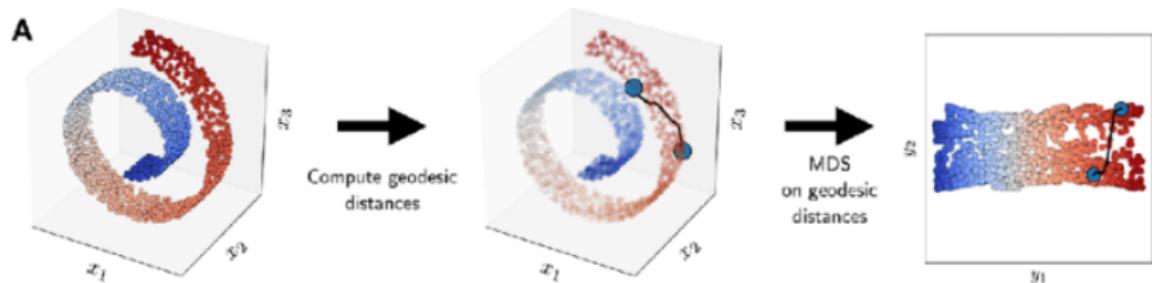


Figure: Figure borrowed from Tsai [2010].

# Isomap vs MDS

The advantage of a nearest-neighbors approach...

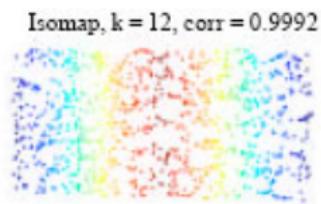
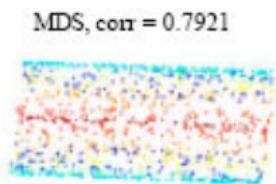
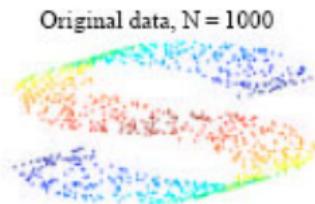


Figure: Figure borrowed from Tsai [2010].

# Steps of Isomap

1. Construct a graph  $G = (V, E)$  where:
  - ▶  $V = \mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$
  - ▶ Edges  $E$  defined using:
    - ▶  $k$ -nearest neighbors or
    - ▶  $\varepsilon$ -radius ball
2. Compute geodesic distance matrix  $D^{\text{geo}} \in \mathbb{R}^{n \times n}$ :
  - ▶ Use shortest path distances in  $G$
  - ▶ Dijkstra's or Floyd–Warshall algorithm
3. Apply classical MDS to  $D^{\text{geo}}$

## When to Use Isomap

- ▶ Your data lies on a low-dimensional **nonlinear** manifold
- ▶ You want a global embedding that preserves intrinsic geometry
- ▶ You want a distance-preserving map like MDS, but for curved manifolds

**Note:** Isomap reduces to MDS if the data lies in Euclidean space and the full distance matrix is used.

# Lab 5: MDS & Isomap

*Go to `Lab5_IsomapMDS.ipynb`*

# t-SNE

## Stochastic Neighbor Embedding (SNE): Overview

SNE constructs a probability distribution over neighbors for each point  $\mathbf{x}_i$  based on distances in high-dimensional space.

$$g(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}\|_2^2}{2\sigma_i^2}\right).$$

## Stochastic Neighbor Embedding (SNE): Overview

SNE constructs a probability distribution over neighbors for each point  $\mathbf{x}_i$  based on distances in high-dimensional space.

$$g(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}\|_2^2}{2\sigma_i^2}\right).$$

The conditional probability that  $\mathbf{x}_i$  picks  $\mathbf{x}_j$  as neighbor:

$$p_{j|i} = \frac{g(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{r \neq i} g(\mathbf{x}_i, \mathbf{x}_r)}.$$

# Stochastic Neighbor Embedding (SNE): Overview

SNE constructs a probability distribution over neighbors for each point  $\mathbf{x}_i$  based on distances in high-dimensional space.

$$g(\mathbf{x}_i, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}\|_2^2}{2\sigma_i^2}\right).$$

The conditional probability that  $\mathbf{x}_i$  picks  $\mathbf{x}_j$  as neighbor:

$$p_{j|i} = \frac{g(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{r \neq i} g(\mathbf{x}_i, \mathbf{x}_r)}.$$

Note:  $p_{j|i} \neq p_{i|j}$  due to different  $\sigma_i$  values.

# Low-dimensional Embedding and Objective

Embed points  $\mathbf{z}_i \in \mathbb{R}^k$  with kernel

$$h(\mathbf{z}_i, \mathbf{z}_j) = \exp(-\|\mathbf{z}_i - \mathbf{z}_j\|_2^2),$$

and conditional probabilities

$$q_{j|i} = \frac{h(\mathbf{z}_i, \mathbf{z}_j)}{\sum_{r \neq i} h(\mathbf{z}_i, \mathbf{z}_r)}.$$

SNE minimizes the KL divergence between these distributions:

$$C = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

SNE uses gradient descent on  $\{\mathbf{z}_i\}$ , where gradients resemble forces pulling embedded points closer or pushing them apart.

SNE minimizes the difference between two graphs

SNE minimizes the KL divergence between these distributions:

$$C = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

- ▶  $p_{j|i}$  probability that  $x_i$  picks  $x_j$  as its neighbor in ambient space
- ▶  $q_{j|i}$  probability that  $x_i$  picks  $x_j$  as its neighbor in reduced space

# Perplexity and Bandwidth Selection

Perplexity controls local scale for each point:

$$\text{Perp}(P_i) = 2^{H(P_i)}, \quad H(P_i) = - \sum_j p_{j|i} \log_2 p_{j|i}.$$

- ▶ Low perplexity  $\implies$  tight local neighborhoods
- ▶ High perplexity  $\implies$  broader neighborhoods

Binary search is used to find  $\sigma_i$  so that the perplexity matches a user-defined value.

- ▶ Small  $\sigma_i$   $\implies$  dense distribution
- ▶ Large  $\sigma_i$   $\implies$  sparse distribution

# Challenges of SNE

1. **Crowding problem:** Moderate distances in high-D collapse into tight clusters in low-D.
2. **Asymmetry:**  $p_{j|i} \neq p_{i|j}$  complicates interpretation.
3. **Optimization difficulty:** KL divergence is hard to optimize and sensitive.

# t-SNE: Symmetrized Probabilities

**t-SNE** improves on SNE by:

- ▶ Using a **joint probability** distribution

$$p_{ij} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2) + \exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2/2\sigma_j^2)}{2 \sum_{r \neq s} \exp(-\|\mathbf{x}_r - \mathbf{x}_s\|^2/2\sigma_r^2)}$$

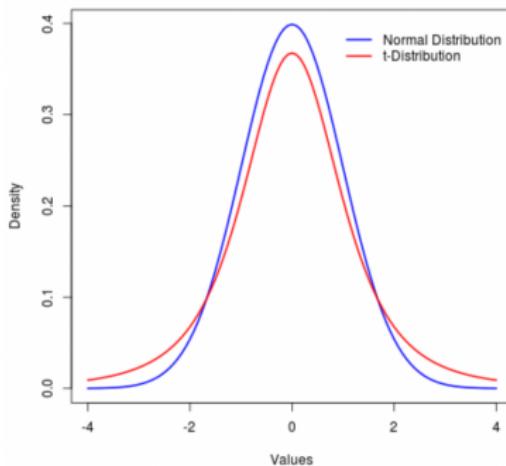
- ▶ Modeling  $q_{ij}$  in the low-dimensional space with a **Student-t distribution**:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{r \neq s} (1 + \|\mathbf{y}_r - \mathbf{y}_s\|^2)^{-1}}$$

# t-SNE: Fixing the Crowding Problem

## Why use Student-t?

The heavy tails of the t-distribution allow moderate distances to be represented more accurately — solving the **crowding problem**.



# t-SNE: Gradient Optimization

t-SNE minimizes the KL divergence between  $p_{ij}$  and  $q_{ij}$ :

$$C = \sum_{i,j} p_{ij} \log \left( \frac{p_{ij}}{q_{ij}} \right)$$

**Gradient with respect to  $\mathbf{y}_i$ :**

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$$

**Interpretation:** Points repel or attract depending on similarity; repulsion is bounded (unlike in SNE).

# Parameter Tuning

## **t-SNE parameters:**

- ▶ **Perplexity** (typical range: 5–50)
- ▶ **Learning rate**
- ▶ **Number of iterations**

**Effect of tuning:** t-SNE is sensitive to parameter choices

**See:** a website for parameter tuning intuition

# Summary

- ▶ SNE and t-SNE preserve local structure using probability distributions
- ▶ t-SNE improves SNE via:
  - ▶ Symmetric probabilities
  - ▶ Heavy-tailed Student-t distribution
  - ▶ Better optimization
- ▶ Widely used for visualizing high-dimensional data

# Lab 6: t-SNE

*Go to Lab6\_tSNE.ipynb &*  
<https://distill.pub/2016/misread-tsne/>

UMAP

# Uniform Manifold Approximation (UMAP)

**Goal:** preserve the topological structure of the data manifold

# Uniform Manifold Approximation (UMAP)

**Goal:** preserve the topological structure of the data manifold

**Assumptions:**

1. There exists an underlying manifold with uniform data distribution
2. The manifold is locally connected

## Inputs

Given dataset  $\{\mathbf{x}_1, \dots, \mathbf{x}_p\}$

# Inputs

Given dataset  $\{\mathbf{x}_1, \dots, \mathbf{x}_p\}$

## User-specified:

- ▶ Number of neighbors  $n$
- ▶ Metric  $d$
- ▶ Output dimension  $k$
- ▶ Minimum distance  $\min\_dist$

# Inputs

Given dataset  $\{\mathbf{x}_1, \dots, \mathbf{x}_p\}$

## User-specified:

- ▶ Number of neighbors  $n$
- ▶ Metric  $d$
- ▶ Output dimension  $k$
- ▶ Minimum distance  $\min\_dist$

## Choice of $n$ :

- ▶ Small  $n \Rightarrow$  fine detail structure (local)
- ▶ Large  $n \Rightarrow$  coarser global structure

# Inputs

Given dataset  $\{\mathbf{x}_1, \dots, \mathbf{x}_p\}$

## User-specified:

- ▶ Number of neighbors  $n$
- ▶ Metric  $d$
- ▶ Output dimension  $k$
- ▶ Minimum distance  $\min\_dist$

## Choice of $n$ :

- ▶ Small  $n \Rightarrow$  fine detail structure (local)
- ▶ Large  $n \Rightarrow$  coarser global structure

*UMAP is based in category theory & mathematical topology.. we will sweep this foundation under the rug and consider the computational machinery instead*

## Algorithm Step 1: AmbientGraph Building

- a. Define number of neighbors as  $n$  & minimum distance between  $\mathbf{x}_i$  and neighbors  $\rho_i$
- b. Find  $\sigma_i$  via binary search:

$$\sum_{j=1}^n \exp\left(\frac{-\max(0, d(\mathbf{x}_i, \mathbf{x}_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(n)$$

- c. Edge weights from  $i$  to  $i_j$ :

$$\mathbf{w}_h(\mathbf{x}_i, \mathbf{x}_{i_j}) = \exp\left(\frac{-\max(0, d(\mathbf{x}_i, \mathbf{x}_{i_j}) - \rho_i)}{\sigma_i}\right)$$

- d. Let  $\mathbf{A}$  be adjacency matrix, then:

$$\mathbf{B} = \mathbf{A} + \mathbf{A}^\top - \mathbf{A} \circ \mathbf{A}^\top$$

*Interpretation:* edge weights are probabilities that a 1-simplex exists;  
combined weights: probability that at least one edge exists.

## Algorithm Step 1: Reduced Graph Building

Edge weights between  $\mathbf{z}_i$  and  $\mathbf{z}_j$  in reduced space are

$$\mathbf{w}_I(\mathbf{z}_i, \mathbf{z}_j) = \frac{1}{a\|\mathbf{z}_i - \mathbf{z}_j\|_2^{2b}}.$$

Where  $a$  and  $b$  be hyperparameters chosen by a non-linear least squares fit to

$$\begin{cases} 1 & \|\mathbf{z}_i - \mathbf{z}_j\|_2 < \text{min\_dist} \\ \exp(-\|\mathbf{z}_i - \mathbf{z}_j\|_2 - \text{min\_dist}) & \text{otherwise} \end{cases}$$

*Default parameters:*  $a = 1.577$ ,  $b = 0.8951$

## Algorithm Step 1: Reduced Graph Building

Edge weights between  $\mathbf{z}_i$  and  $\mathbf{z}_j$  in reduced space are

$$\mathbf{w}_I(\mathbf{z}_i, \mathbf{z}_j) = \frac{1}{a\|\mathbf{z}_i - \mathbf{z}_j\|_2^{2b}}.$$

Where  $a$  and  $b$  be hyperparameters chosen by a non-linear least squares fit to

$$\begin{cases} 1 & \|\mathbf{z}_i - \mathbf{z}_j\|_2 < \text{min\_dist} \\ \exp(-\|\mathbf{z}_i - \mathbf{z}_j\|_2 - \text{min\_dist}) & \text{otherwise} \end{cases}$$

*Default parameters:*  $a = 1.577$ ,  $b = 0.8951$

But, when  $a = b = 1$  edges are sampled from Student-t distribution.

## Algorithm Step 2: Optimization

## Algorithm Step 2: Optimization

- ▶ Initialize with spectral embedding (e.g., Laplacian Eigenmaps from Belkin and Niyogi [2001])

## Algorithm Step 2: Optimization

- ▶ Initialize with spectral embedding (e.g., Laplacian Eigenmaps from Belkin and Niyogi [2001])
- ▶ Use stochastic gradient descent to minimize cross-entropy:

## Algorithm Step 2: Optimization

- ▶ Initialize with spectral embedding (e.g., Laplacian Eigenmaps from Belkin and Niyogi [2001])
- ▶ Use stochastic gradient descent to minimize cross-entropy:

$$C = \sum_e w_h(e) \log \left( \frac{w_h(e)}{w_l(e)} \right) + (1 - w_h(e)) \log \left( \frac{1 - w_h(e)}{1 - w_l(e)} \right)$$

# Advantages of UMAP

- ▶ Preserves global and local structure
- ▶ Faster than t-SNE ( $\mathcal{O}(n \log n)$  vs  $\mathcal{O}(n^2)$ )
- ▶ Based on topology + category theory
- ▶ Can be used for general dimensionality reduction (not just visualization)
- ▶ Approximately invertible
- ▶ Aligned UMAP for cross-dataset comparison

# Parameter Tuning & Resources

## Key Parameters:

- ▶ **Number of neighbors:** balance local vs. global structure
- ▶ **Minimum distance:** desired separation of points in latent space

Visualizing parameter tuning and comparison to t-SNE

## Resources:

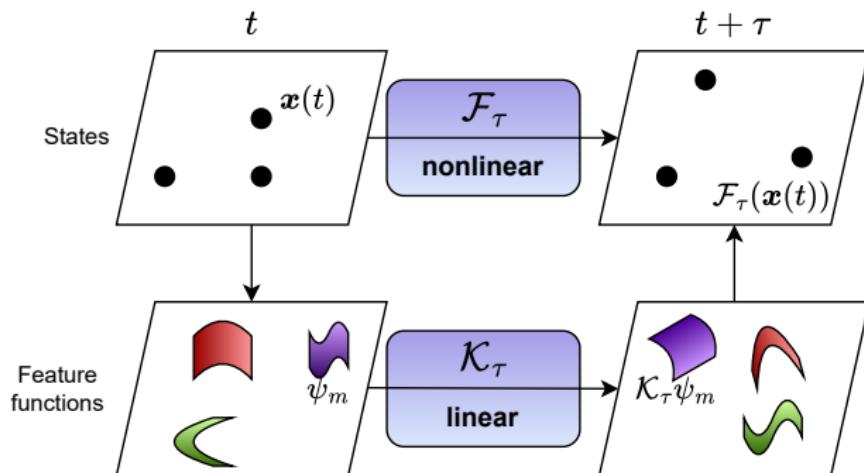
- ▶ UMAP: Uniform Manifold Approximation and Projection
- ▶ UMAP Documentation
- ▶ Understanding UMAP
- ▶ Inverse transform
- ▶ Aligned UMAP

# Lab 7: UMAP

*Go to Lab7\_UMAP.ipynb &*  
[`https://pair-code.github.io/understanding-umap/`](https://pair-code.github.io/understanding-umap/)

# Koopman Mode Decomposition

# Koopman Mode Decomposition



# Koopman Mode Decomposition

# Koopman Mode Decomposition

$\psi_m$  “observable” (a.k.a. feature) function

# Koopman Mode Decomposition

$\psi_m$  “observable” (a.k.a. feature) function **Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

# Koopman Mode Decomposition

$\psi_m$  “observable” (a.k.a. feature) function **Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

**Koopman mode decomposition** [Rowley et al., 2009; Mezić, 2013]

$$\mathbf{x}(t + \tau) = \sum_{j=1}^r \mathbf{b}_j e^{\omega_j \tau} \phi_j(\mathbf{x}(t)).$$

# Koopman Mode Decomposition

$\psi_m$  “observable” (a.k.a. feature) function **Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

**Koopman mode decomposition** [Rowley et al., 2009; Mezić, 2013]

$$\mathbf{x}(t + \tau) = \sum_{j=1}^r \mathbf{b}_j e^{\omega_j \tau} \phi_j(\mathbf{x}(t)).$$

- ▶  $\mathbf{b}_j$  **spatial patterns** (dynamic modes)

# Koopman Mode Decomposition

$\psi_m$  “observable” (a.k.a. feature) function **Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

**Koopman mode decomposition** [Rowley et al., 2009; Mezić, 2013]

$$\mathbf{x}(t + \tau) = \sum_{j=1}^r \mathbf{b}_j e^{\omega_j \tau} \phi_j(\mathbf{x}(t)).$$

- ▶  **$\mathbf{b}_j$  spatial patterns** (dynamic modes)
- ▶  **$\omega_j$  temporal characteristics** (continuous time eigenvalues)

# Koopman Mode Decomposition

$\psi_m$  “observable” (a.k.a. feature) function **Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

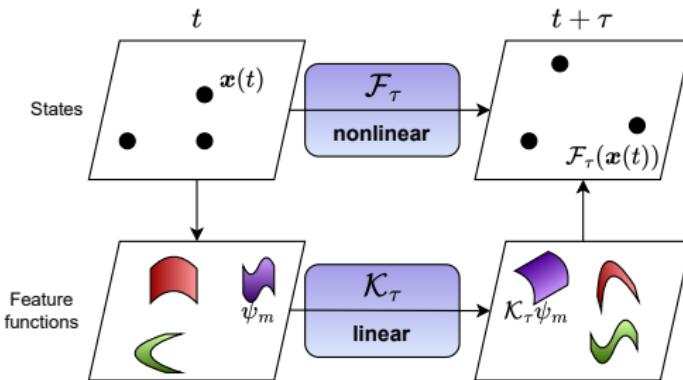
$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

**Koopman mode decomposition** [Rowley et al., 2009; Mezić, 2013]

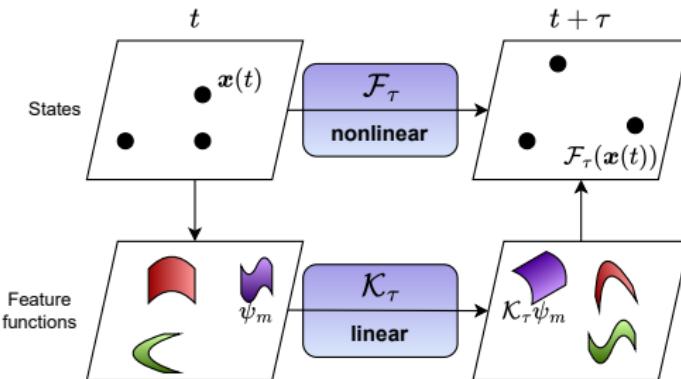
$$\mathbf{x}(t + \tau) = \sum_{j=1}^r \mathbf{b}_j e^{\omega_j \tau} \phi_j(\mathbf{x}(t)).$$

- ▶  **$\mathbf{b}_j$  spatial patterns** (dynamic modes)
- ▶  **$\omega_j$  temporal characteristics** (continuous time eigenvalues)
- ▶  $\phi_j$  Koopman eigenfunctions

# Koopman Mode Decomposition



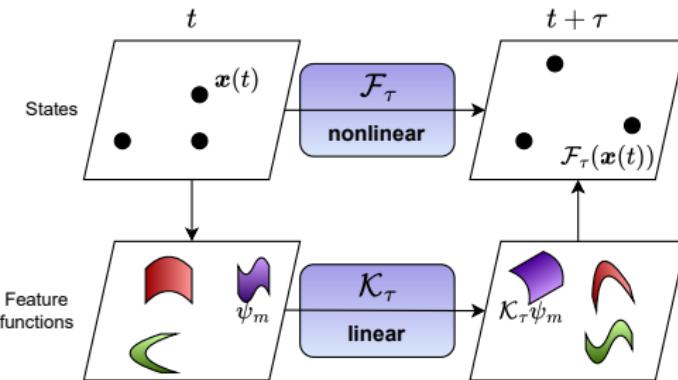
# Koopman Mode Decomposition



**Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

# Koopman Mode Decomposition



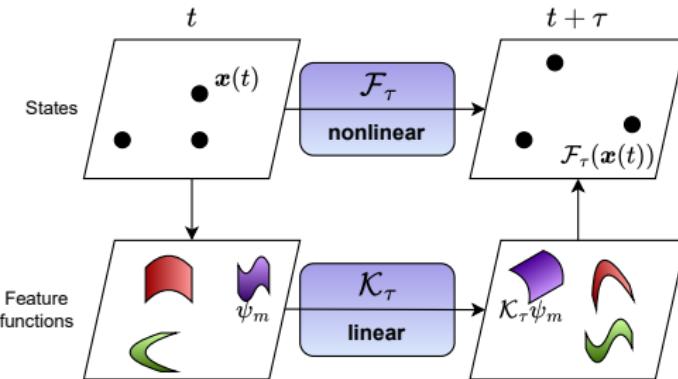
**Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

**Koopman mode decomposition** [Rowley et al., 2009; Mezić, 2013]

$$\mathbf{x}(t + \tau) = \sum_{j=1}^r \mathbf{b}_j e^{\omega_j \tau} \phi_j(\mathbf{x}(t)).$$

# Koopman Mode Decomposition



**Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

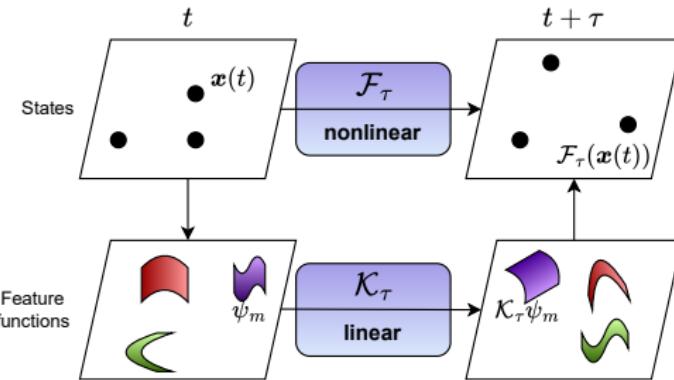
$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

**Koopman mode decomposition** [Rowley et al., 2009; Mezić, 2013]

$$\mathbf{x}(t + \tau) = \sum_{j=1}^r \mathbf{b}_j e^{\omega_j \tau} \phi_j(\mathbf{x}(t)).$$

- ▶  $\mathbf{b}_j$  spatial patterns (dynamic modes)

# Koopman Mode Decomposition



**Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

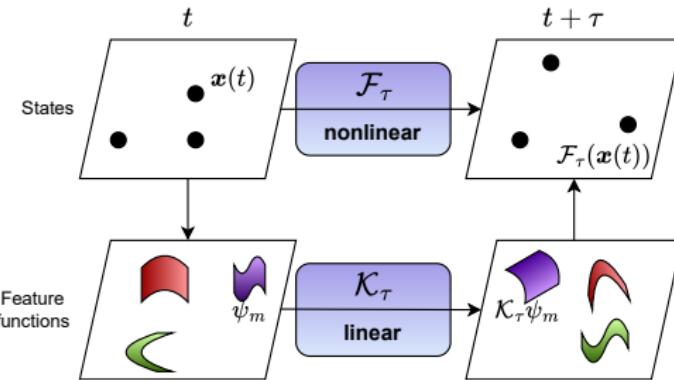
$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

**Koopman mode decomposition** [Rowley et al., 2009; Mezić, 2013]

$$\mathbf{x}(t + \tau) = \sum_{j=1}^r \mathbf{b}_j e^{\omega_j \tau} \phi_j(\mathbf{x}(t)).$$

- ▶  $\mathbf{b}_j$  spatial patterns (dynamic modes)
- ▶  $\omega_j$  temporal characteristics (continuous time eigenvalues)

# Koopman Mode Decomposition



**Koopman operator** (denoted  $\mathcal{K}_\tau$ ) [Koopman, 1931]

$$\mathcal{K}_\tau[\psi_m](\mathbf{x}(t)) = \psi_m(\mathcal{F}_\tau \mathbf{x}(t)) = \psi_m(\mathbf{x}(t + \tau))$$

**Koopman mode decomposition** [Rowley et al., 2009; Mezić, 2013]

$$\mathbf{x}(t + \tau) = \sum_{j=1}^r \mathbf{b}_j e^{\omega_j \tau} \phi_j(\mathbf{x}(t)).$$

- ▶  $\mathbf{b}_j$  spatial patterns (dynamic modes)
- ▶  $\omega_j$  temporal characteristics (continuous time eigenvalues)
- ▶  $\phi_j$  Koopman eigenfunctions

# Kernel Koopman Mode Decomposition [Williams et al., 2015]

## Kernel Koopman Mode Decomposition [Williams et al., 2015]

- ▶ Embed states into a Reproducing Kernel Hilbert Space (RKHS), denoted  $\mathcal{H}$

## Kernel Koopman Mode Decomposition [Williams et al., 2015]

- ▶ Embed states into a Reproducing Kernel Hilbert Space (RKHS), denoted  $\mathcal{H}$
- ▶ Let  $\psi : \mathbb{R}^M \rightarrow \mathcal{H}$  be the feature map for some kernel  $k(\cdot, \cdot)$

## Kernel Koopman Mode Decomposition [Williams et al., 2015]

- ▶ Embed states into a Reproducing Kernel Hilbert Space (RKHS), denoted  $\mathcal{H}$
- ▶ Let  $\psi : \mathbb{R}^M \rightarrow \mathcal{H}$  be the feature map for some kernel  $k(\cdot, \cdot)$
- ▶ Define the snapshot feature matrices

## Kernel Koopman Mode Decomposition [Williams et al., 2015]

- ▶ Embed states into a Reproducing Kernel Hilbert Space (RKHS), denoted  $\mathcal{H}$
- ▶ Let  $\psi : \mathbb{R}^M \rightarrow \mathcal{H}$  be the feature map for some kernel  $k(\cdot, \cdot)$
- ▶ Define the snapshot feature matrices

$$\Psi_{\mathbf{x}} = \left[ \psi(\mathbf{x}(1)) \cdots \psi(\mathbf{x}(T-\tau)) \right] \quad (6)$$

$$\Psi_{\mathbf{x}'} = \left[ \psi(\mathbf{x}(\tau)) \cdots \psi(\mathbf{x}'(T)) \right]. \quad (7)$$

## Kernel Koopman Mode Decomposition [Williams et al., 2015]

- ▶ Embed states into a Reproducing Kernel Hilbert Space (RKHS), denoted  $\mathcal{H}$
- ▶ Let  $\psi : \mathbb{R}^M \rightarrow \mathcal{H}$  be the feature map for some kernel  $k(\cdot, \cdot)$
- ▶ Define the snapshot feature matrices

$$\Psi_{\mathbf{x}} = \left[ \psi(\mathbf{x}(1)) \cdots \psi(\mathbf{x}(T-\tau)) \right] \quad (6)$$

$$\Psi_{\mathbf{x}'} = \left[ \psi(\mathbf{x}(\tau)) \cdots \psi(\mathbf{x}'(T)) \right]. \quad (7)$$

- ▶ Seek finite-dimensional Koopman operator estimate is  $\tilde{\mathbf{K}}$ ,

# Kernel Koopman Mode Decomposition [Williams et al., 2015]

- ▶ Embed states into a Reproducing Kernel Hilbert Space (RKHS), denoted  $\mathcal{H}$
- ▶ Let  $\psi : \mathbb{R}^M \rightarrow \mathcal{H}$  be the feature map for some kernel  $k(\cdot, \cdot)$
- ▶ Define the snapshot feature matrices

$$\Psi_{\mathbf{x}} = \left[ \psi(\mathbf{x}(1)) \cdots \psi(\mathbf{x}(T-\tau)) \right] \quad (6)$$

$$\Psi_{\mathbf{x}'} = \left[ \psi(\mathbf{x}(\tau)) \cdots \psi(\mathbf{x}'(T)) \right]. \quad (7)$$

- ▶ Seek finite-dimensional Koopman operator estimate is  $\tilde{\mathbf{K}}$ , which can be found via kernel Gram matrices

$$\mathbf{K}_{\mathbf{x}} = \Psi_{\mathbf{x}}^\top \Psi_{\mathbf{x}}, \quad \mathbf{K}_{\mathbf{x}', \mathbf{x}} = \Psi_{\mathbf{x}'}^\top \Psi_{\mathbf{x}}.$$

# Kernel Koopman Mode Decomposition

# Kernel Koopman Mode Decomposition

- ▶ Using the eigendecomposition  $\mathbf{K}_x = \mathbf{Q}\Sigma^2\mathbf{Q}^\top$

## Kernel Koopman Mode Decomposition

- ▶ Using the eigendecomposition  $\mathbf{K}_x = \mathbf{Q}\Sigma^2\mathbf{Q}^\top$
- ▶ The finite-dimensional Koopman operator estimate is  
$$\tilde{\mathbf{K}} = \Sigma^{-1}\mathbf{Q}^\top \mathbf{K}_{x',x} \mathbf{Q}\Sigma^{-1}$$

## Kernel Koopman Mode Decomposition

- ▶ Using the eigendecomposition  $\mathbf{K}_x = \mathbf{Q}\Sigma^2\mathbf{Q}^\top$
- ▶ The finite-dimensional Koopman operator estimate is  
$$\tilde{\mathbf{K}} = \Sigma^{-1}\mathbf{Q}^\top \mathbf{K}_{x',x} \mathbf{Q}\Sigma^{-1}$$
- ▶ Eigenvalue decomposition  $\tilde{\mathbf{K}}\mathbf{W} = \mathbf{W}\Lambda$

# Kernel Koopman Mode Decomposition

- ▶ Using the eigendecomposition  $\mathbf{K}_x = \mathbf{Q}\Sigma^2\mathbf{Q}^\top$
- ▶ The finite-dimensional Koopman operator estimate is  $\tilde{\mathbf{K}} = \Sigma^{-1}\mathbf{Q}^\top \mathbf{K}_{x',x} \mathbf{Q}\Sigma^{-1}$
- ▶ Eigenvalue decomposition  $\tilde{\mathbf{K}}\mathbf{W} = \mathbf{W}\Lambda$

---

Name	Definition
Eigenfunctions of $\mathbf{X}$ Koopman modes	$\Phi = \mathbf{K}_x \mathbf{Q} \Sigma^\dagger \mathbf{W}$ $\mathbf{B} = \mathbf{X} \mathbf{Q}^\top \Sigma^\dagger \mathbf{W}^{-1}$

---

# Bonus lab!

*Tutorials here:*

- ▶ <https://github.com/PyDMD/PyDMD/blob/master/tutorials/tutorial17/tutorial-17-edmd.ipynb>
- ▶ <https://github.com/dynamicslab/pykoopman>

# Autoencoders

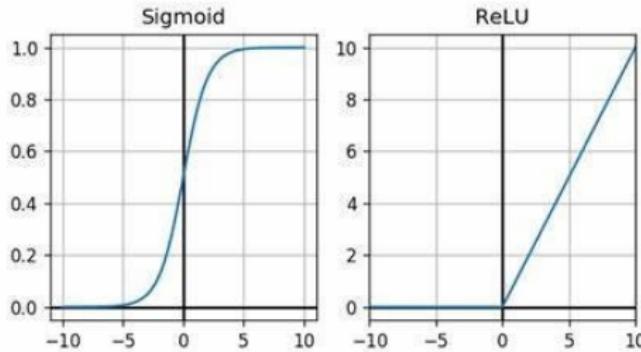
# Outline

1. Vanilla Autoencoders
2. Denoising Autoencoders
3. Variational Autoencoders
4. Convolutional Autoencoders

# Neural Networks

Composed of linear and nonlinear functions.

- ▶ **Linear functions:**  $W \in \mathbb{R}^{\text{out} \times \text{in}}$
- ▶ **Nonlinear (activation) functions:**  $\sigma$

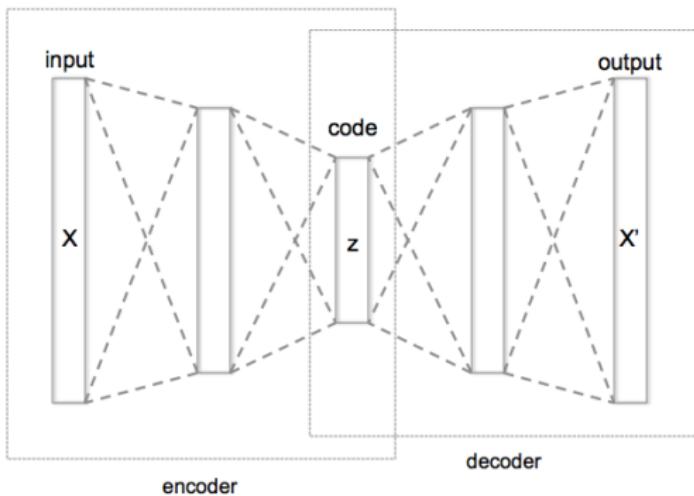


**The neural network function** maps

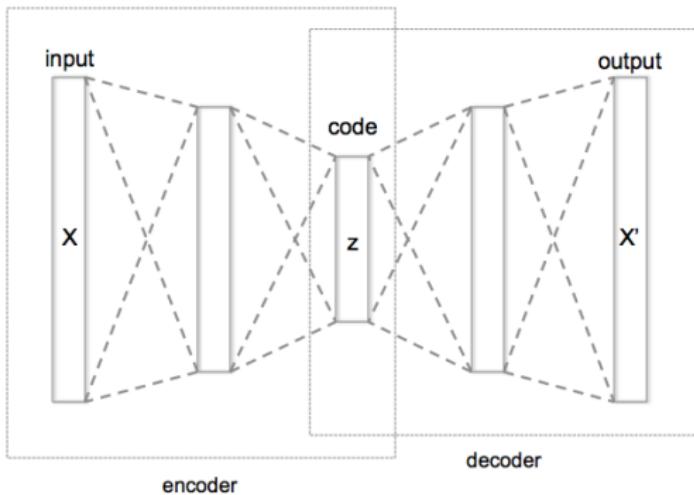
$$\mathbf{x} \mapsto \sigma(\mathbf{W}_\ell \sigma(\mathbf{W}_{\ell-1} \cdots \sigma(\mathbf{W}_1(\mathbf{x})) \cdots))$$

# Vanilla Autoencoder

# Vanilla Autoencoder

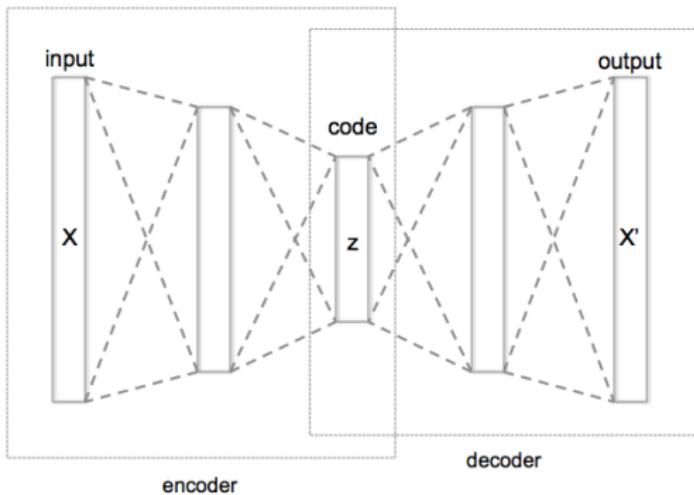


# Vanilla Autoencoder



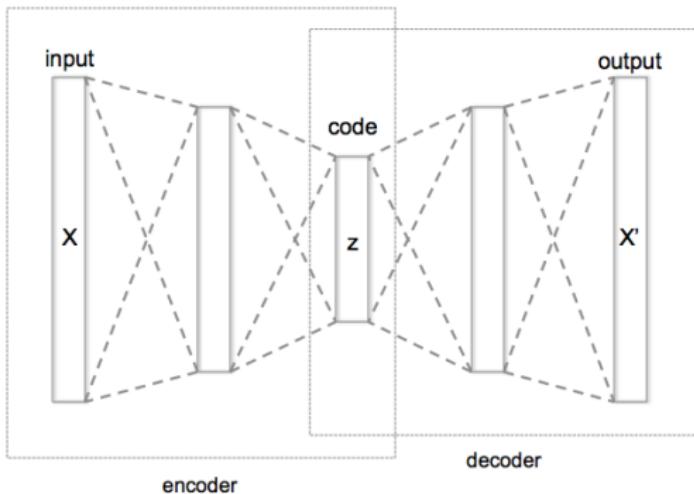
- ▶ Encoder compresses the input into a lower-D representation

# Vanilla Autoencoder



- ▶ Encoder compresses the input into a lower-D representation
- ▶ Decoder reconstructs the input from the lower-D representation

# Vanilla Autoencoder



- ▶ Encoder compresses the input into a lower-D representation
- ▶ Decoder reconstructs the input from the lower-D representation
- ▶ Loss function: Typically Mean Squared Error (MSE) between the input and the reconstructed output

# Autoencoders vs PCA

***Autoencoders*** optimize

$$\arg \min_{\text{Enc}, \text{Dec}} \sum_{i=1}^n L(\mathbf{x}_i, \text{Dec} \circ \text{Enc}(\mathbf{x}_i)).$$

- ▶ Define  $\text{Enc}(\mathbf{x}) = \mathbf{A}^\top \mathbf{x}$  and  $\text{Dec}(\mathbf{z}) = \mathbf{A}\mathbf{z}$ .
- ▶ Take the mean squared error (MSE) loss  $L(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$ .

$$\arg \min_{\mathbf{A}^\top \mathbf{A} = \mathbf{I}} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{A}\mathbf{A}^\top \mathbf{x}_i\|_2^2.$$

- ▶ This is *exactly* PCA

But, there's a deeper connection... see Rolinek et al. [2019]; Bao et al. [2020].

## Denoising Autoencoder [Vincent et al., 2008]

## Denoising Autoencoder [Vincent et al., 2008]

- ▶ Similar to the vanilla autoencoder, but with ***noisy input data***

## Denoising Autoencoder [Vincent et al., 2008]

- ▶ Similar to the vanilla autoencoder, but with ***noisy input data***
- ▶ The goal is to reconstruct the clean data from the noisy version

## Denoising Autoencoder [Vincent et al., 2008]

- ▶ Similar to the vanilla autoencoder, but with ***noisy input data***
- ▶ The goal is to reconstruct the clean data from the noisy version
- ▶ Used for removing noise from data (e.g., images or signals)

## Denoising Autoencoder [Vincent et al., 2008]

- ▶ Similar to the vanilla autoencoder, but with ***noisy input data***
- ▶ The goal is to reconstruct the clean data from the noisy version
- ▶ Used for removing noise from data (e.g., images or signals)
- ▶ Loss function is also typically MSE, between clean data and reconstructions

# Variational Autoencoder

# Variational Autoencoder (VAE)

# Variational Autoencoder (VAE)

Goal: estimate true data distribution  $p_*$  given model parameters  $\theta$ .

# Variational Autoencoder (VAE)

Goal: estimate true data distribution  $p_*$  given model parameters  $\theta$ .

- ▶ **Estimated distribution:**  $p_\theta$
- ▶ Maximize  $p_\theta(\mathbf{x})$  for each  $\mathbf{x}$  in dataset.

# Variational Autoencoder (VAE)

Goal: estimate true data distribution  $p_*$  given model parameters  $\theta$ .

- ▶ **Estimated distribution:**  $p_\theta$
- ▶ Maximize  $p_\theta(\mathbf{x})$  for each  $\mathbf{x}$  in dataset.

*Intuition*  $p_*$  can be described by compressed features.

# Variational Autoencoder (VAE)

Goal: estimate true data distribution  $p_*$  given model parameters  $\theta$ .

- ▶ **Estimated distribution:**  $p_\theta$
- ▶ Maximize  $p_\theta(\mathbf{x})$  for each  $\mathbf{x}$  in dataset.

*Intuition*  $p_*$  can be described by compressed features. E.g., reduced (a.k.a. latent) representation of a dataset of pictures of circles could be radius and center.

# Variational Autoencoder (VAE)

Goal: estimate true data distribution  $p_*$  given model parameters  $\theta$ .

- ▶ **Estimated distribution:**  $p_\theta$
- ▶ Maximize  $p_\theta(\mathbf{x})$  for each  $\mathbf{x}$  in dataset.

*Intuition*  $p_*$  can be described by compressed features. E.g., reduced (a.k.a. latent) representation of a dataset of pictures of circles could be radius and center.

- ▶ Inference of latent space space:  $p_\theta(\mathbf{z}|\mathbf{x})$  (likelihood)
- ▶ Generation from latent space:  $p_\theta(\mathbf{x}|\mathbf{z})$  (posterior)

# Variational Autoencoder (VAE)

Goal: estimate true data distribution  $p_*$  given model parameters  $\theta$ .

- ▶ **Estimated distribution:**  $p_\theta$
- ▶ Maximize  $p_\theta(\mathbf{x})$  for each  $\mathbf{x}$  in dataset.

*Intuition*  $p_*$  can be described by compressed features. E.g., reduced (a.k.a. latent) representation of a dataset of pictures of circles could be radius and center.

- ▶ Inference of latent space space:  $p_\theta(\mathbf{z}|\mathbf{x})$  (likelihood)
- ▶ Generation from latent space:  $p_\theta(\mathbf{x}|\mathbf{z})$  (posterior)

**Problem:** Marginal likelihood is intractable

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int_{\mathbf{z}} p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$

# Enter Bayes

# Enter Bayes

Bayes Rule:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\theta}(\mathbf{x})}$$

# Enter Bayes

Bayes Rule:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\theta}(\mathbf{x})}$$

- ▶ prior  $p_{\theta}(\mathbf{x})$
- ▶ likelihood  $p_{\theta}(\mathbf{z}|\mathbf{x})$
- ▶ marginal likelihood  $p_{\theta}(\mathbf{x})$
- ▶ posterior probability  $p_{\theta}(\mathbf{x}|\mathbf{z})$

# Enter Bayes

Bayes Rule:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\theta}(\mathbf{x})}$$

- ▶ prior  $p_{\theta}(\mathbf{x})$
- ▶ likelihood  $p_{\theta}(\mathbf{z}|\mathbf{x})$
- ▶ marginal likelihood  $p_{\theta}(\mathbf{x})$
- ▶ posterior probability  $p_{\theta}(\mathbf{x}|\mathbf{z})$

Maximize prior equivalent to maximizing

$$p_{\theta}(\mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x})}$$

# Enter Bayes

Bayes Rule:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\theta}(\mathbf{x})}$$

- ▶ prior  $p_{\theta}(\mathbf{x})$
- ▶ likelihood  $p_{\theta}(\mathbf{z}|\mathbf{x})$
- ▶ marginal likelihood  $p_{\theta}(\mathbf{x})$
- ▶ posterior probability  $p_{\theta}(\mathbf{x}|\mathbf{z})$

Maximize prior equivalent to maximizing

$$p_{\theta}(\mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x})}$$

But  $p_{\theta}(\mathbf{z}|\mathbf{x})$  is also intractible...

# Enter Bayes

Bayes Rule:

$$p_{\theta}(\mathbf{z}|\mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\theta}(\mathbf{x})}$$

- ▶ prior  $p_{\theta}(\mathbf{x})$
- ▶ likelihood  $p_{\theta}(\mathbf{z}|\mathbf{x})$
- ▶ marginal likelihood  $p_{\theta}(\mathbf{x})$
- ▶ posterior probability  $p_{\theta}(\mathbf{x}|\mathbf{z})$

Maximize prior equivalent to maximizing

$$p_{\theta}(\mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\theta}(\mathbf{x})} = \frac{p_{\theta}(\mathbf{x}, \mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x})}$$

But  $p_{\theta}(\mathbf{z}|\mathbf{x})$  is also intractible...

But we can estimate  $p_{\theta}(\mathbf{z}|\mathbf{x})$

## Estimate $p_\theta(\mathbf{z}|\mathbf{x})$

**Goal:** approximate  $p = p_\theta(\mathbf{z}|\mathbf{x})$  with  $q = q_\phi(\mathbf{z}|\mathbf{x})$ .

### KL Divergence

$$D_{\text{KL}}(q||p) = \int_{\mathbf{z}} q \log(q/p) d\mathbf{z} \quad (8)$$

$$= \mathbb{E}_q [\log(q/p)] \quad (9)$$

$$= \mathbb{E}_q [\log q] - \mathbb{E}_q [\log p] \quad (10)$$

$$= \mathbb{E}_q [\log q] - \mathbb{E}_q [\log(p_\theta(\mathbf{z}, \mathbf{x})/p_\theta(\mathbf{x}))] \quad (11)$$

$$= \mathbb{E}_q [\log q] - \mathbb{E}_q [\log p_\theta(\mathbf{z}, \mathbf{x})] + \mathbb{E}_q [\log p_\theta(\mathbf{x})] \quad (12)$$

$$= \log p_\theta(\mathbf{x}) - \mathbb{E}_q [\log p_\theta(\mathbf{z}, \mathbf{x}) + \log q_\phi(\mathbf{x})] \quad (13)$$

The expected information loss when we model  $p$  with  $q$ .

**Problem:**  $\log p_\theta(\mathbf{x})$  intractible

# More with KL Divergence

From before, we have

$$\log p_\theta(x) = D_{\text{KL}}(q||p) + \mathbb{E}_q[\log p_\theta(\mathbf{z}, \mathbf{x}) + \log q_\theta(x)]$$

Notice  $D_{\text{KL}}(q||p) > 0$

Enter the Evidence Lower BOund! – **ELBO**

$$\log p_\theta(x) \geq \mathbb{E}_q[\log(p_\theta(\mathbf{z}, \mathbf{x}) + \log(q_\theta(x))] = \text{ELBO}$$

# The Glory of ELBO

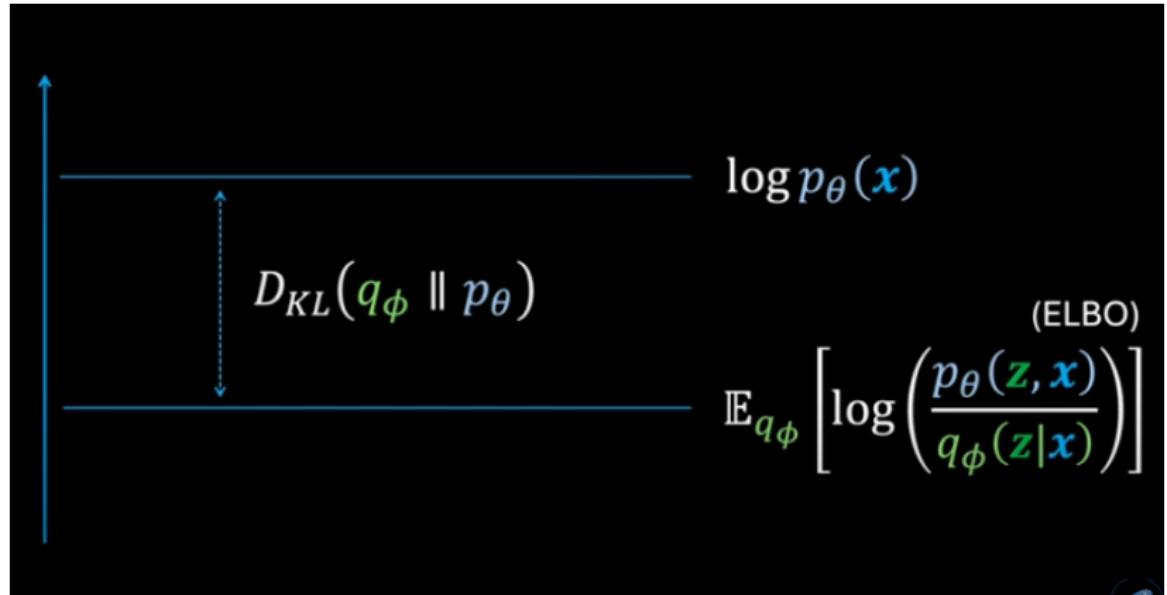


Figure: Image borrowed from

<https://www.youtube.com/watch?v=HBYQvKlaE0A>.

Maximizing ELBO maximizes  $p_\theta(\mathbf{x})$  and  $D_{KL}(q||p)$ !

# ELBO Optimization

Stochastic gradient descent on ELBO... but lots of work...

## **Reparameterization Trick (in practice)**

- ▶  $q_\phi(\mathbf{z}|\mathbf{x}) = g(\phi, \mathbf{x}, \epsilon)$
- ▶  $\epsilon \sim p(\epsilon)$

Now expectation is taken over  $p$

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{p(\epsilon)}[\log(p_\theta(\mathbf{z}, \mathbf{x}) + \log(q_\theta(\mathbf{z})))]$$

*Now ... gradient of expectation is expectation of gradient.* And can be estimated using Montecarlo sampling.

$$\text{ELBO} = \sum_i \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}))$$

# Simplifying Assumptions

From before ...

$$\text{ELBO} = \sum_i \log p_{\theta}(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}(q_{\phi}(\mathbf{z}|x) || p_{\theta}(\mathbf{z}))$$

# Simplifying Assumptions

From before ...

$$\text{ELBO} = \sum_i \log p_{\theta}(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}(q_{\phi}(\mathbf{z}|x) || p_{\theta}(\mathbf{z}))$$

## Gaussian Assumptions

- ▶ **Prior:**  $p(z) = \mathcal{N}(0, I)$
- ▶ **Posterior:**  $q_{\phi}(z|x) = \mathcal{N}(\mu, \sigma)$
- ▶  $\mu, \sigma = f(\mathbf{x})$

# Simplifying Assumptions

From before ...

$$\text{ELBO} = \sum_i \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}(q_\phi(\mathbf{z}|x) || p_\theta(\mathbf{z}))$$

## Gaussian Assumptions

- ▶ **Prior:**  $p(z) = \mathcal{N}(0, I)$
- ▶ **Posterior:**  $q_\phi(z|x) = \mathcal{N}(\mu, \sigma)$
- ▶  $\mu, \sigma = f(\mathbf{x})$

The loss becomes

$$D_{\text{KL}}(q_\phi(\mathbf{z}|x) || p_\theta(\mathbf{z})) = \frac{-1}{2} [\log \sigma^2 - \mu^2 - \sigma^2 + 1].$$

# Simplifying Assumptions

From before ...

$$\text{ELBO} = \sum_i \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}(q_\phi(\mathbf{z}|x) || p_\theta(\mathbf{z}))$$

## Gaussian Assumptions

- ▶ **Prior:**  $p(z) = \mathcal{N}(0, I)$
- ▶ **Posterior:**  $q_\phi(z|x) = \mathcal{N}(\mu, \sigma)$
- ▶  $\mu, \sigma = f(\mathbf{x})$

The loss becomes

$$D_{\text{KL}}(q_\phi(\mathbf{z}|x) || p_\theta(\mathbf{z})) = \frac{-1}{2} [\log \sigma^2 - \mu^2 - \sigma^2 + 1].$$

**Modeling**  $\log p_\theta(\mathbf{x}|\mathbf{z})$

# Simplifying Assumptions

From before ...

$$\text{ELBO} = \sum_i \log p_\theta(\mathbf{x}|\mathbf{z}) - \text{D}_{\text{KL}}(q_\phi(\mathbf{z}|x) || p_\theta(\mathbf{z}))$$

## Gaussian Assumptions

- ▶ **Prior:**  $p(z) = \mathcal{N}(0, I)$
- ▶ **Posterior:**  $q_\phi(z|x) = \mathcal{N}(\mu, \sigma)$
- ▶  $\mu, \sigma = f(\mathbf{x})$

The loss becomes

$$\text{D}_{\text{KL}}(q_\phi(\mathbf{z}|x) || p_\theta(\mathbf{z})) = \frac{-1}{2} [\log \sigma^2 - \mu^2 - \sigma^2 + 1].$$

## Modeling $\log p_\theta(\mathbf{x}|\mathbf{z})$

- ▶ Gaussian assumption:  $(\mathbf{x} - \hat{\mathbf{x}})^2$
- ▶ Bernoulli assumption:  $\text{BCE}(\hat{\mathbf{x}}, \mathbf{x})$

# VAE Architecture

# VAE Architecture

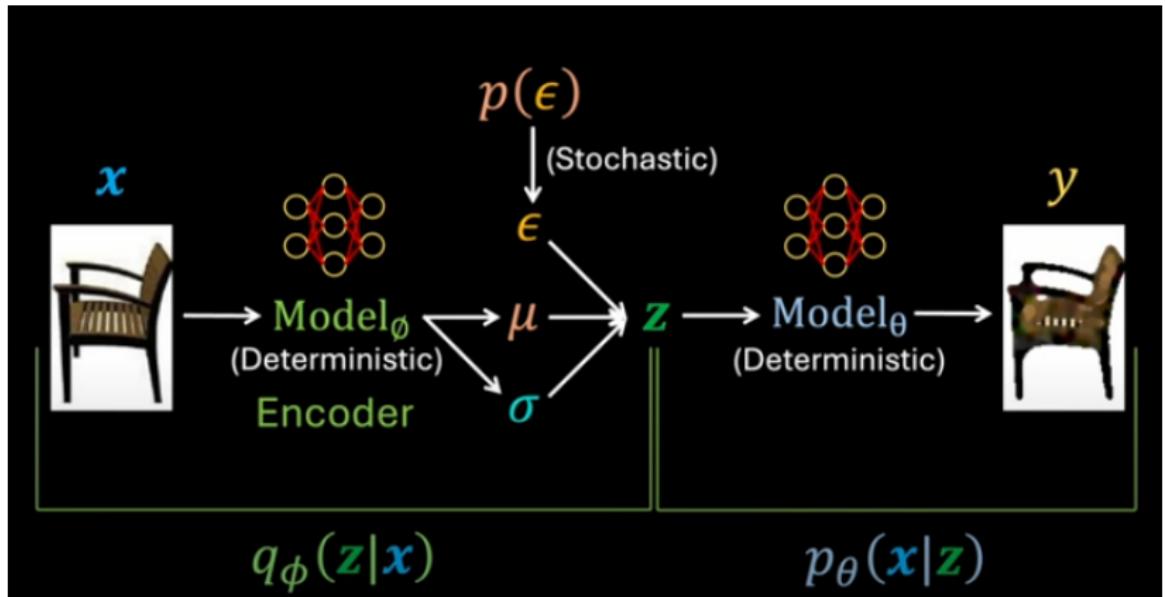


Figure: Image borrowed from

<https://www.youtube.com/watch?v=HBYQvKlaE0A>.

# Convolutional Autoencoder

# Continuous Convolution

# Continuous Convolution

Consider functions:  $f, g : \mathbb{R} \rightarrow \mathbb{R}$ .

$$f * g(x) = \int_u f(u)g(x - u)du$$

# Continuous Convolution

Consider functions:  $f, g : \mathbb{R} \rightarrow \mathbb{R}$ .

$$f * g(x) = \int_u f(u)g(x - u)du$$

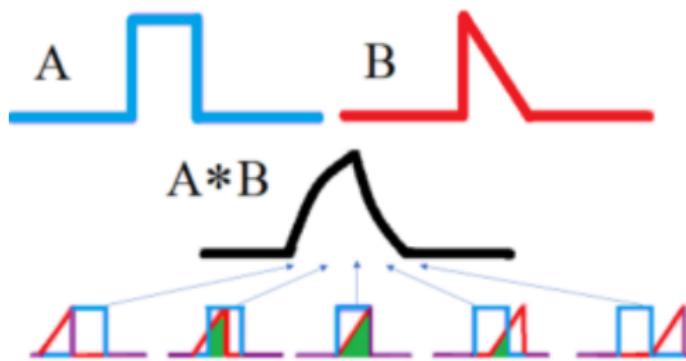


Figure:

<https://www.statisticshowto.com/convolution-integral-simple-definition/>

# Discrete Convolution

# Discrete Convolution

Consider functions:  $f, g : \mathbb{Z} \rightarrow \mathbb{Z}$ .

$$f * g(n) = \sum_m f(m)g(n - m)$$

# Discrete Convolution

Consider functions:  $f, g : \mathbb{Z} \rightarrow \mathbb{Z}$ .

$$f * g(n) = \sum_m f(m)g(n - m)$$

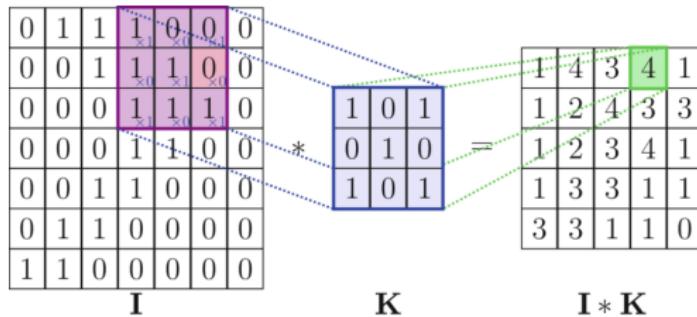


Figure: <https://www.researchgate.net/figure/>

A-visualization-of-the-discrete-convolution-operation-in-2D\_fig5\_372609466

# Convolutional Filter

Sliding convolutional filter.

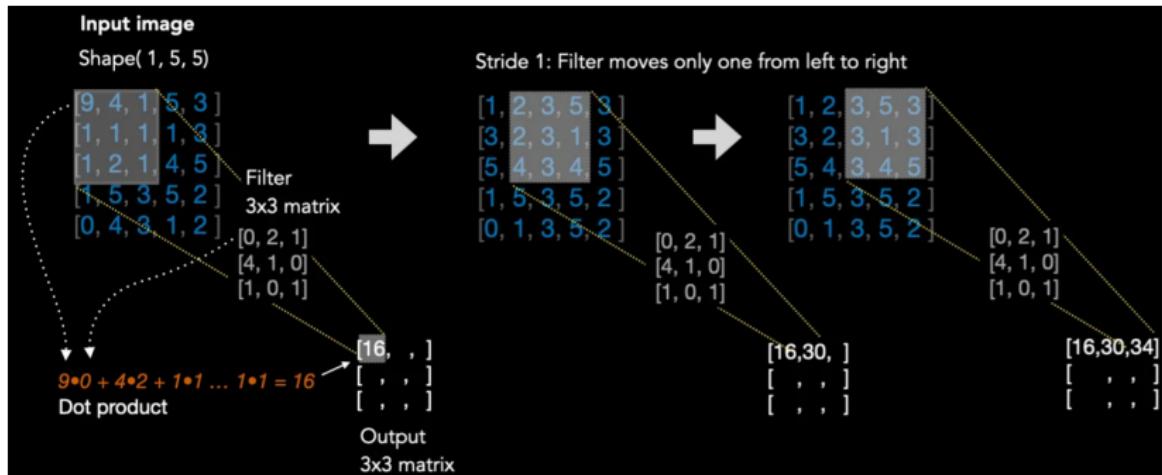


Figure:

<https://medium.com/advanced-deep-learning/cnn-operation-with-2-kernels-resulting-in-2-feature-mapsunderstanding-the-convolutional-filter-c4aad26cf32>

# Edge Detectors

Certain filters extract different patterns.

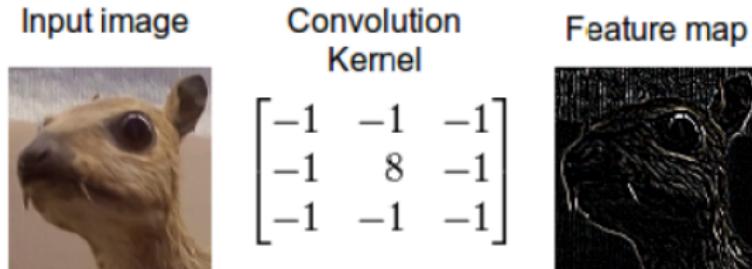


Figure: <https://timdettmers.com/2015/03/26/convolution-deep-learning/>

# Convolutional Autoencoders

Encoder and decoder are convolutional filters

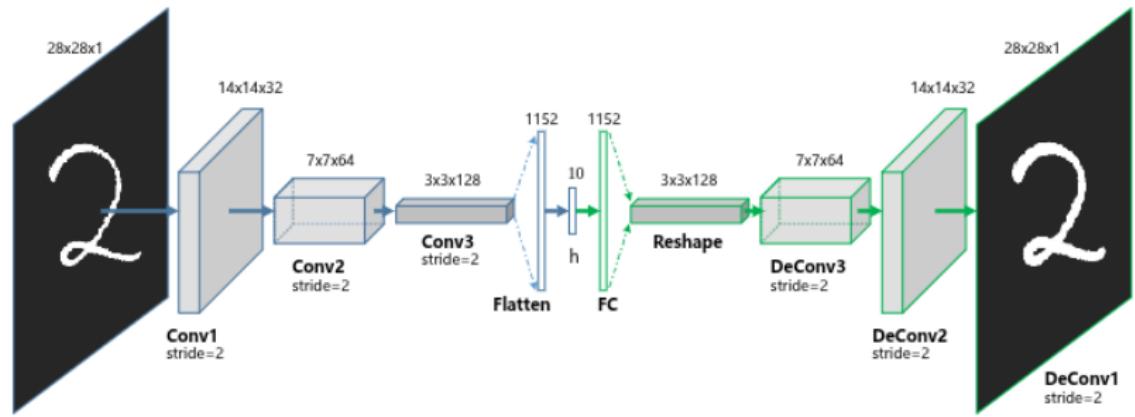


Figure: [Guo et al., 2017]

# Graph Convolutional Autoencoder

# Convolutions and Fourier Transform

# Convolutions and Fourier Transform

Let  $\mathcal{F}$  be the Fourier transform. Consider functions:  $f, g : \mathbb{R} \rightarrow \mathbb{R}$ . Then, under certain assumptions

$$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$$

# Convolutions and Fourier Transform

Let  $\mathcal{F}$  be the Fourier transform. Consider functions:  $f, g : \mathbb{R} \rightarrow \mathbb{R}$ . Then, under certain assumptions

$$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$$

- ▶ Multiplication is easy!
- ▶ Decreases the computational cost of convolution

# Discrete Fourier Transform (DFT)

# Discrete Fourier Transform (DFT)

Let  $\omega := e^{-2\pi i/N} = \cos(-2\pi/N) + i \sin(-2\pi/N)$

# Discrete Fourier Transform (DFT)

Let  $\omega := e^{-2\pi i/N} = \cos(-2\pi/N) + i\sin(-2\pi/N)$

$$\mathbf{V} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{d-1} \\ 1 & \omega^2 & \dots & \omega^{2(d-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{p-1} & \dots & \omega^{(d-1)(d-1)} \end{bmatrix}.$$

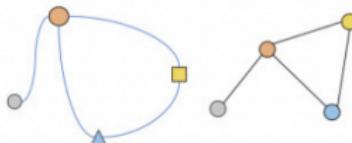
# Discrete Fourier Transform (DFT)

Let  $\omega := e^{-2\pi i / N} = \cos(-2\pi / N) + i \sin(-2\pi / N)$

$$\mathbf{V} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{d-1} \\ 1 & \omega^2 & \cdots & \omega^{2(d-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{p-1} & \cdots & \omega^{(d-1)(d-1)} \end{bmatrix}.$$

- ▶ The DFT of a timeseries  $\mathbf{x} = [x(1)|x(2)|\cdots|x(d)]^\top$  is  $\mathbf{Vx}$
- ▶ Transforms from time to frequency domain
- ▶ Can also be used on other signals (features other than time)

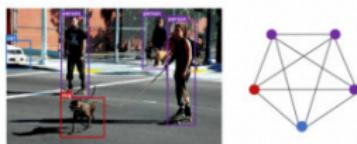
# Graph Data



(a) Physics



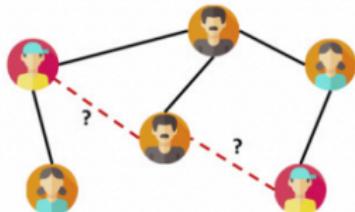
(b) Molecule



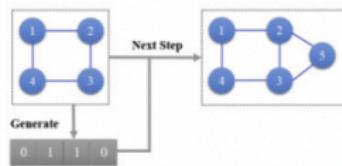
(c) Image



(d) Text



(e) Social Network

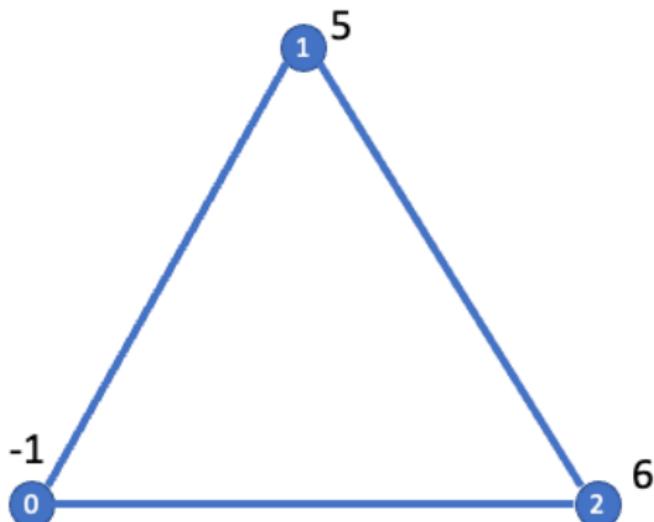


(f) Generation

Figure: [Zhou et al., 2020]

# Graph Signals

A graph signal is a vector with a value at every node on the graph.



# Discrete Graph Fourier Transform (GFT)

1. **A** adjacency matrix for the graph

# Discrete Graph Fourier Transform (GFT)

1.  $\mathbf{A}$  adjacency matrix for the graph
2.  $\mathbf{x}$  signal on graph

# Discrete Graph Fourier Transform (GFT)

1. **A** adjacency matrix for the graph
2. **x** signal on graph
3. **D** degree matrix

# Discrete Graph Fourier Transform (GFT)

1.  $\mathbf{A}$  adjacency matrix for the graph
2.  $\mathbf{x}$  signal on graph
3.  $\mathbf{D}$  degree matrix
4.  $\mathbf{L} = \mathbf{D} - \mathbf{A}$  graph Laplacian

# Discrete Graph Fourier Transform (GFT)

1.  $\mathbf{A}$  adjacency matrix for the graph
2.  $\mathbf{x}$  signal on graph
3.  $\mathbf{D}$  degree matrix
4.  $\mathbf{L} = \mathbf{D} - \mathbf{A}$  graph Laplacian
5.  $\mathbf{L}\mathbf{U} = \mathbf{U}\Lambda$  eigenvalue decomposition

# Discrete Graph Fourier Transform (GFT)

1.  $\mathbf{A}$  adjacency matrix for the graph
2.  $\mathbf{x}$  signal on graph
3.  $\mathbf{D}$  degree matrix
4.  $\mathbf{L} = \mathbf{D} - \mathbf{A}$  graph Laplacian
5.  $\mathbf{L}\mathbf{U} = \mathbf{U}\Lambda$  eigenvalue decomposition
6.  $\mathcal{F}(\mathbf{x}) = \mathbf{U}^\top \mathbf{x}$  discrete graph Fourier transform

# Discrete Graph Fourier Transform (GFT)

1.  $\mathbf{A}$  adjacency matrix for the graph
2.  $\mathbf{x}$  signal on graph
3.  $\mathbf{D}$  degree matrix
4.  $\mathbf{L} = \mathbf{D} - \mathbf{A}$  graph Laplacian
5.  $\mathbf{L}\mathbf{U} = \mathbf{U}\Lambda$  eigenvalue decomposition
6.  $\mathcal{F}(\mathbf{x}) = \mathbf{U}^\top \mathbf{x}$  discrete graph Fourier transform
7.  $\mathcal{F}^{-1}(\mathbf{z}) = \mathbf{U}\mathbf{z}$  inverse discrete graph Fourier transform

# Discrete Graph Fourier Transform (GFT)

1.  $\mathbf{A}$  adjacency matrix for the graph
2.  $\mathbf{x}$  signal on graph
3.  $\mathbf{D}$  degree matrix
4.  $\mathbf{L} = \mathbf{D} - \mathbf{A}$  graph Laplacian
5.  $\mathbf{LU} = \mathbf{U}\Lambda$  eigenvalue decomposition
6.  $\mathcal{F}(\mathbf{x}) = \mathbf{U}^\top \mathbf{x}$  discrete graph Fourier transform
7.  $\mathcal{F}^{-1}(\mathbf{z}) = \mathbf{U}\mathbf{z}$  inverse discrete graph Fourier transform

***On the cycle graph, the GFT reduces to the DFT.***

# Graph Convolution

Graph signals  $\mathbf{x}, \mathbf{g}$

# Graph Convolution

Graph signals  $\mathbf{x}, \mathbf{g}$

$$\mathcal{F}(\mathbf{x} * \mathbf{g}) = \mathcal{F}(\mathbf{x}) \cdot \mathcal{F}(\mathbf{g}) = \Lambda_{\mathbf{U}^\top \mathbf{g}} \mathbf{U}^\top \mathbf{x}$$

Where  $\cdot$  is point-wise multiplication and

$$\Lambda_g = \begin{bmatrix} (\mathbf{U}^\top \mathbf{g})_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & (\mathbf{U}^\top \mathbf{g})_d \end{bmatrix}$$

# Graph Convolution

Graph signals  $\mathbf{x}, \mathbf{g}$

$$\mathcal{F}(\mathbf{x} * \mathbf{g}) = \mathcal{F}(\mathbf{x}) \cdot \mathcal{F}(\mathbf{g}) = \mathbf{\Lambda}_{\mathbf{U}^\top \mathbf{g}} \mathbf{U}^\top \mathbf{x}$$

Where  $\cdot$  is point-wise multiplication and

$$\mathbf{\Lambda}_g = \begin{bmatrix} (\mathbf{U}^\top \mathbf{g})_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & (\mathbf{U}^\top \mathbf{g})_d \end{bmatrix}$$

Graph convolution

$$\mathbf{x} * \mathbf{g} = \mathcal{F}^{-1} \circ \mathcal{F}(\mathbf{x} * \mathbf{g}) = \mathbf{U} \mathbf{\Lambda}_{\mathbf{U}^\top \mathbf{g}} \mathbf{U}^\top \mathbf{x}$$

## Graph Convolutional Layer [Kipf and Welling, 2016a]

## Graph Convolutional Layer [Kipf and Welling, 2016a]

Replace graph signal  $\mathbf{g}$  with signal in frequency domain as a function of  $\Lambda$ :  $\mathbf{g}_\theta(\Lambda)$ .

## Graph Convolutional Layer [Kipf and Welling, 2016a]

Replace graph signal  $\mathbf{g}$  with signal in frequency domain as a function of  $\Lambda$ :  $\mathbf{g}_\theta(\Lambda)$ . Convolution becomes

$$\mathbf{U}\mathbf{g}_\theta(\Lambda)\mathbf{U}^\top \mathbf{x}$$

## Graph Convolutional Layer [Kipf and Welling, 2016a]

Replace graph signal  $\mathbf{g}$  with signal in frequency domain as a function of  $\Lambda$ :  $\mathbf{g}_\theta(\Lambda)$ . Convolution becomes

$$\mathbf{U}\mathbf{g}_\theta(\Lambda)\mathbf{U}^\top \mathbf{x}$$

Approximate using Chebyshev polynomials

$$\mathbf{U}\mathbf{g}_\theta(\Lambda)\mathbf{U}^\top \mathbf{x} \approx \theta_1 \mathbf{x} - \theta_2 \mathbf{L}\mathbf{x} \approx \theta(\mathbf{I} - \mathbf{L})\mathbf{x}$$

## Graph Convolutional Layer [Kipf and Welling, 2016a]

Replace graph signal  $\mathbf{g}$  with signal in frequency domain as a function of  $\Lambda$ :  $\mathbf{g}_\theta(\Lambda)$ . Convolution becomes

$$\mathbf{U}\mathbf{g}_\theta(\Lambda)\mathbf{U}^\top \mathbf{x}$$

Approximate using Chebyshev polynomials

$$\mathbf{U}\mathbf{g}_\theta(\Lambda)\mathbf{U}^\top \mathbf{x} \approx \theta_1 \mathbf{x} - \theta_2 \mathbf{L}\mathbf{x} \approx \theta(\mathbf{I} - \mathbf{L})\mathbf{x}$$

Renormalization trick & gives us the graph convolutional layer:

$$\text{GC}(\mathbf{X}) = \sigma(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{X} \Theta) \quad (14)$$

- ▶  $\tilde{\mathbf{A}} = \mathbf{I} + \mathbf{A}$
- ▶  $\tilde{\mathbf{D}}$  degree matrix
- ▶  $\mathbf{X}$  input matrix of nodes  $\times$  features
- ▶ Parameter matrix  $\Theta$

## Graph Convolutional Layer [Kipf and Welling, 2016a]

Replace graph signal  $\mathbf{g}$  with signal in frequency domain as a function of  $\Lambda$ :  $\mathbf{g}_\theta(\Lambda)$ . Convolution becomes

$$\mathbf{U}\mathbf{g}_\theta(\Lambda)\mathbf{U}^\top \mathbf{x}$$

Approximate using Chebyshev polynomials

$$\mathbf{U}\mathbf{g}_\theta(\Lambda)\mathbf{U}^\top \mathbf{x} \approx \theta_1 \mathbf{x} - \theta_2 \mathbf{L}\mathbf{x} \approx \theta(\mathbf{I} - \mathbf{L})\mathbf{x}$$

Renormalization trick & gives us the graph convolutional layer:

$$\text{GC}(\mathbf{X}) = \sigma(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{X} \Theta) \quad (14)$$

- ▶  $\tilde{\mathbf{A}} = \mathbf{I} + \mathbf{A}$
- ▶  $\tilde{\mathbf{D}}$  degree matrix
- ▶  $\mathbf{X}$  input matrix of nodes  $\times$  features
- ▶ Parameter matrix  $\Theta$

See Kipf and Welling [2016b] for details.

Graph (Variational) Autoencoders [Kipf and Welling, 2016b]

## Graph (Variational) Autoencoders [Kipf and Welling, 2016b]

Reconstructed adjacency matrix:  $\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^\top)$

# Graph (Variational) Autoencoders [Kipf and Welling, 2016b]

Reconstructed adjacency matrix:  $\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^\top)$

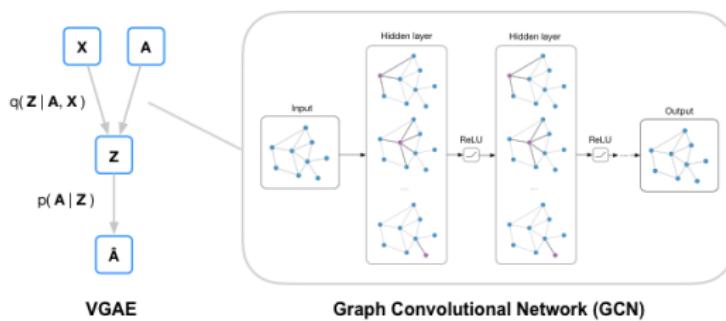


Figure: Code and image from <https://github.com/tkipf/gae>

# Graph (Variational) Autoencoders [Kipf and Welling, 2016b]

Reconstructed adjacency matrix:  $\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^\top)$

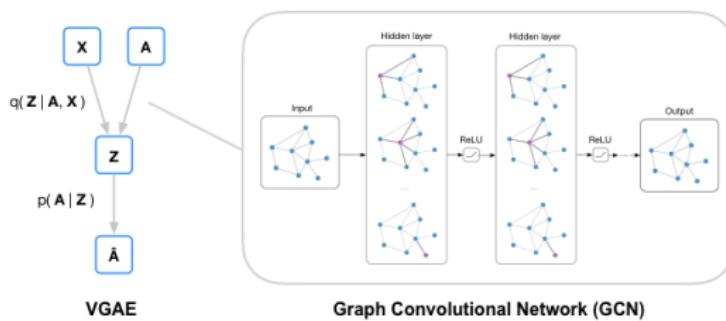
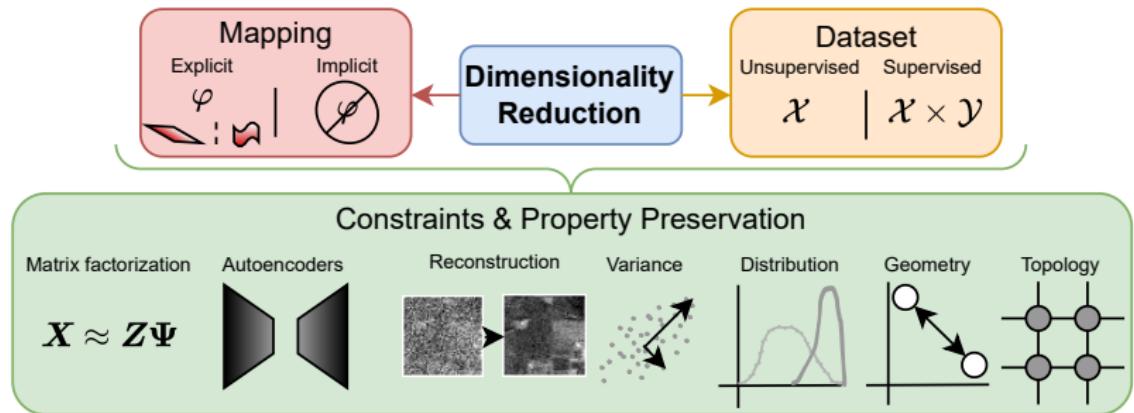


Figure: Code and image from <https://github.com/tkipf/gae>

Further reading: graph-based architectures [Wu et al., 2020] and generalizations of graph autoencoders [Hajij et al., 2021]

# Conclusions

# DR Summary



## Summary

Method	Map		Data		Constr. & Prop. Pres.						Code	
	Exp.	Lin.	Sup.		M.F.	A.E.	Rec.	Var.	Dist.	Geo.	Top	
PCA	✓	✓		✗	✓	✗	✓	✓	✗	✗	✗	sklearn
LDA	✓	✓		✓	✓	✗	✓	✓	✗	✗	✗	sklearn
VAE	✓	✗		✗	✓	✗	✗	✓	✗	✗	✗	github
MDS	✗	✗		✗	✗	✗	✗	✗	✗	✓	✗	sklearn
Isomap	✗	✗		✗	✗	✗	✗	✗	✗	✓	✗	sklearn
t-SNE	✗	✗		✗	✗	✗	✓	✓	✗	✓	✓	sklearn
UMAP	✗	✗		✗	✗	✗	✗	✗	✗	✗	✓	sklearn

## References I

- Travis Askham and J Nathan Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 17(1):380–416, 2018.
- Mukund Balasubramanian and Eric L Schwartz. The isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002.
- Xuchan Bao, James Lucas, Sushant Sachdeva, and Roger B Grosse. Regularized linear autoencoders recover the principal components, eventually. *Advances in Neural Information Processing Systems*, 33: 6971–6981, 2020.
- Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.
- Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.
- Matthew J Colbrook. The multiverse of dynamic mode decomposition algorithms. *arXiv preprint arXiv:2312.00137*, 2023.

## References II

- Scott TM Dawson, Maziar S Hemati, Matthew O Williams, and Clarence W Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57:1–19, 2016.
- P Thomas Fletcher, Conglin Lu, Stephen M Pizer, and Sarang Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE transactions on medical imaging*, 23(8):995–1005, 2004.
- Matan Gavish and David L Donoho. The optimal hard threshold for singular values is  $4/\sqrt{3}$ . *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014.
- Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14–18, 2017, Proceedings, Part II 24*, pages 373–382. Springer, 2017.

## References III

- Mustafa Hajij, Ghada Zamzmi, Theodore Papamarkou, Vasileios Maroulas, and Xuanting Cai. Simplicial complex representation learning. *arXiv preprint arXiv:2103.04046*, 2021.
- Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.
- Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.
- Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.
- Bernard O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5): 315–318, 1931.
- Joseph B Kruskal and Myron Wish. *Multidimensional scaling*. Number 11. Sage, 1978.
- John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

## References IV

- Gilad Lerman and Tyler Maunu. An overview of robust subspace recovery. *Proceedings of the IEEE*, 106(8):1380–1410, 2018.
- Nathan Mankovich, Gustau Camps-Valls, and Tolga Birdal. Fun with flags: Robust principal directions via flag manifolds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 330–340, 2024.
- Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
- Igor Mezić. Analysis of fluid flows via spectral properties of the koopman operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013.
- Xavier Pennec. Barycentric subspace analysis on manifolds. *The Annals of Statistics*, 2018.
- Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue PCA directions (by accident). In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12406–12415, 2019.

## References V

- Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009.
- Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.
- Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.
- Gilbert Strang. *Linear algebra and its applications*. 2000.
- Tom Szwagier and Xavier Pennec. The curse of isotropy: From principal components to principal subspaces. 2024. URL <https://arxiv.org/abs/2307.15348>.
- Tom Szwagier and Xavier Pennec. Nested subspace learning with flags. *arXiv preprint arXiv:2502.06022*, 2025.
- Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, and Aboul Ella Hassanien. Linear discriminant analysis: A detailed tutorial. *AI communications*, 30(2):169–190, 2017.

## References VI

- FS Tsai. Comparative study of dimensionality reduction techniques for data visualization. *Journal of artificial intelligence*, 3(3):119–134, 2010.
- Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.
- Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25: 1307–1346, 2015.
- Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

## References VII

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1: 57–81, 2020.