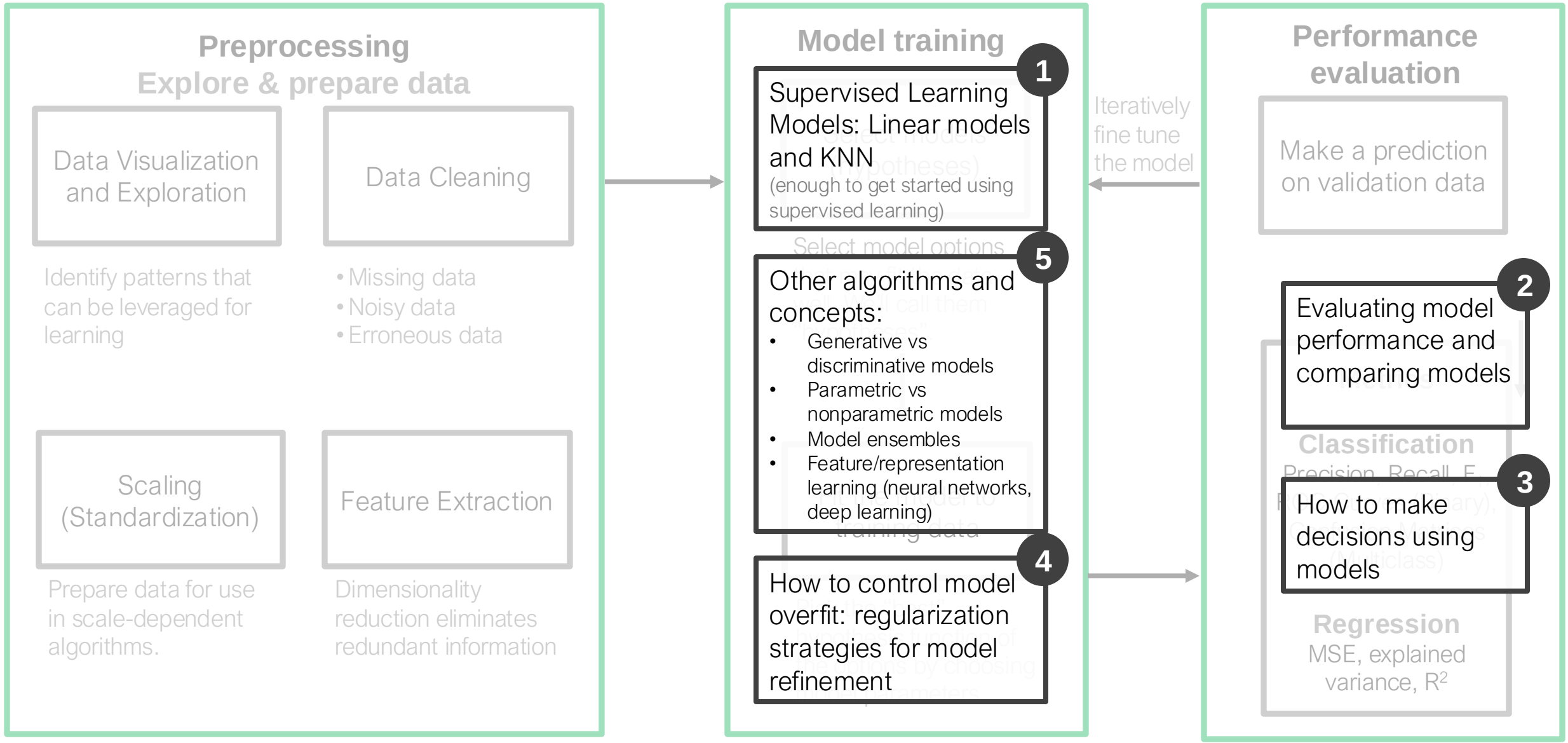
Unsupervised Learning

Supervised learning in practice



Types of machine learning

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Goal	**Predict** …from examples	**Describe** ...structure in data	**Strategize** learn by trial and error
Data	(x, y)	x	delayed feedback
Types	• Classification • Regression	• Density estimation • Clustering • Dimensionality reduction • Anomaly detection	• Model-free learning • Model-based learning

Unsupervised learning: describing data

1

Dimensionality Reduction

Developing new data representations

- Feature subset selection
- Feature projections
- Supervised approaches

2

Density Estimation

Quantifying data distributions

- Histograms
- Nonparametric density estimation
- Parametric models

3

Clustering

Grouping similar data

- Hierarchical
- Centroid-based
- Distribution-based
- Density-based

4

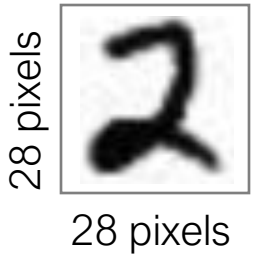
Other Unsupervised Learning Tools

- Anomaly detection
- Representation learning
- Generative models

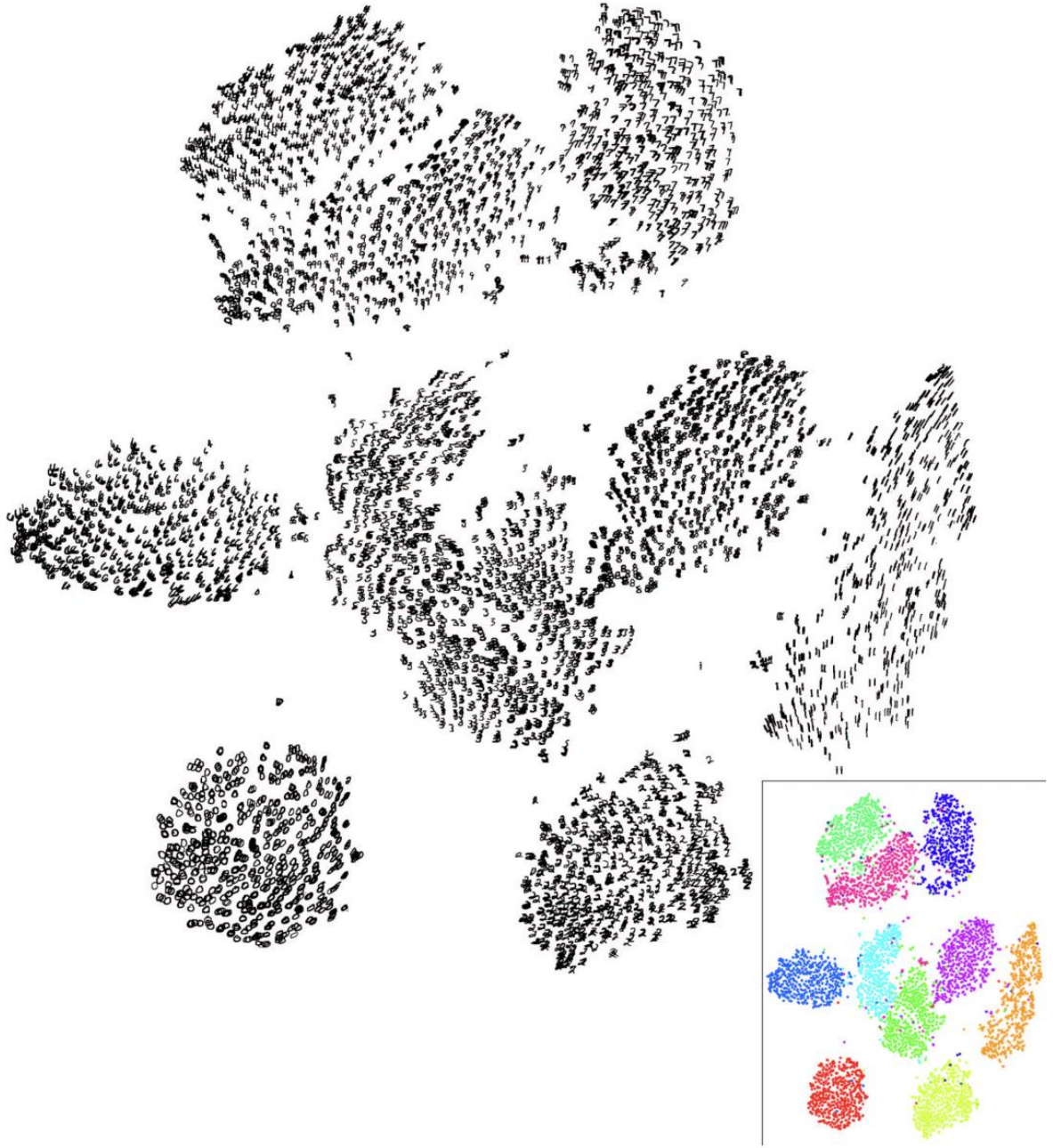
Dimensionality Reduction

Linear approaches (PCA)
Nonlinear approaches (UMAP) for visualization
Representation learning (Autoencoders) for developing embeddings

Reducing MNIST dimensionality into 2 dimensions



Dimensions = 784



Latent/Embedding Space Demo

This links to the lower dimensional representation space of the data (a.k.a. the latent space or embedding space) for PCA-reduced features with MNIST data

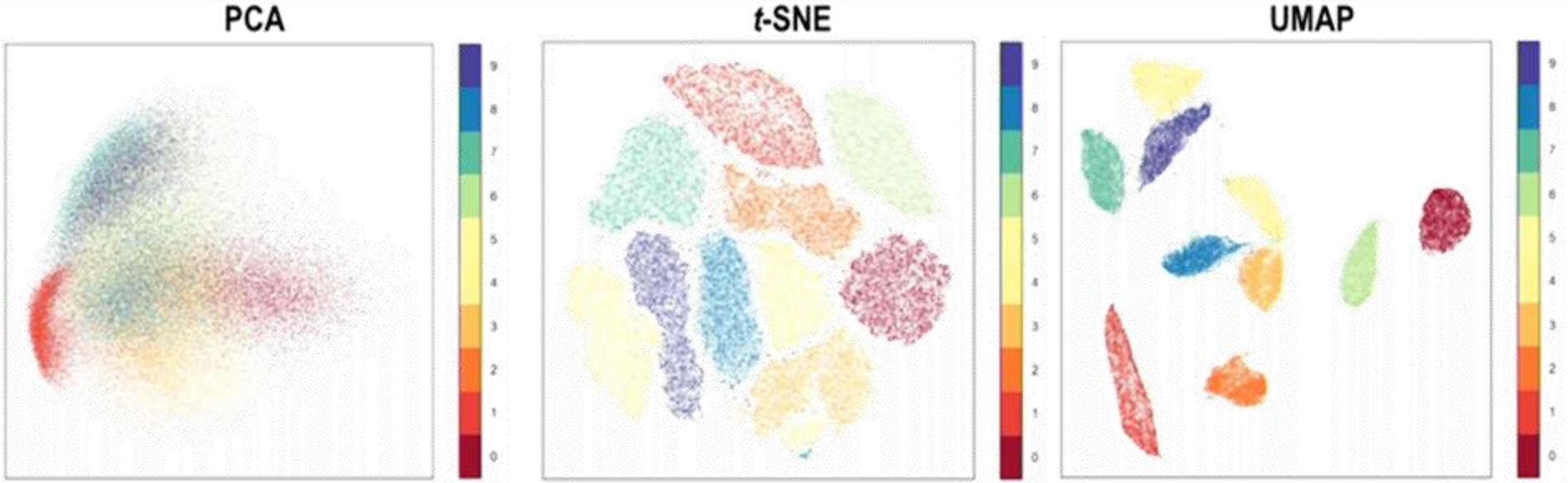
Van der Maaten, L. and Hinton, G., 2008. Visualizing data using t-SNE. *Journal of machine learning research*, *9*(11).

Van der Maaten, L. and Hinton, G., 2008. Visualizing data using t-SNE. *Journal of machine learning research*, *9*(11).

Van der Maaten, L. and Hinton, G., 2008. Visualizing data using t-SNE. *Journal of machine learning research*, *9*(11).

MNIST Dataset reduced to 2 dimensions

Original dimensions:
(28x28 = 784)



Fashion-MNIST Dataset reduced to 2 dimensions

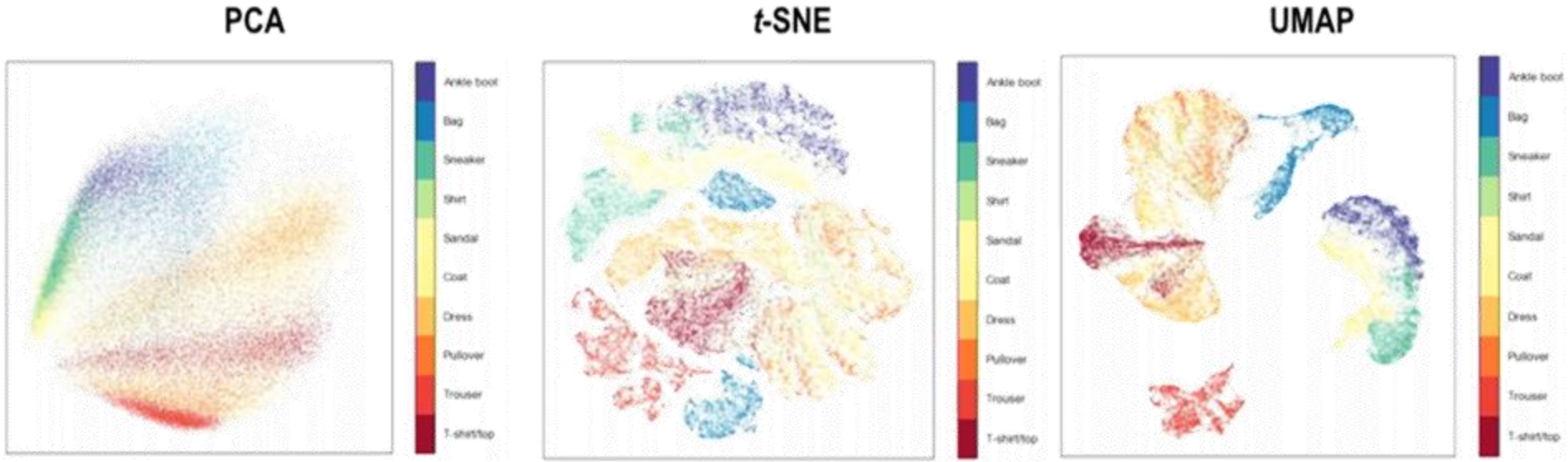


Image from Aizan Fahri: https://meta.caspershire.net/umap/

The Curse of Dimensionality

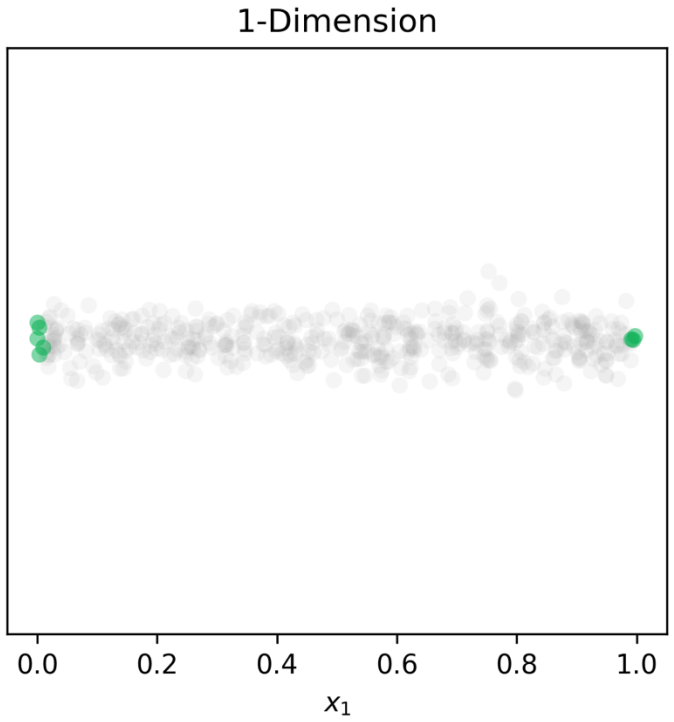
High dimensions = lots of features

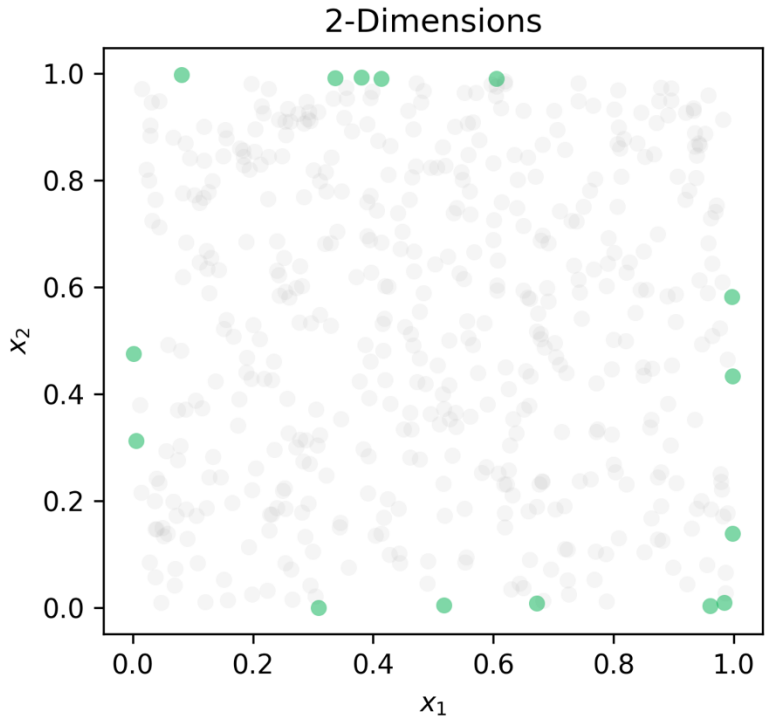
Challenge 1

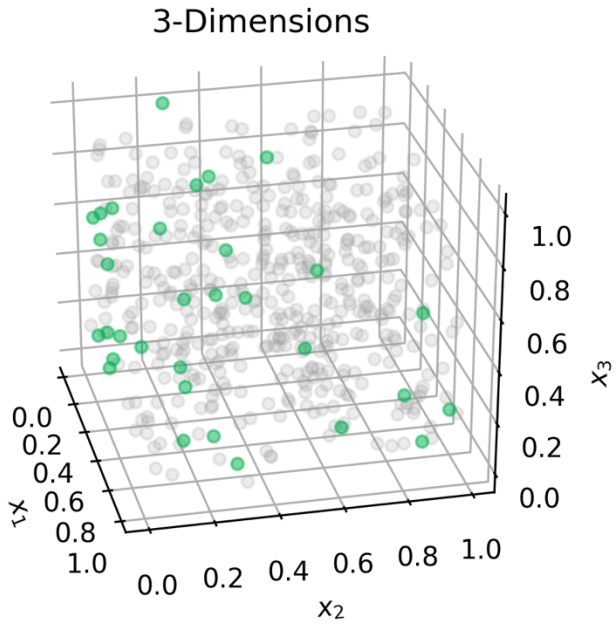
In high dimensions, data become sparse
(increasing the risk of overfitting)

Random data points in a unit hypercube…

- Data point is a distance < 0.01 units from the edge of a unit hypercube
- All other data







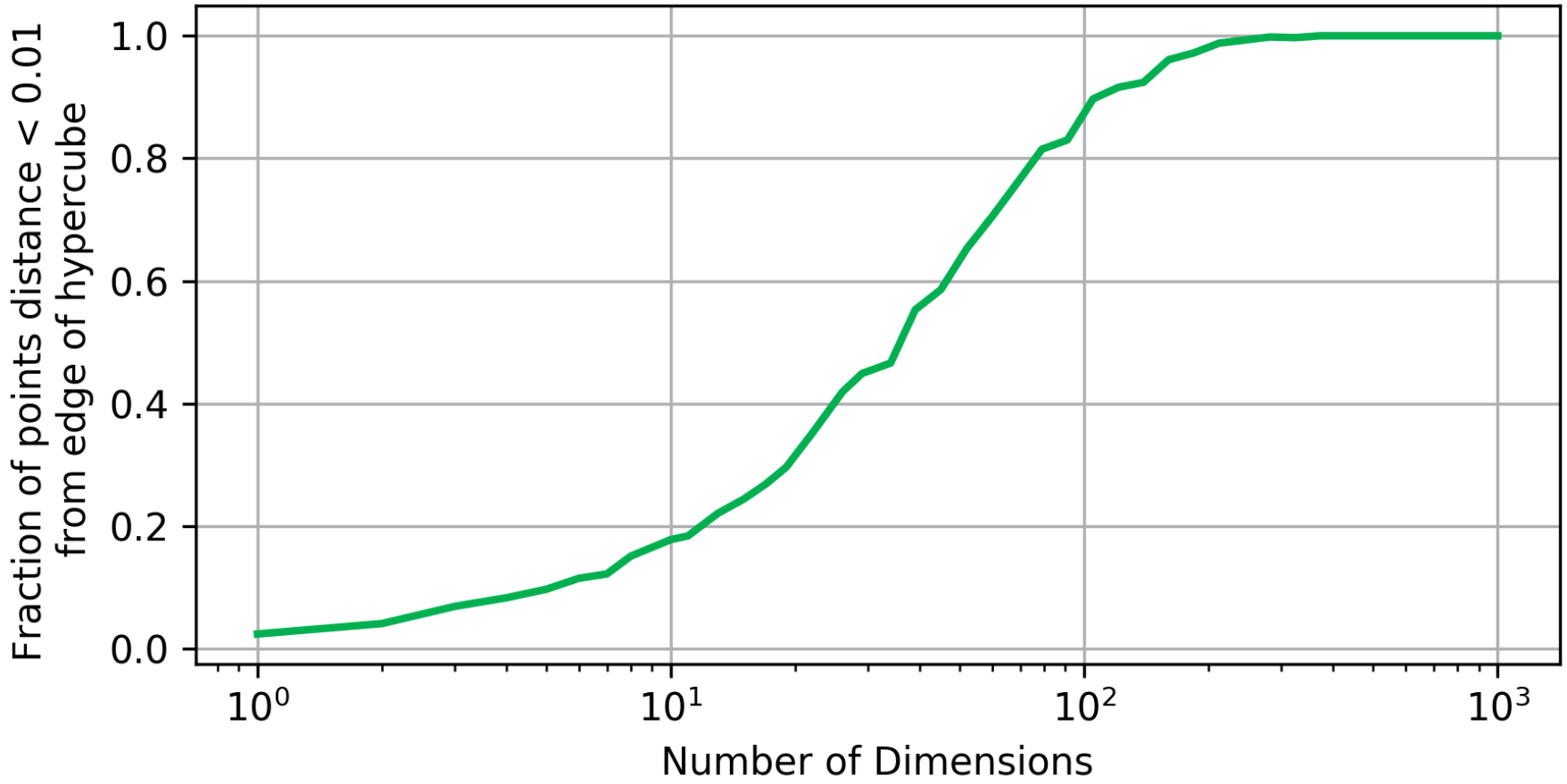
Fraction of edge data = $\frac{\text{●}}{\text{●} + \text{●}}$

0.016

0.030

0.064

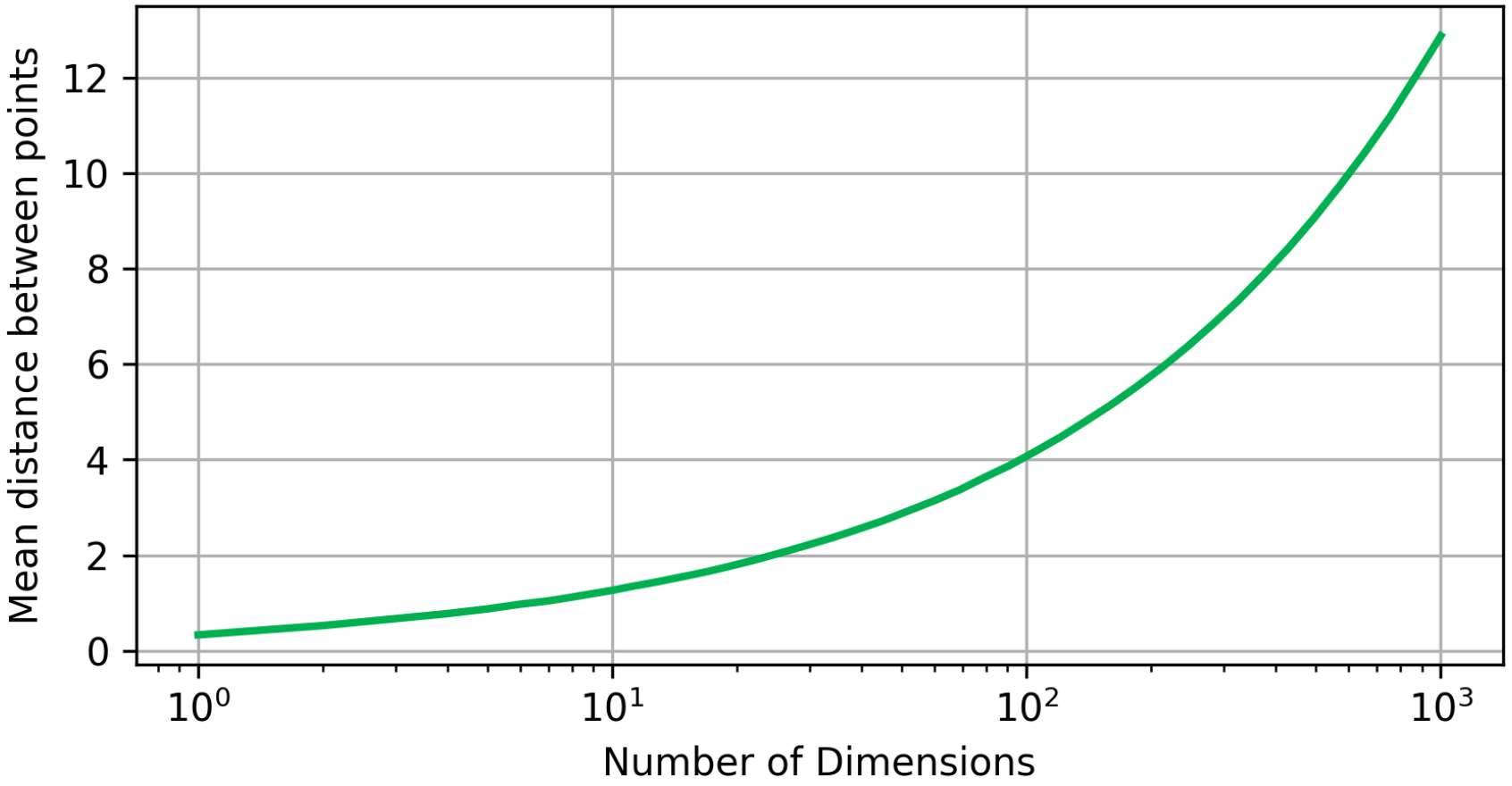
In high dimensions…



…nearly all of the high dimensional space is **far away from the center**

Note: figures constructed using 1,000 random points

In high dimensions…



…data become sparse

Note: figures constructed using 1,000 random points

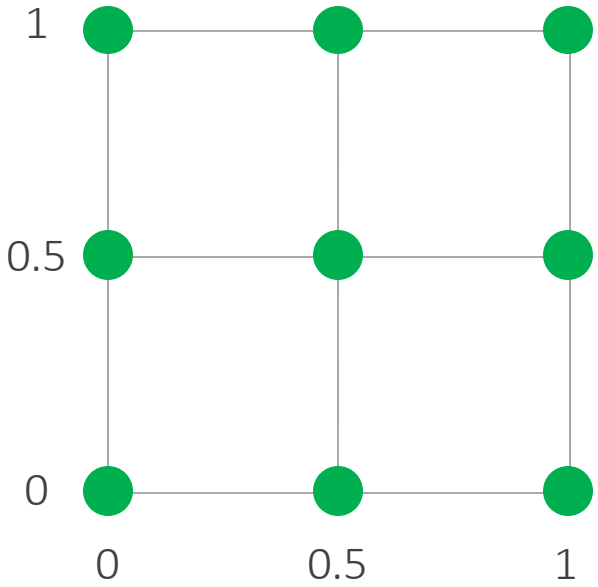
Much more data are needed for sampling higher dimensional spaces

Sample a unit hypercube on a grid spaced at intervals of 0.5

D = 1

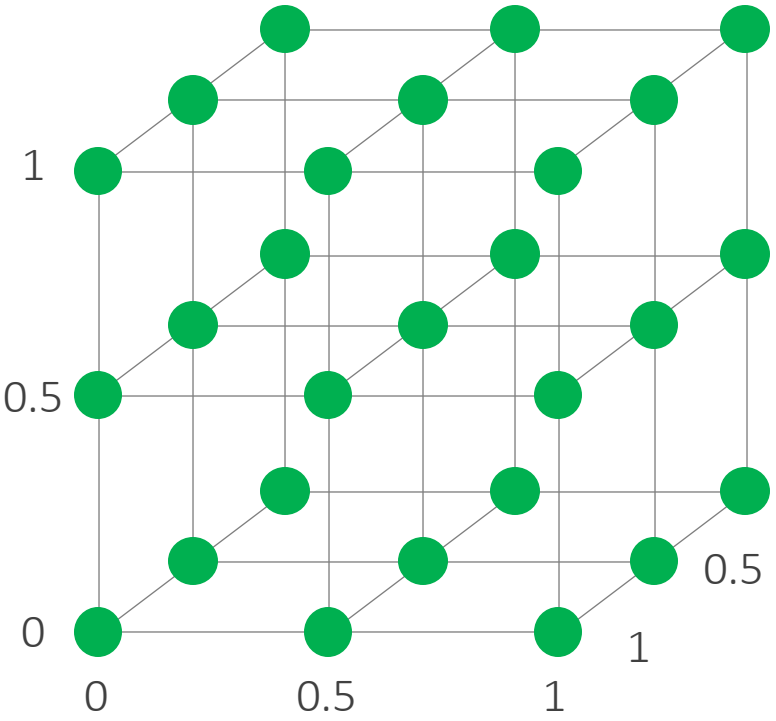
N = 3

D = 2



N = 9

D = 3



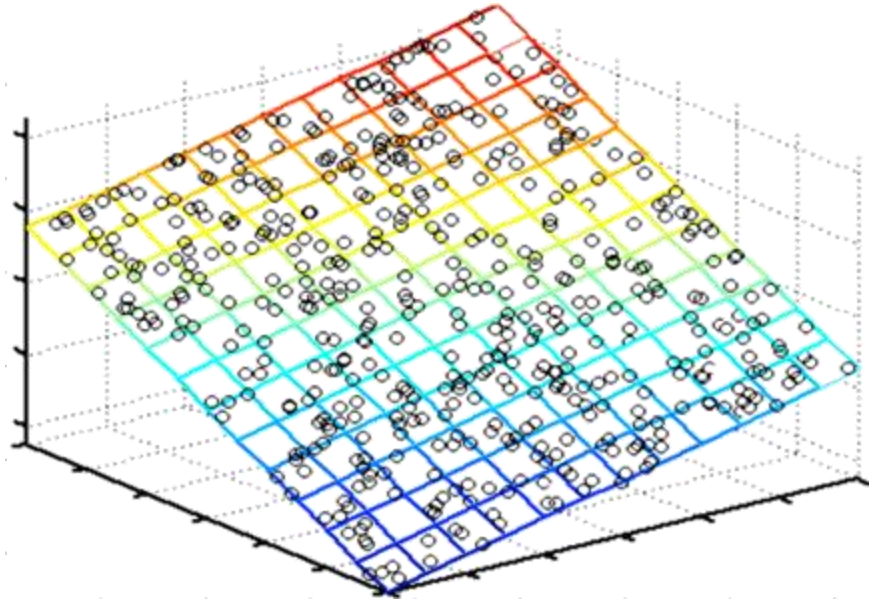
N = 27

Dim (D)	Samples (N)
4	81
5	243
10	59,049
100	5.2×10^{47}
300	**1.4×10^{143}**

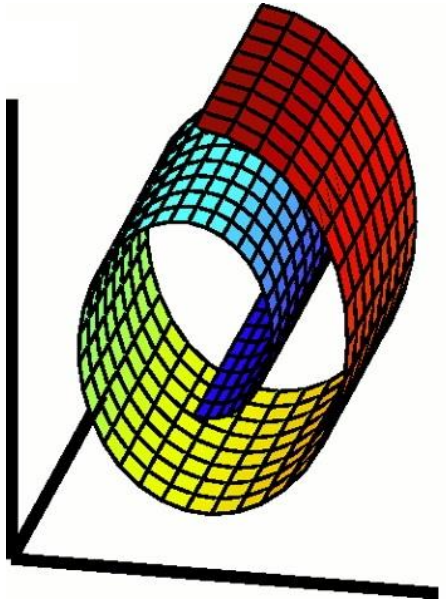
1 million Googles, each with 20 exabytes of data would only be 10^{25} bytes

…it takes more data to learn in high dimensional spaces

Data may lie in lower dimensional subspaces



Linear structure between features



Nonlinear structure between features

Often features are related to one another (are combinations of other features)

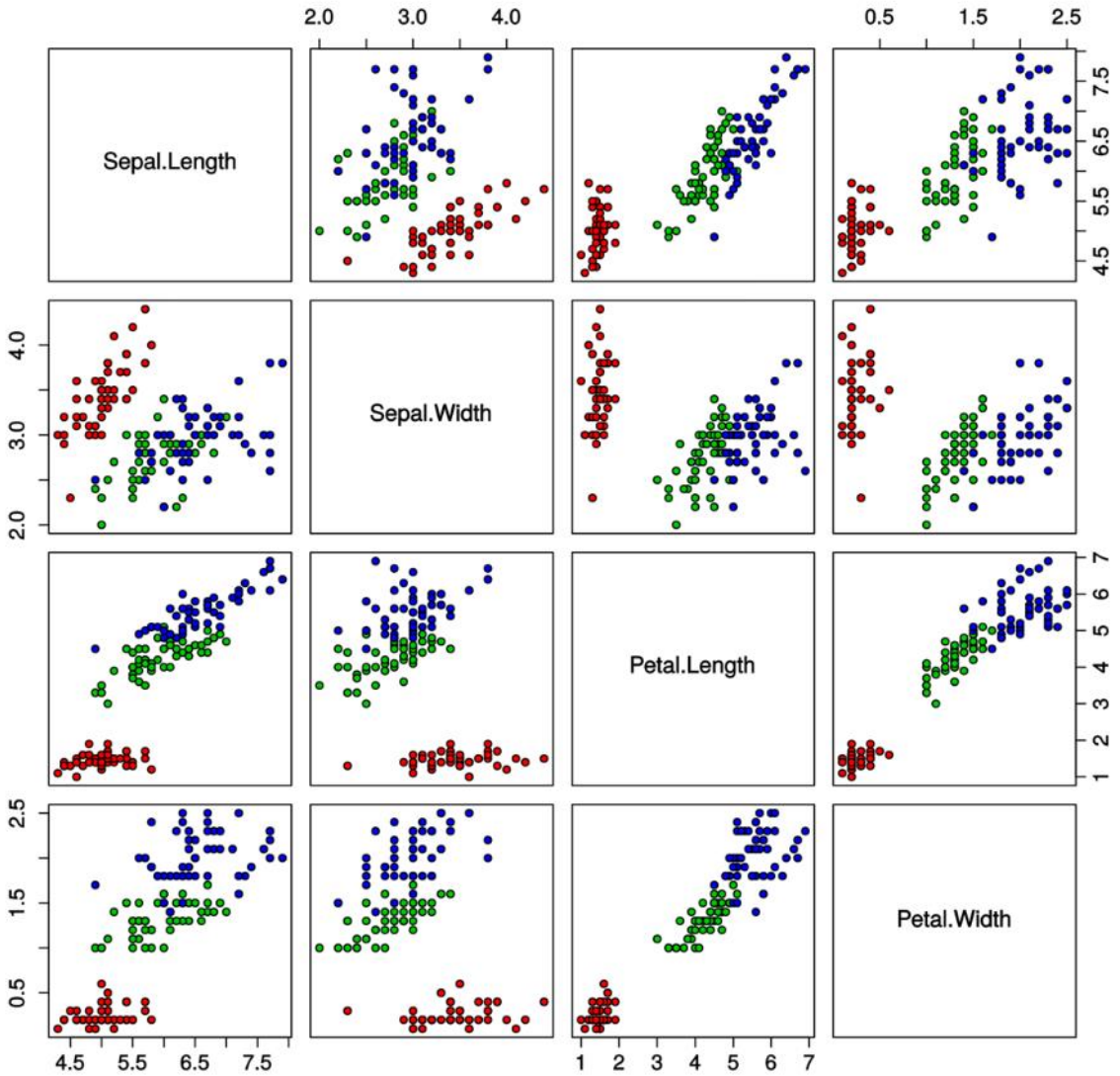
High dimensional data often exist in a lower dimensional subspace

Image Left: Torki, M. and June, Dissertation, 2011. Learning the manifolds of local features and their spatial arrangements. Rutgers University.
Image Right: Roweis, S.T. and Saul, L.K., 2000. Nonlinear dimensionality reduction by locally linear embedding. science, 290(5500), pp.2323-2326.

Challenge 2

Visualization of data becomes challenging in higher dimensions

Scatterplot Matrix



Parallel Coordinates Plot

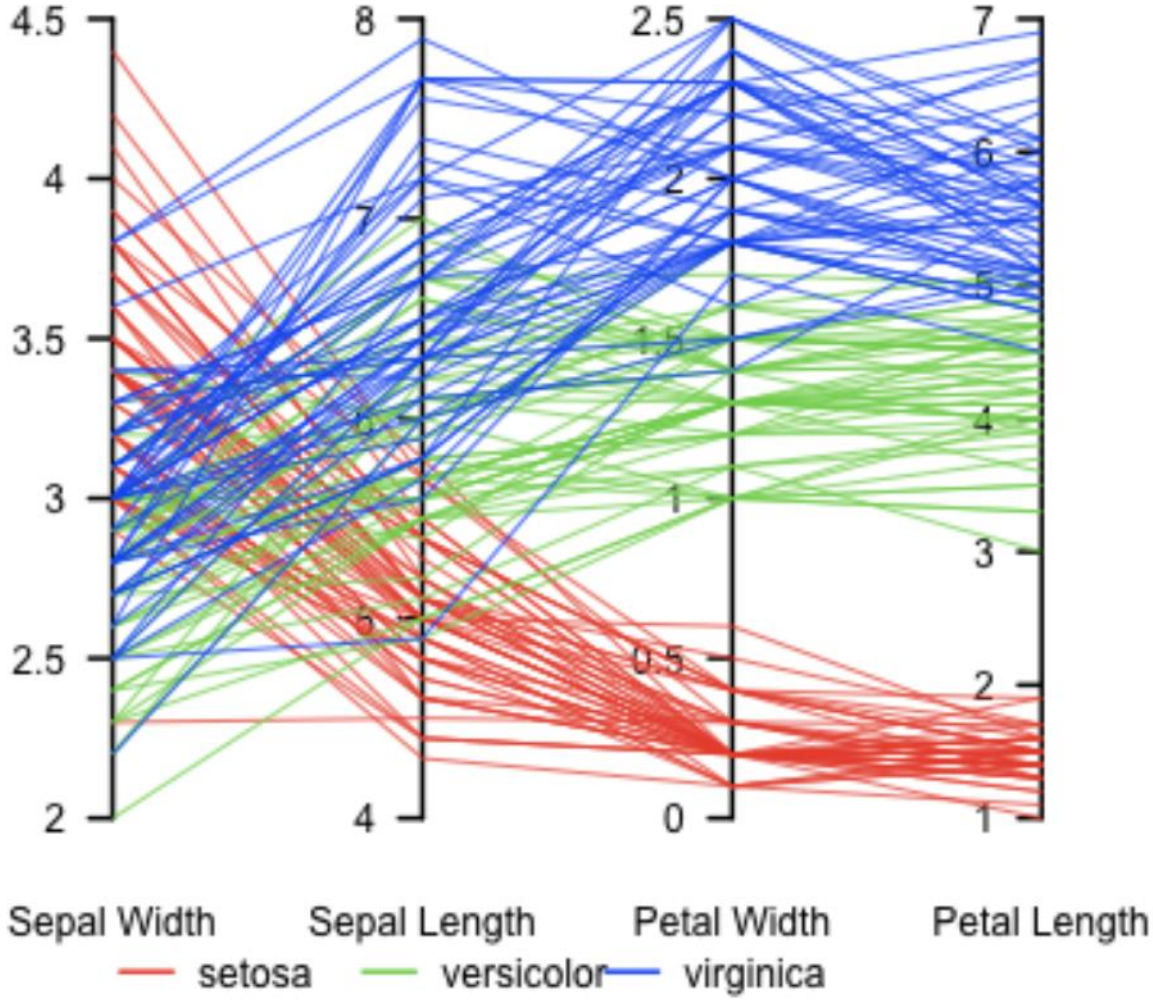


Image: Pezzotti, N., 2019. Dimensionality-Reduction Algorithms for Progressive Visual Analytics.

Dimensionality Reduction

Benefits:

Simplified computation

Data are easier to visualize

Reduced redundancy of features

Drawbacks:

Results in data/information loss
(and therefore accuracy is often reduced)

If used as preprocessing, incorrect application may decrease performance

Feature Subset Selection

- Exhaustive search
- Forward selection
- Backwards selection
- Simulated annealing
- Genetic algorithms
- Particle swarm optimization

Feature projection

- **Principal Components Analysis (PCA)**
- Autoencoder
- t-distributed Stochastic Neighbor Embedding (t-SNE)
- Uniform Manifold Approximation and Projection (UMAP)
- Non-negative Matrix Factorization (NMF)
- Multidimensional scaling (MDS)
- Random projections
- Isomap
- Local linear embeddings

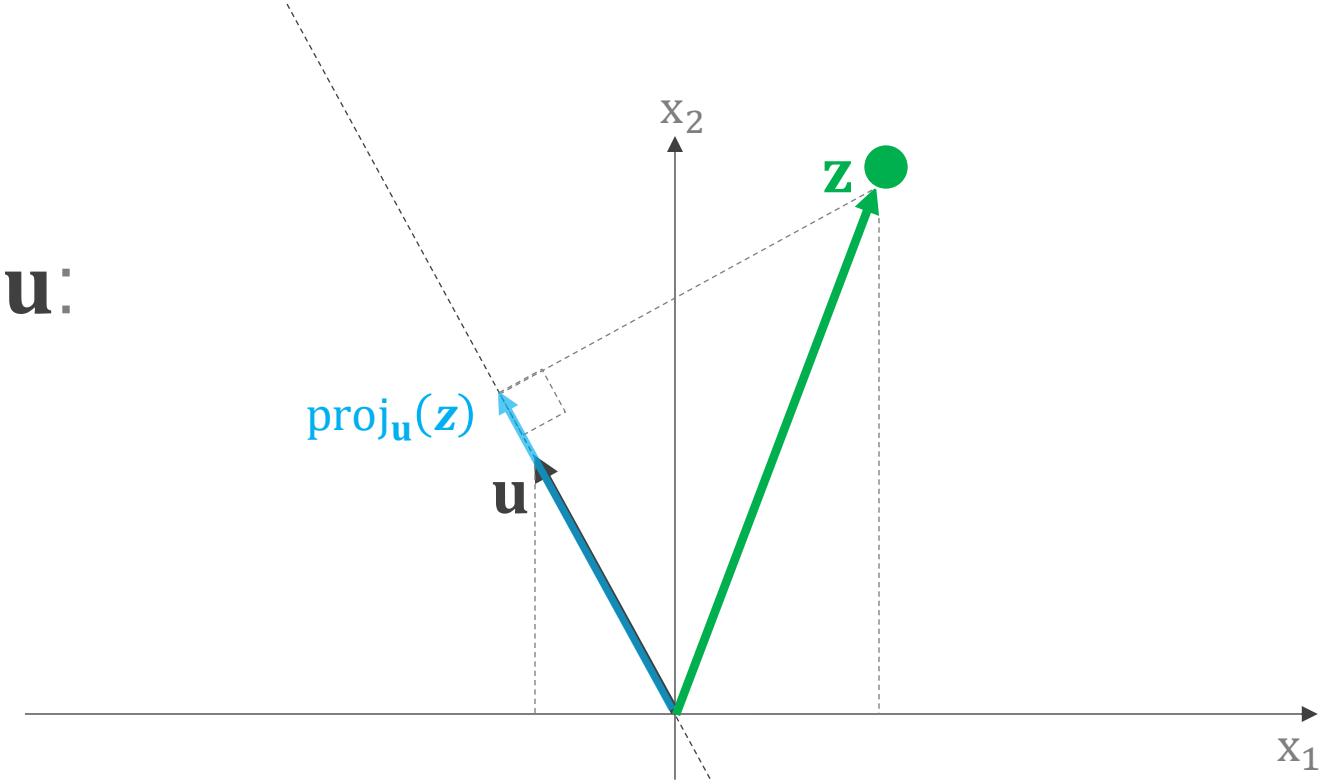
Supervised approaches

- Linear Discriminant Analysis (LDA)
- Partial Least Squares (PLS)

Projections

The vector projection of $\boldsymbol{z}$ onto $\mathbf{u}$:

$$\text{proj}_{\mathbf{u}}(\boldsymbol{z}) = \left(\frac{\mathbf{u}^T\mathbf{z}}{\|\mathbf{u}\|}\right)\frac{\mathbf{u}}{\|\mathbf{u}\|}$$



The scalar length (Euclidean or L_2 norm) of the vector $\mathbf{u}$ is $\|\mathbf{u}\|$

If we assume $\mathbf{u}$ is a unit vector then $\|\mathbf{u}\| = 1$

$$\text{proj}_{\mathbf{u}}(\boldsymbol{z}) = (\mathbf{u}^T\mathbf{z})\mathbf{u}$$

Magnitude of projection onto direction of $\mathbf{u}$

Projections

$$\mathbf{u}^T\mathbf{z} = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

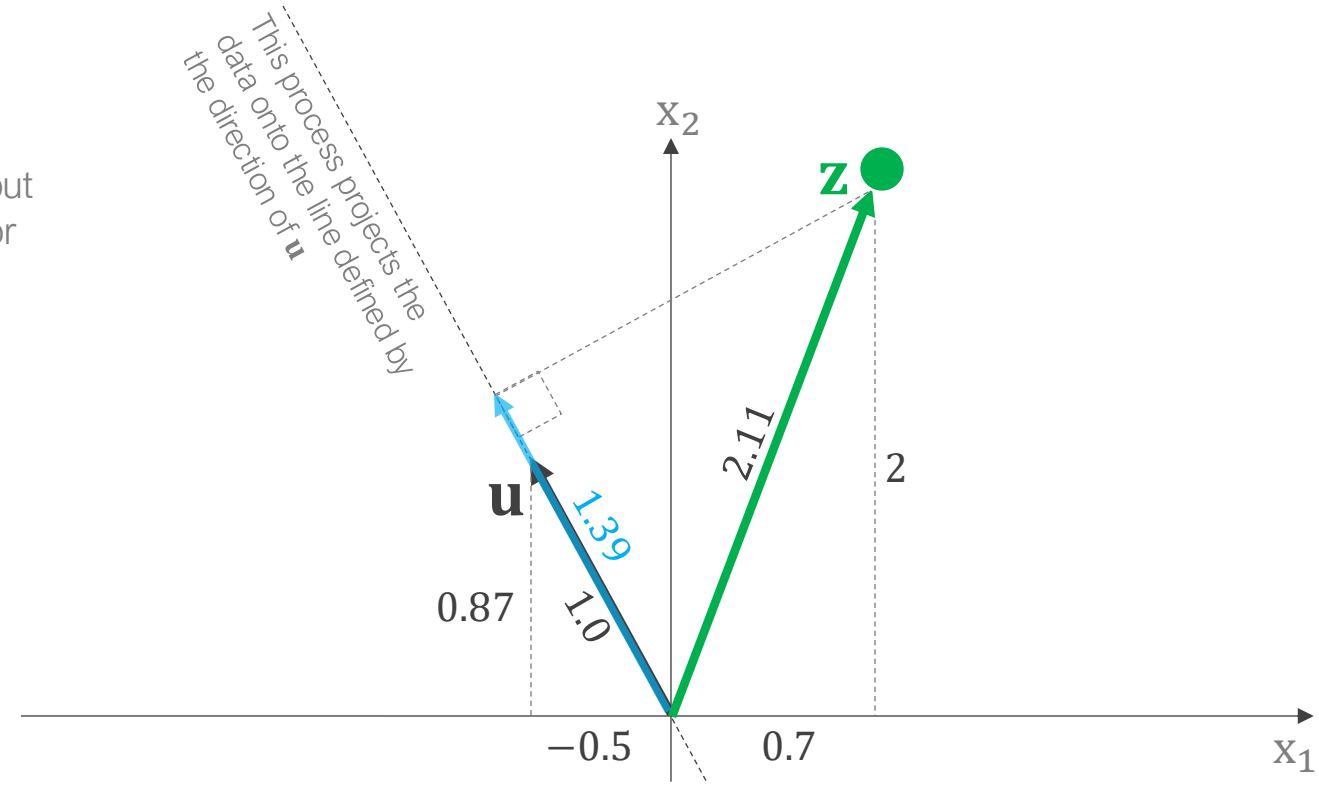
This is an inner product, but assuming $\mathbf{u}$ is a unit vector computes this as the magnitude (length) of the projection of $\mathbf{z}$ onto $\mathbf{u}$

$$= u_1 z_1 + u_2 z_2$$

$$= (-0.5)(0.7) + (0.87)(2)$$

$$= 1.39$$

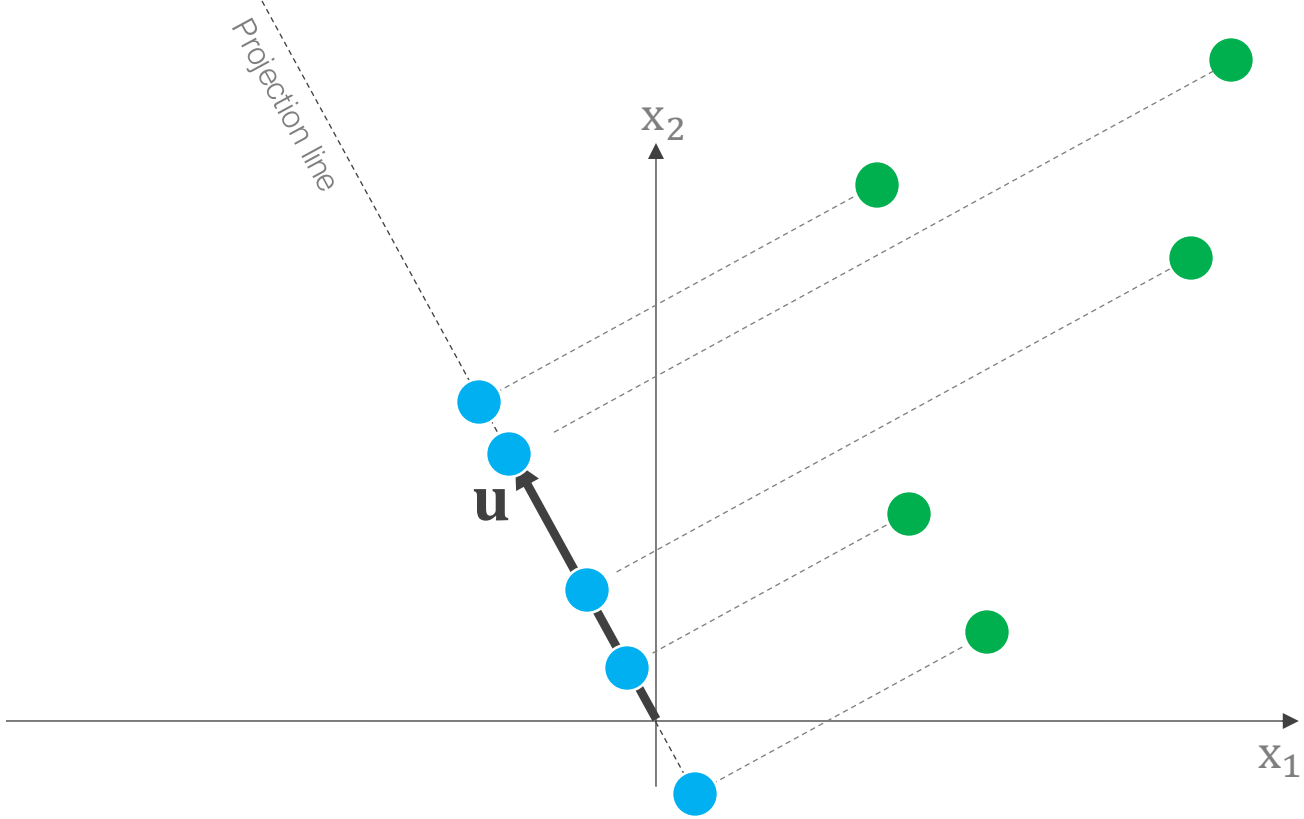
Length (magnitude) of the projection of $\mathbf{z}$ onto $\mathbf{u}$



This is valid because $\mathbf{u}$ is a unit vector (length is 1: $\|\mathbf{u}\|_2 = \sqrt{u_1^2 + u_2^2} = \sqrt{(-0.5)^2 + (0.87)^2} \cong 1$)

Projections

We could project any points in this space onto the line defined by the direction of unit vector $\mathbf{u}$



Principal Components Analysis

Our Goal:

Find a lower dimensional representation of the data that captures the variance in the data

PCA

Before you begin: Normalize the data!

For **each feature**, subtract the mean and divide by the standard deviation

Normalized feature

Raw, unscaled feature

$$x = \frac{x' - \overline{x'}}{\sigma_{x'}}$$

columns = features

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{bmatrix}$$

rows = observations

We normalize each of the columns
N = number of samples
D = number of features

a.k.a.
Karhunen–Loève Transform

Principal components analysis

Transform the data from a high dimensional space to a lower dimensional subspace, while maximizing the projected variance

$\mathbf{A}$ $\quad$ $\mathbf{x}$ $\quad \rightarrow \quad$ $\mathbf{z}$

$\mathbb{R}^{M \times D}$ $\quad$ $\mathbb{R}^{D}$ $\quad$ $\mathbb{R}^{M}$

$M < D$

$$\begin{bmatrix} u_{11} & \cdots & u_{1D} \\ \vdots & \ddots & \vdots \\ u_{M1} & \cdots & u_{MD} \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_D \end{bmatrix} \qquad \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_M \end{bmatrix}$$

linear transformation matrix

(this is what we want to find through PCA)

sample of data in original D-dimensional space

(this is one of N observations)

Transformed data in M-dimensional (lower dimensional) subspace

PCA Features are linear combinations of the input features

$$z_1 = u_{11}x_1 + u_{12}x_2 + \cdots + u_{1D}x_D$$

$$z_2 = u_{21}x_1 + u_{22}x_2 + \cdots + u_{2D}x_D$$

$$\vdots$$

$$z_M = u_{M1}x_1 + u_{M2}x_2 + \cdots + u_{MD}x_D$$

Principal components analysis

$$A$$

$$\begin{bmatrix} u_{11} & \cdots & u_{1D} \\ \vdots & \ddots & \vdots \\ u_{M1} & \cdots & u_{MD} \end{bmatrix} = \begin{bmatrix} - \ \mathbf{u}_1^T \ - \\ \vdots \\ - \ \mathbf{u}_M^T \ - \end{bmatrix}$$

linear transformation matrix

Each $\mathbf{u}_i$ represents a unit vector in $\mathbb{R}^D$

The i^{th} principal component:

$$z_i = \mathbf{u}_i^T \mathbf{x}$$

scalar — unit vector

Since only direction matters, we assume the $\mathbf{u}_i$ are unit vectors

$$\mathbf{u}_i^T \mathbf{u}_i = 1$$

PCA

We want to **maximize the variance** of the **projected data**

Goal

Let's start by finding the **unit vector in the direction of greatest variation** in the dataset

Here the magnitude is unimportant, but the direction matters

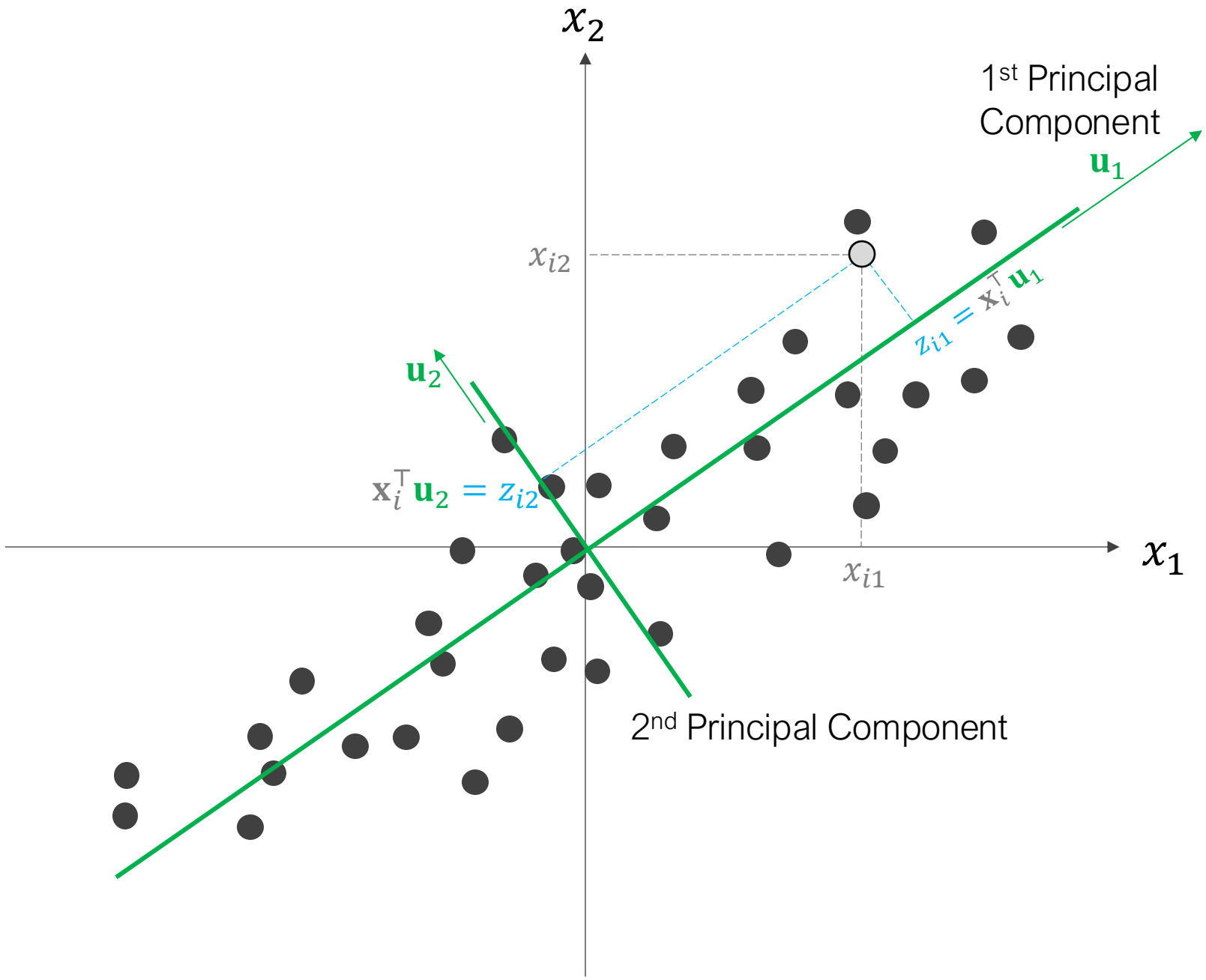
We seek to project each point $\mathbf{x}_i$ onto a unit PC vector. $z_i = \mathbf{u}_1^T \mathbf{x}_i$

Principal Components

Maximum variance formulation

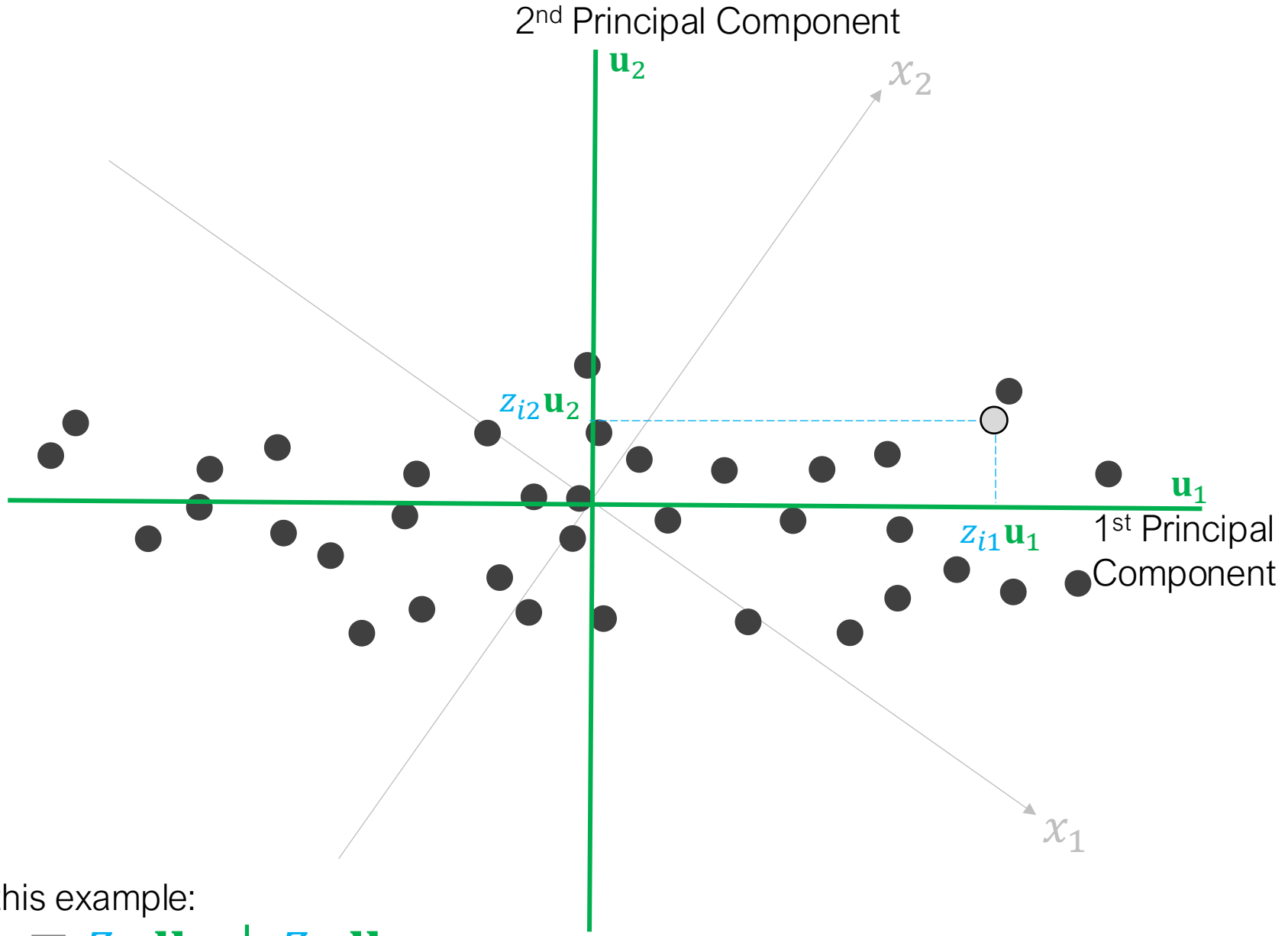
Length of a projection onto a unit vector:

$$z_{i1} = \mathbf{x}_i^\top \mathbf{u}_1$$



$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}$$

Reprojected Data onto Principal Components



Any point $\boldsymbol{x}_i$ can be represented as a combination of the principal components

$$\mathbf{x}_i = \sum_{j=1}^{D} z_{ij}\mathbf{u}_j$$

The $\mathbf{u}_j$'s are an orthogonal basis for the space $\mathbb{R}^D$

In this example:

$$\mathbf{x}_i = z_{i1}\mathbf{u}_1 + z_{i2}\mathbf{u}_2$$

PCA: Compute the variance of the transformed data

(we're starting our search just looking for the single direction of greatest variance)

Mean of the data:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \qquad \mathbf{x}_i \; [D \times 1]$$

The projected mean of the data:

$$\bar{z} = \mathbf{u}_1^T \bar{\mathbf{x}}$$

We can compute the (projected) variance of the data as:

$$\underset{\text{[scalar]}}{\sigma_z^2} = \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})^2$$

$$\sigma_z^2 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{u}_1^T\mathbf{x}_i - \mathbf{u}_1^T\bar{\mathbf{x}})^2$$

The magnitude z_i of our data $\mathbf{x}_i$ projected length on the unit vector $\mathbf{u}_1$ is:

$$z_i = \mathbf{u}_1^T \mathbf{x}_i$$

PCA: Compute the variance of the transformed data

We can compute the variance as:

$$\sigma_z^2 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{u}_1^T\mathbf{x}_i - \mathbf{u}_1^T\bar{\mathbf{x}})^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\mathbf{u}_1^T\underline{(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T}\mathbf{u}_1$$

$$= \mathbf{u}_1^T\boldsymbol{\Sigma}\mathbf{u}_1$$

Variance of the projected data

Define:

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

Covariance matrix of our data

Covariance matrix

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iD} \end{bmatrix}$$

Vector of observation i

x_{ij}

i: Observation index; j: Predictor index

$$\mathbf{x} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

Observations ↓ Predictors →

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \rightarrow [D \times D]$$

$$[D \times 1][1 \times D]$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1D} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{D1} & \Sigma_{D2} & \cdots & \Sigma_{DD} \end{bmatrix}$$

$$\Sigma_{jk} = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$$

$$= \mathrm{cov}(X_j, X_k)$$

$$= E[(X_j - \mu_j)(X_k - \mu_k)]$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \Sigma_{12} & \cdots & \Sigma_{1D} \\ \Sigma_{21} & \sigma_2^2 & \cdots & \Sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{D1} & \Sigma_{D2} & \cdots & \sigma_D^2 \end{bmatrix}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \bar{x}_j)^2$$

$$= E[(X_j - \mu_j)^2]$$

Mean of each predictor

If $\bar{x}_j = 0$ for all j

This will be the case IF the data are standardized

$$\Sigma_{jk} = \frac{1}{N}\sum_{i=1}^{N} x_{ij}x_{ik}$$

$$= \frac{1}{N}\mathbf{x}_j^T\mathbf{x}_k$$

$$= E[X_jX_k]$$

$$\boldsymbol{\Sigma} = \frac{1}{N}\mathbf{X}^{\mathrm{T}}\mathbf{X}$$

Covariance and Correlation

Relationship between covariance and correlation

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X)\text{var}(Y)}}$$

When $\text{var}(X) = \text{var}(Y) = 1$, then:

$$\text{corr}(X, Y) = \text{cov}(X, Y)$$

If each of the features have been standardized, this means this matrix is:

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1D} \\ \rho_{21} & 1 & \cdots & \rho_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{D1} & \rho_{D2} & \cdots & 1 \end{bmatrix}$$

Covariance matrix properties

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \Sigma_{12} & \cdots & \Sigma_{1D} \\ \Sigma_{21} & \sigma_2^2 & \cdots & \Sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{D1} & \Sigma_{D2} & \cdots & \sigma_D^2 \end{bmatrix}$$

Positive semidefinite ($\mathbf{v}^T\boldsymbol{\Sigma}\mathbf{v} \geq 0$ for all $\mathbf{v}$) and symmetric ($\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{\mathrm{T}}$)

All eigenvalues are positive

Eigenvectors are orthogonal

If the features (predictors), $x_1, x_2, \ldots, x_D$ are independent, $\boldsymbol{\Sigma}$ is diagonal because $\mathbf{cov}\left(X_j, X_k\right) = 0$ if $j \neq k$

PCA: Maximize the variance

We want to **maximize variance** $\sigma_z^2 = \mathbf{u}_1^T \mathbf{\Sigma} \mathbf{u}_1$

subject to $\mathbf{u}_1^T \mathbf{u}_1 = 1$ (unit vectors)

We can use **Lagrange multipliers**:

Maximize $f(x)$

subject to the constraint $g(x)$

We maximize this: $L(x, \lambda) = f(x) - \lambda g(x)$

$$f(\mathbf{x}, \mathbf{u}_i) = \mathbf{u}_1^T \mathbf{\Sigma} \mathbf{u}_1$$

$$g(\mathbf{x}, \mathbf{u}_i) = \mathbf{u}_1^T \mathbf{u}_1 - 1 = 0$$

For our case: $L(\mathbf{x}, \mathbf{u_1}, \lambda) = \mathbf{u}_1^T \mathbf{\Sigma} \mathbf{u}_1 - \lambda(\mathbf{u}_1^T \mathbf{u}_1 - 1)$

We take the derivative and set it equal to zero

PCA

$$L(\mathbf{x}, \mathbf{u_1}, \lambda) = \mathbf{u}_1^T \mathbf{\Sigma} \mathbf{u}_1 - \lambda(\mathbf{u}_1^T \mathbf{u}_1 - 1)$$

We take the derivative with respect to $\mathbf{u}_1$ and set it equal to zero

$$\frac{\partial L}{\partial \mathbf{u}_1} = \frac{\partial}{\partial \mathbf{u}_1} \mathbf{u}_1^T \mathbf{\Sigma} \mathbf{u}_1 - \frac{\partial}{\partial \mathbf{u}_1} \lambda(\mathbf{u}_1^T \mathbf{u}_1 - 1)$$

$$= 2\mathbf{\Sigma} \mathbf{u}_1 - 2\lambda \mathbf{u}_1 = 0 \quad \text{(since } \Sigma \text{ is symmetric)}$$

$$\mathbf{\Sigma} \mathbf{u}_1 = \lambda \mathbf{u}_1 \quad \rightarrow \quad \mathbf{u}_1 \text{ is an eigenvector of the covariance matrix } \mathbf{\Sigma}, \text{ and } \lambda \text{ is an eigenvalue}$$

How do we know which eigenvector to use as the first principal component?

PCA

Since we want to maximize the variance in the projected features:

We want to maximize: $\sigma_z^2 = \mathbf{u}_1^T \mathbf{\Sigma} \mathbf{u}_1$

To do that this must be true: (shown on last slide) $\mathbf{\Sigma u}_1 = \lambda \mathbf{u}_1$

So we can write: $\sigma_z^2 = \mathbf{u}_1^T \lambda \mathbf{u}_1 = \lambda \mathbf{u}_1^T \mathbf{u}_1 = \lambda$

Variance corresponding to our first principle component

Therefore we choose as our first principle component the eigenvector that corresponds to the **largest eigenvalue**

The first PC will account for the most variance, the second PC to the second most, etc.

PCA: Variance explained

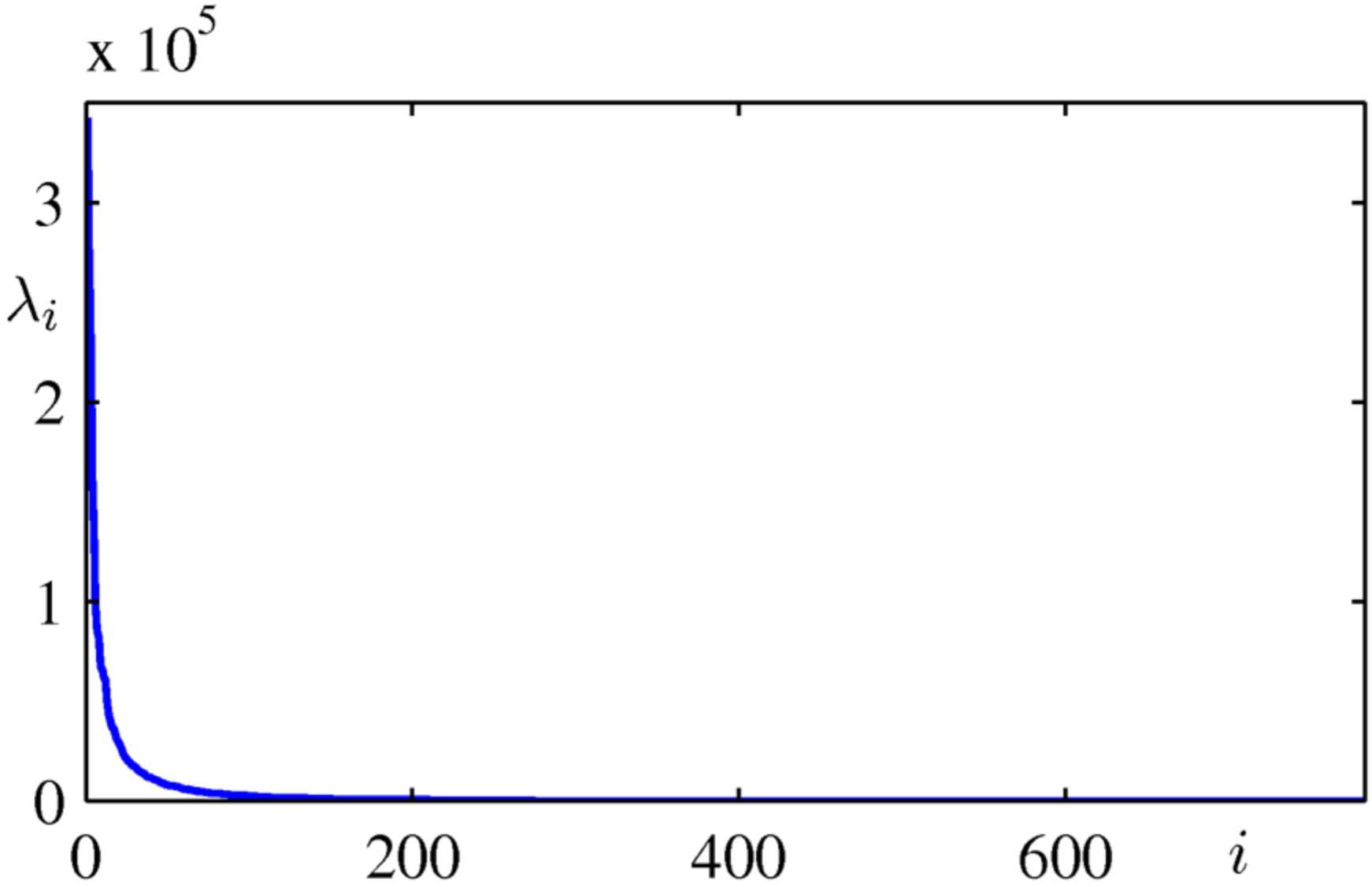
The **fraction of variance explained** $= \frac{\sum_{i=1}^{M} \lambda_i}{\sum_{i=1}^{D} \lambda_i}$

M = dimensionality of the subspace

D = dimensionality of the original data space

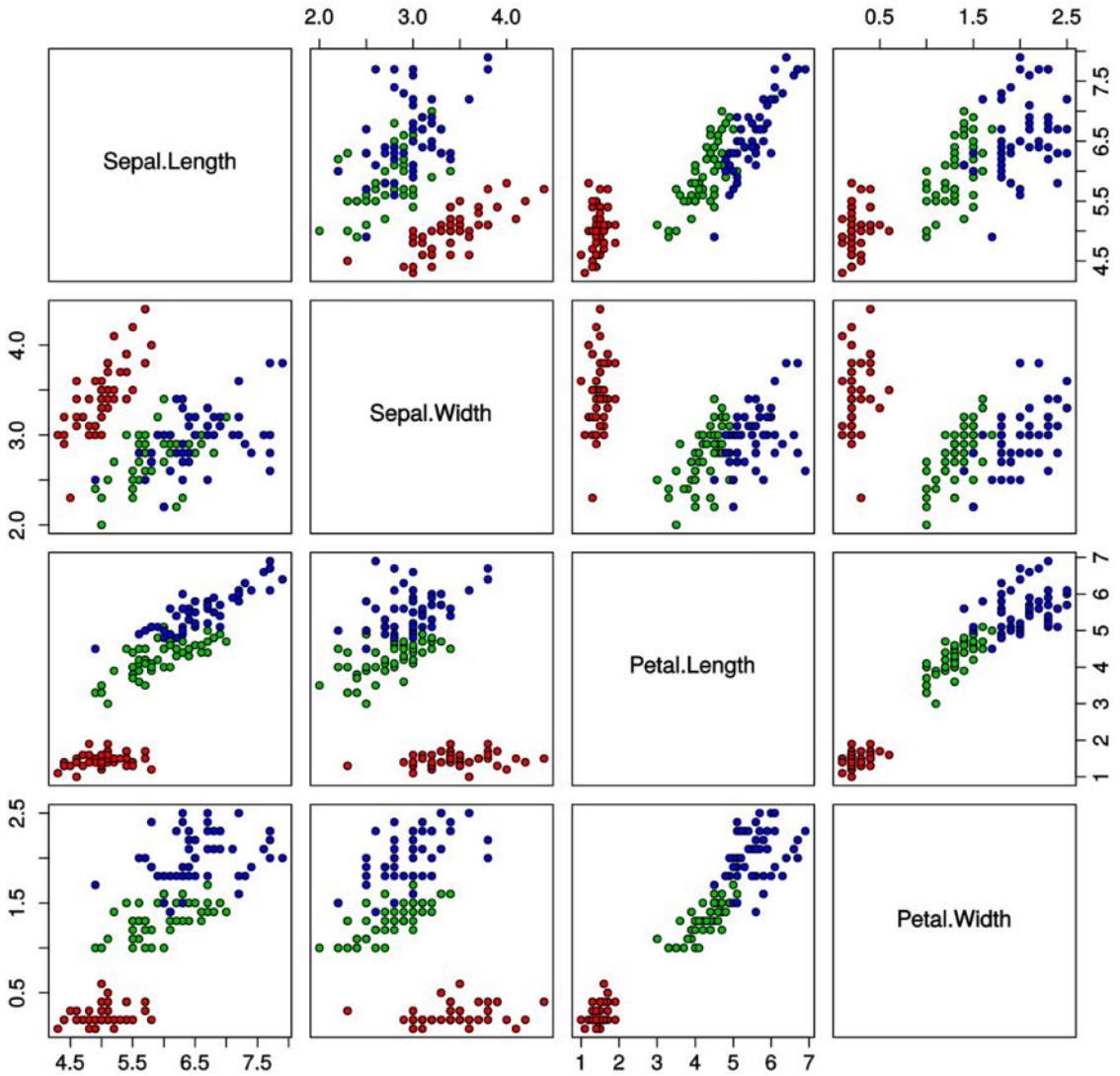
The more principle components included, the more of the variance will be represented in the projected data

Eigenvalues by principal component i



Bishop, Pattern Recognition, 2006

Scatterplot Matrix



Parallel Coordinates Plot

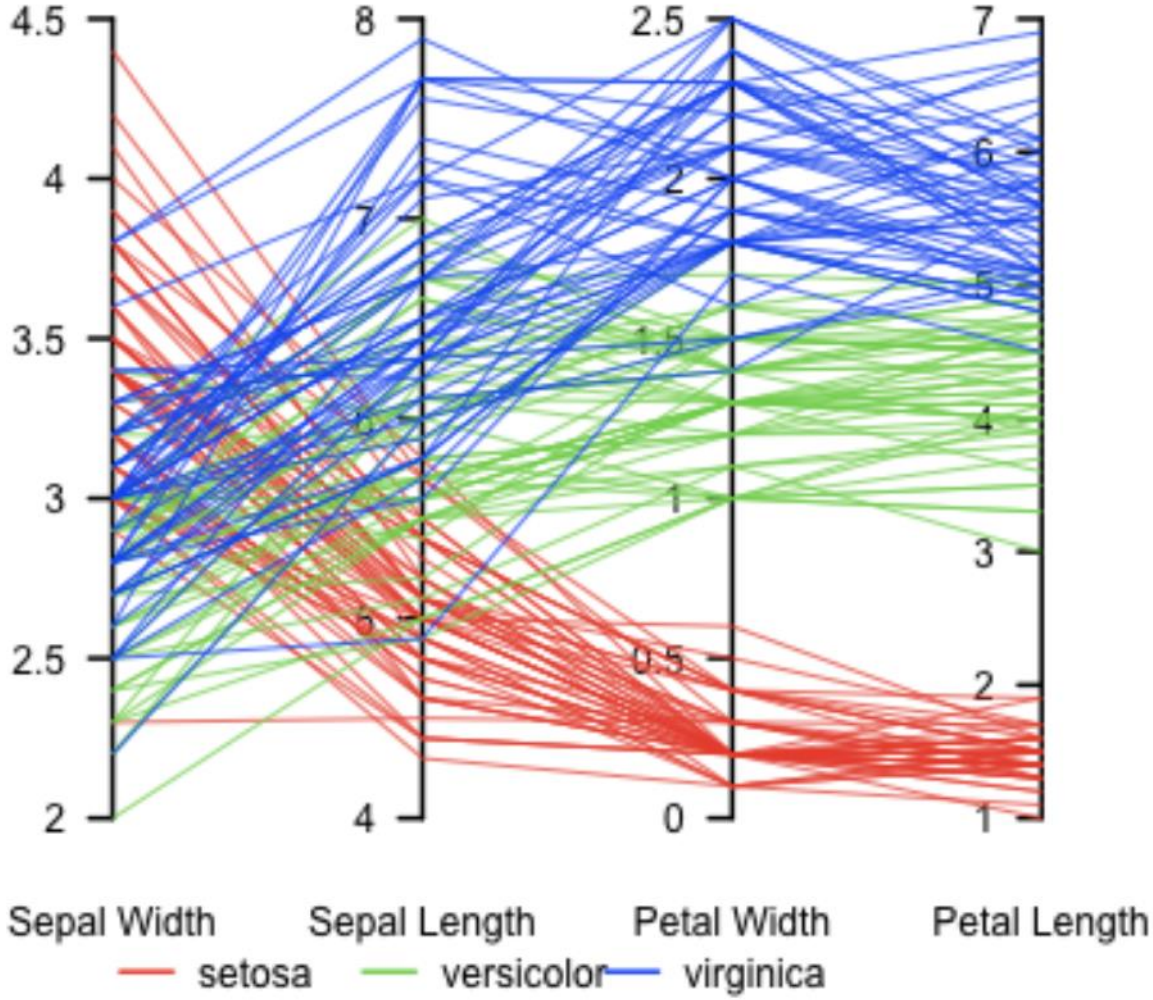


Image: Pezzotti, N., 2019. Dimensionality-Reduction Algorithms for Progressive Visual Analytics.

Interpreting PCA

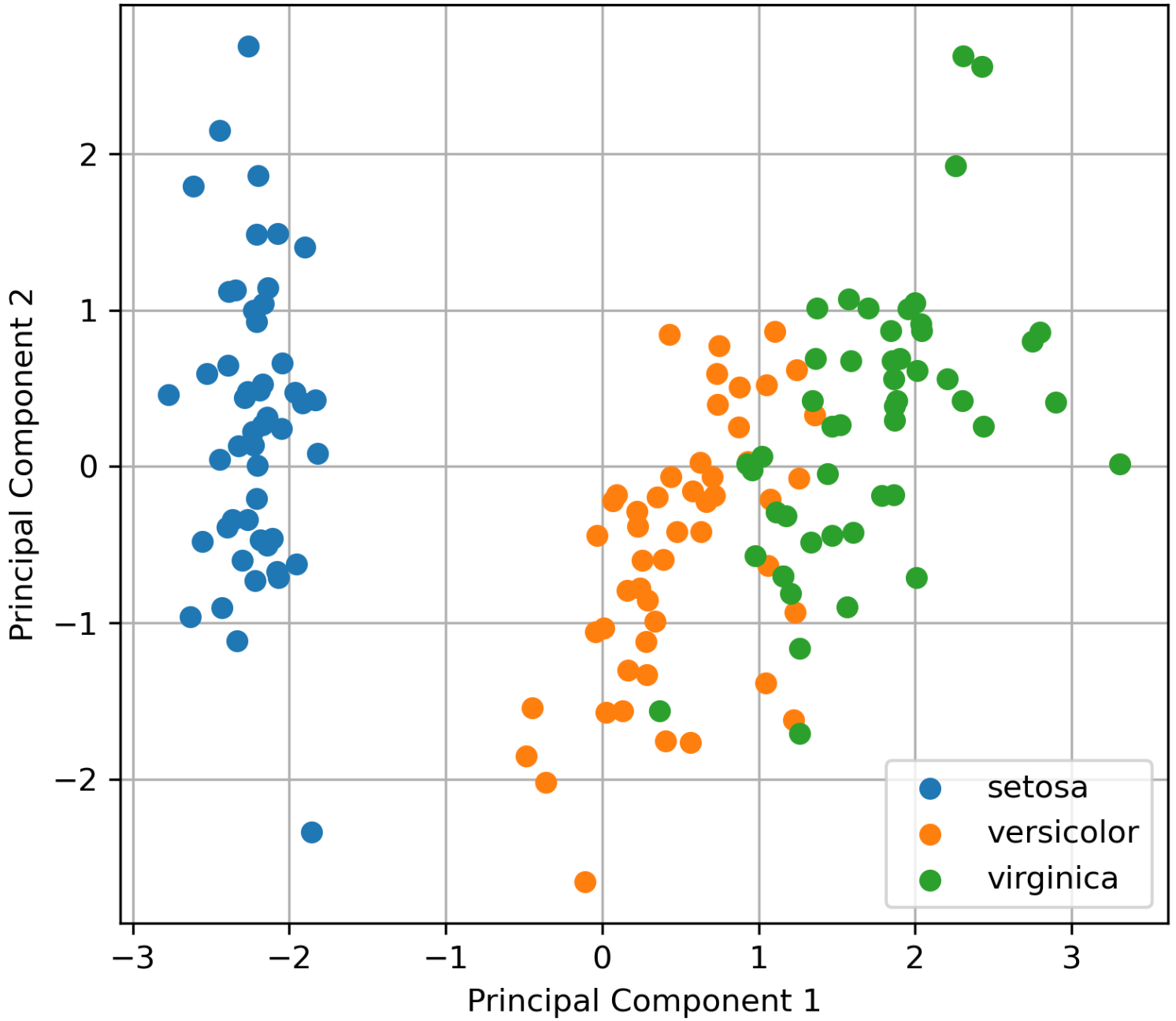
We learn a transformation:

$$\mathbf{z} = A\mathbf{x}$$

Here we can see our PCA-transformed IRIS data

What if we transformed a vector for each ORIGINAL feature ?

(sepal width, sepal length, petal width, petal length)



Interpreting PCA

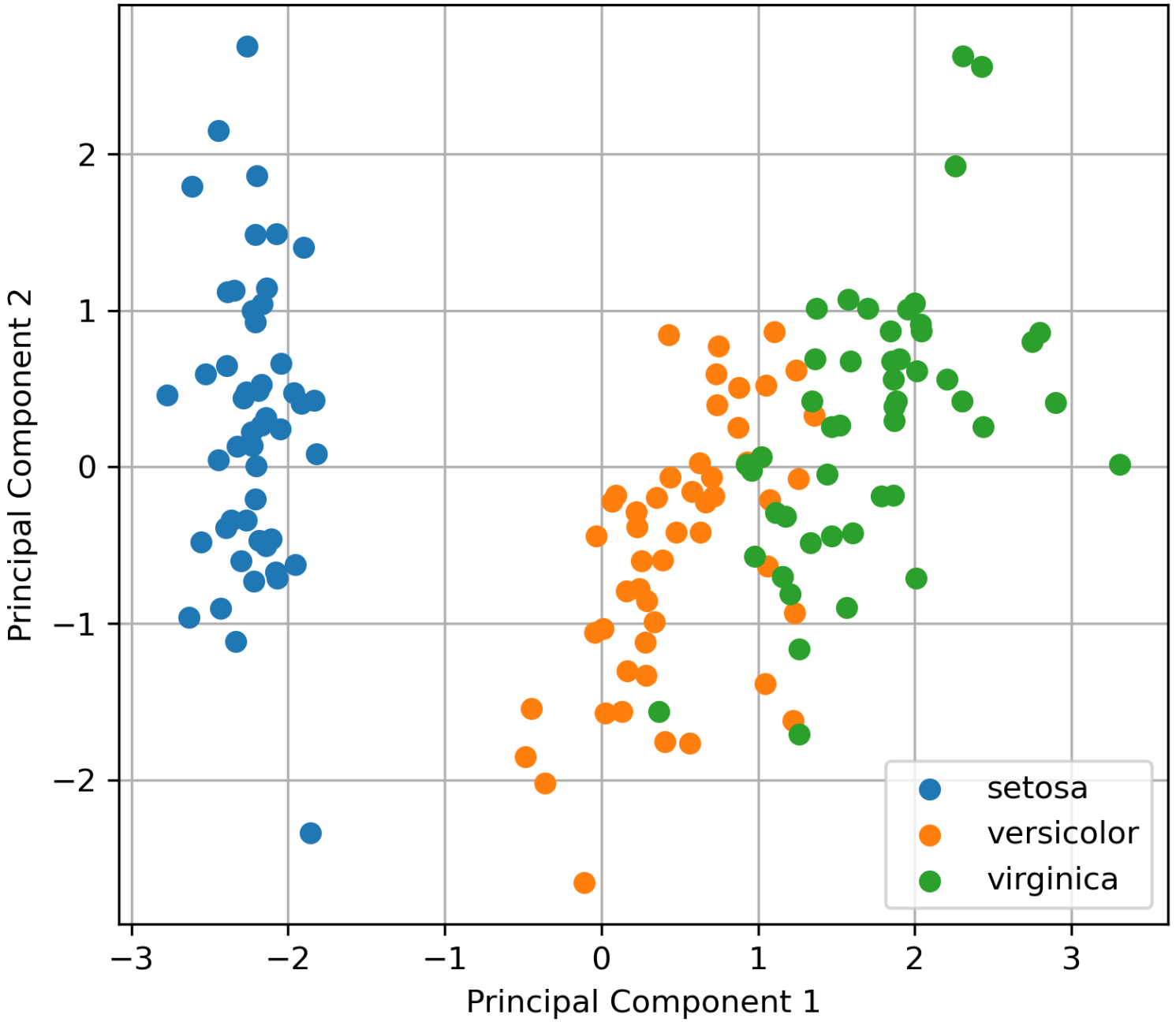
We learn a transformation:

$$\mathbf{z} = A\mathbf{x}$$

$$A = \begin{array}{c} \begin{matrix} \text{PC 1} & \text{PC 2} \end{matrix} \\ \begin{bmatrix} 0.52 & 0.38 \\ -0.27 & 0.92 \\ 0.58 & 0.02 \\ 0.56 & 0.07 \end{bmatrix} \end{array} \begin{array}{l} \text{sepal width} \\ \text{sepal length} \\ \text{petal width} \\ \text{petal length} \end{array}$$

$$\underset{\text{2-D}}{\mathbf{z}_{\text{sepal width}}} = A\underset{\text{4-D}}{\mathbf{x}_{\text{sepal width}}}$$

$$\mathbf{x}_{\text{sepal width}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{array}{l} \text{sepal width} \\ \text{sepal length} \\ \text{petal width} \\ \text{petal length} \end{array}$$



Interpreting PCA

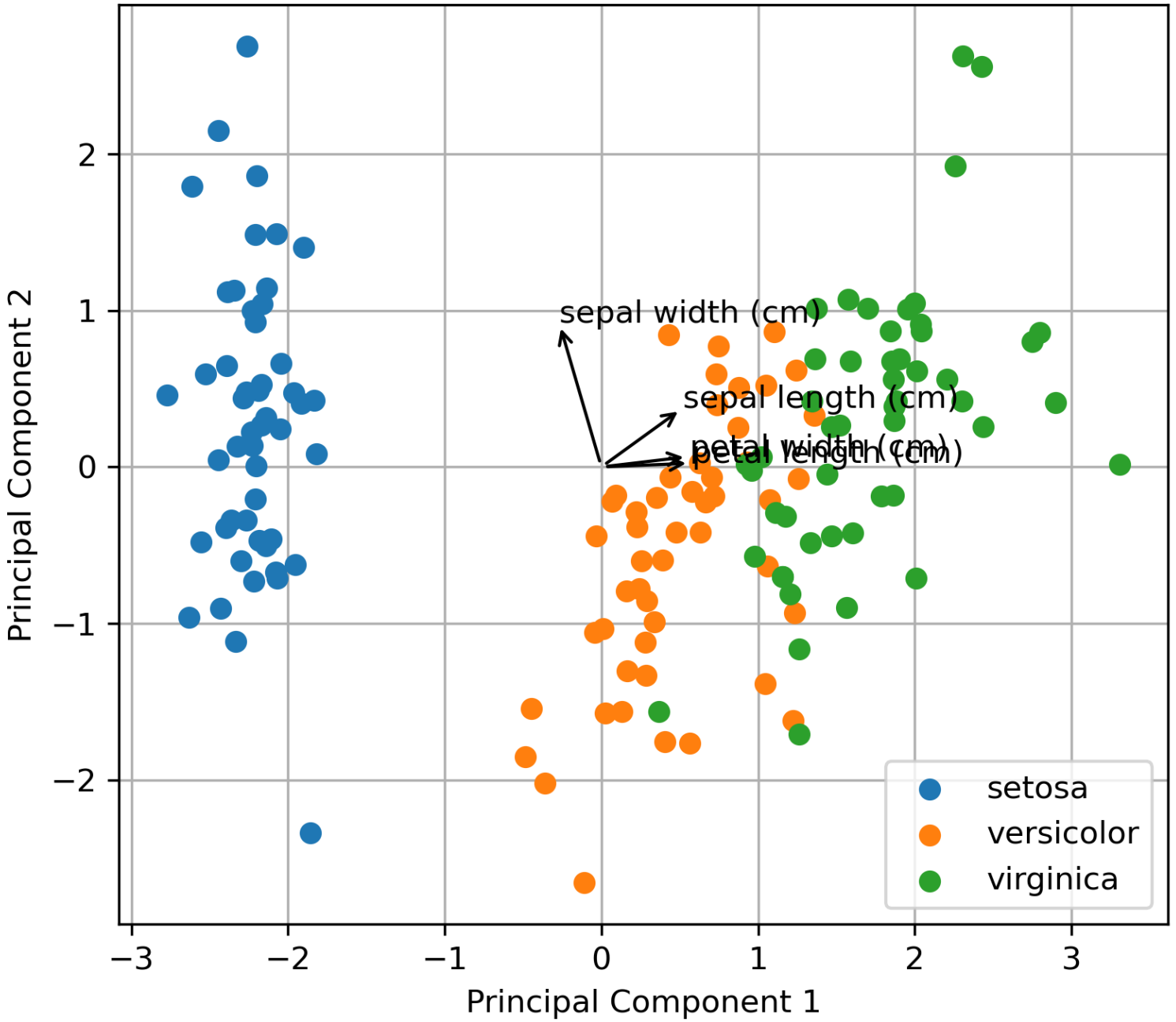
We learn a transformation:

$$\mathbf{z} = A\mathbf{x}$$

$$A = \begin{array}{c} \begin{matrix} \text{PC 1} & \text{PC 2} \end{matrix} \\ \begin{bmatrix} 0.52 & 0.38 \\ -0.27 & 0.92 \\ 0.58 & 0.02 \\ 0.56 & 0.07 \end{bmatrix} \end{array} \begin{array}{l} \text{sepal width} \\ \text{sepal length} \\ \text{petal width} \\ \text{petal length} \end{array}$$

$$\underset{\text{2-D}}{\mathbf{z}_{\text{sepal width}}} = A\underset{\text{4-D}}{\mathbf{x}_{\text{sepal width}}}$$

$$\mathbf{x}_{\text{sepal width}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{array}{l} \text{sepal width} \\ \text{sepal length} \\ \text{petal width} \\ \text{petal length} \end{array}$$



Interpreting PCA

Zooming in…

We learn a transformation:

$$\mathbf{z} = A\mathbf{x}$$

$$A = \begin{bmatrix} 0.52 & 0.38 \\ -0.27 & 0.92 \\ 0.58 & 0.02 \\ 0.56 & 0.07 \end{bmatrix}$$

PC 1 PC 2

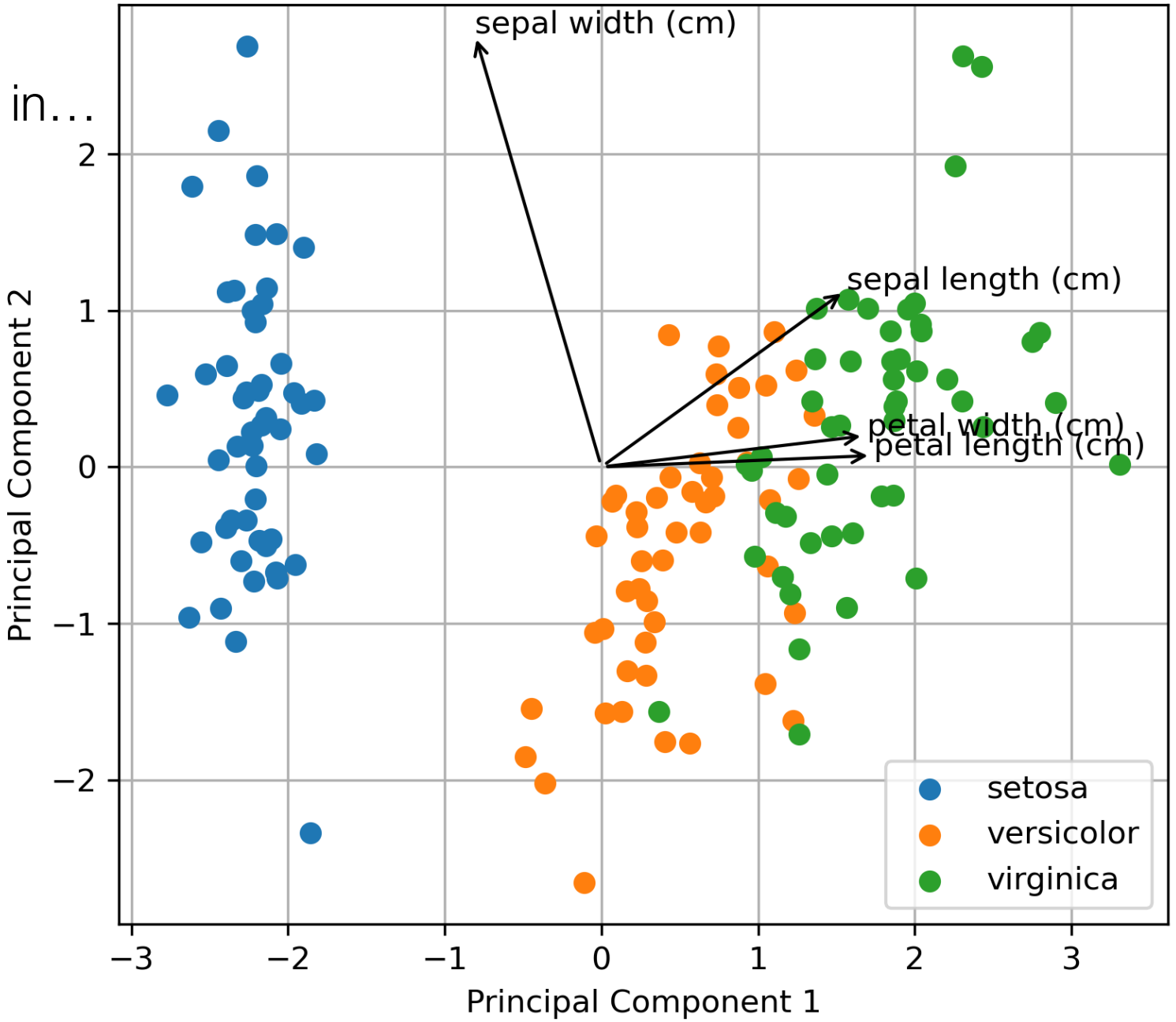
sepal width
sepal length
petal width
petal length

$$\mathbf{z}_{\text{sepal width}} = A\mathbf{x}_{\text{sepal width}}$$

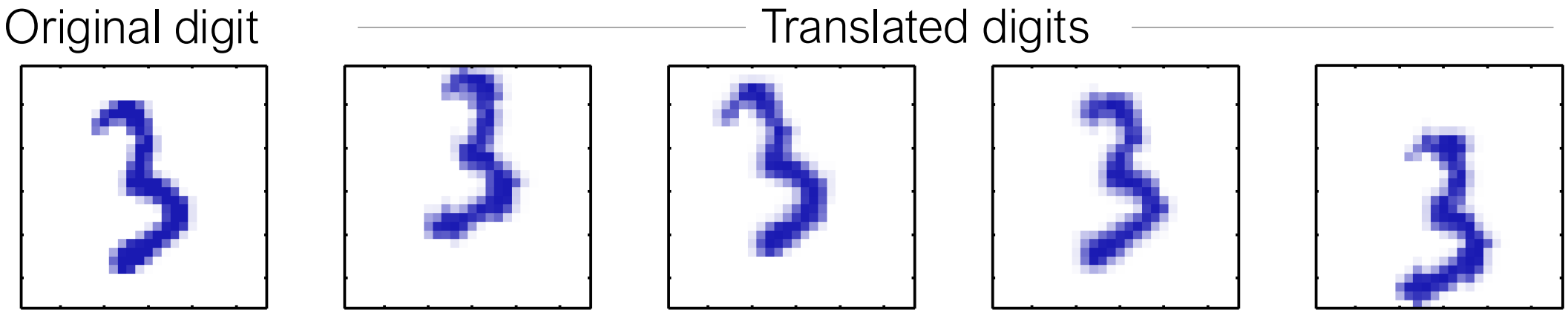
2-D 4-D

$$\mathbf{x}_{\text{sepal width}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

sepal width
sepal length
petal width
petal length



Example: translated digits



Types of translation:
1. Horizontal translation
2. Vertical translation
3. Rotation

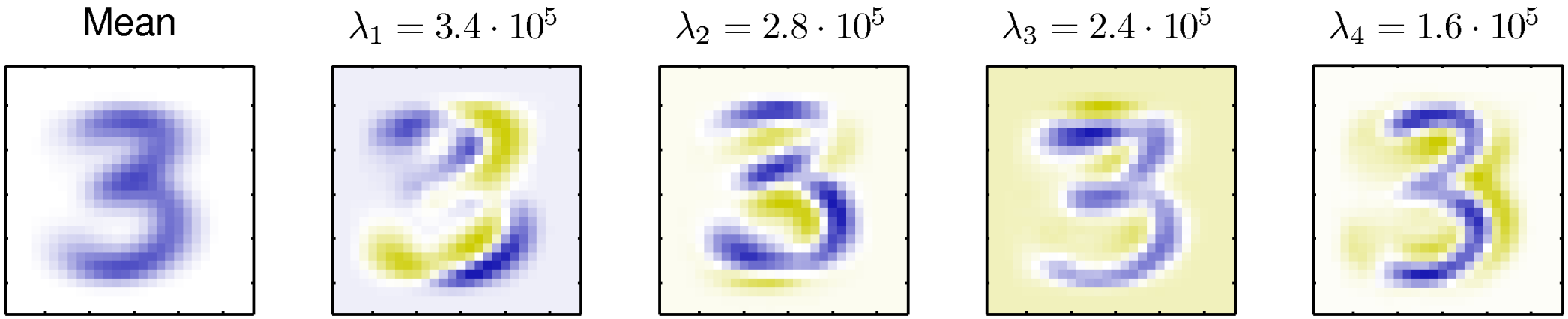
Size: 100 x 100 pixels

Bishop, Pattern Recognition, 2006

Example: translated digits

PCA identifies a new way to represent the data

Examples of first four principle component eigenvectors and eigenvalues:



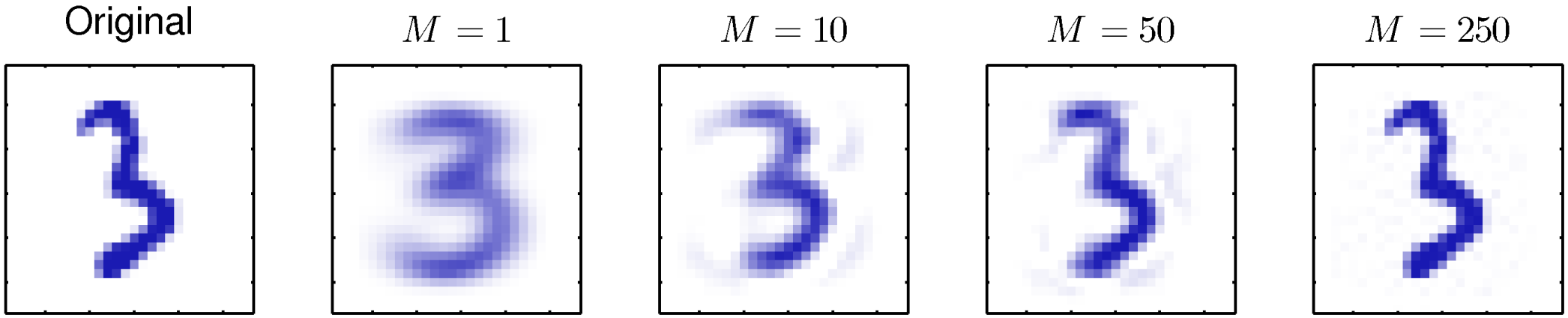
Latent Space DEMO

This links to the lower dimensional representation space of the data
(a.k.a. the latent space or embedding space) for PCA-reduced features with MNIST data

Bishop, Pattern Recognition, 2006

Example: translated digits

Reconstructed examples using different numbers of principal components:



Bishop, Pattern Recognition, 2006

Extracting principal components

size | example

0 **Goal**: reduce the dimensionality of our data from D to M, where $M < D$

columns = features (D)

$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{bmatrix}$$ rows = observations (N)

$[N \times D]$

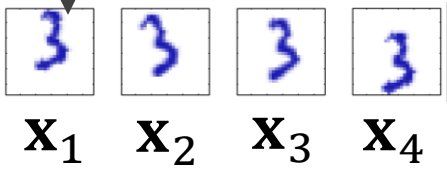
1 Normalize each feature to mean zero and a standard deviation of 1

Each observation as a vector:

$$\mathbf{x}_i \quad i = 1, \ldots, D$$

$[D \times 1]$

Each pixel represents a feature



$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$

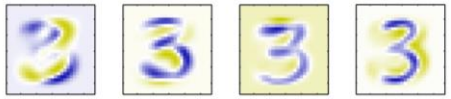
2 Determine the principal components

Calculate the eigenvectors and eigenvalues of the data covariance matrix, $\boldsymbol{\Sigma}$

Eigenvectors in descending order of their eigenvalues are the principal components

$\mathbf{u}_i$ eigenvectors / principal components

$[D \times 1]$



$\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_3$ $\mathbf{u}_4$

λ_i eigenvalues (how much of the variance is explained)

$[\text{scalar}]$

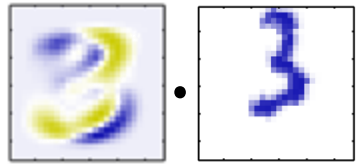
$i = 1, \ldots, D$

3 Project the data features on the principal components

$$z_{ij} = \mathbf{u}_j^T \mathbf{x}_i \qquad \begin{matrix} j = 1, \ldots, D \\ i = 1, \ldots, N \end{matrix}$$

$[\text{scalar}]$

$\mathbf{u}_1 \cdot \mathbf{x}_1 = z_{11}$



$= 2.43$

$[D \times 1]$ $[D \times 1][\text{scalar}]$

4 Keep the top M principal components to reduce into a lower dimension

$$\mathbf{A} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_M]$$

$[D \times M]$

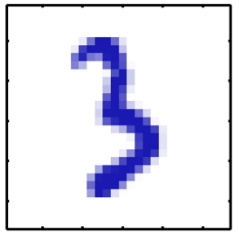
$$\mathbf{z}_i = \mathbf{A}^T \mathbf{x}_i \qquad i = 1, \ldots, N$$

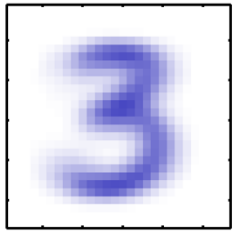
Images from Bishop, Pattern Recognition, 2006

Reconstructing our data from principal components

Sum the product of our projected data, $\mathbf{z}_i$, and our principle components

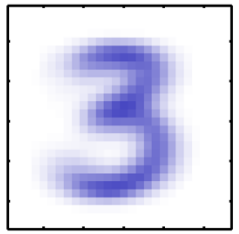
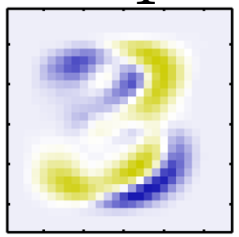
$$\hat{\mathbf{x}}_i = \sum_{j=1}^{M} z_{ij}\mathbf{u}_j$$

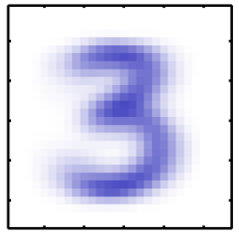
Example: the i[th] observation: $\mathbf{x}_i =$ 

$\bar{\mathbf{x}} =$ 

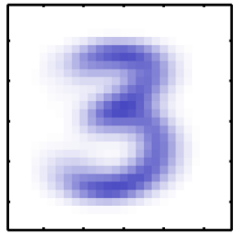
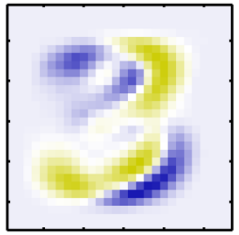
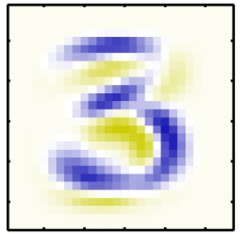
PCA-projected data: $\mathbf{z}_i = [z_{i1}, z_{i2}, \ldots, z_{iM}]$

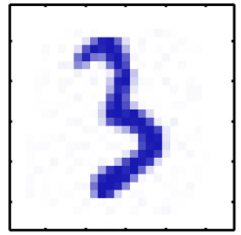
$M = 1$

$\hat{\mathbf{x}}_i =$  $+ z_{i1} \cdot$ 

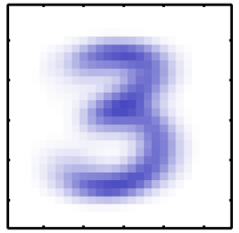
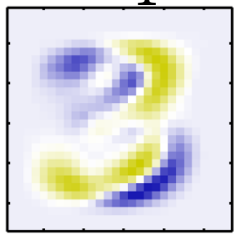
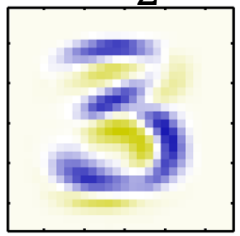
$=$ 

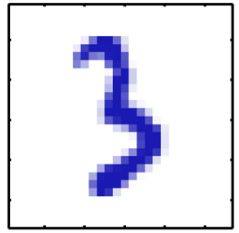
$M = 250$

$\hat{\mathbf{x}}_i =$  $+ z_{i1} \cdot$  $+ z_{i2} \cdot$  $+ \sum_{j=3}^{250} z_{ij}\mathbf{u}_j$

$=$ 

$M = 10{,}000$

$\hat{\mathbf{x}}_i =$  $+ z_{i1} \cdot$  $+ z_{i2} \cdot$  $+ \sum_{j=3}^{10{,}000} z_{ij}\mathbf{u}_j$

$=$ 

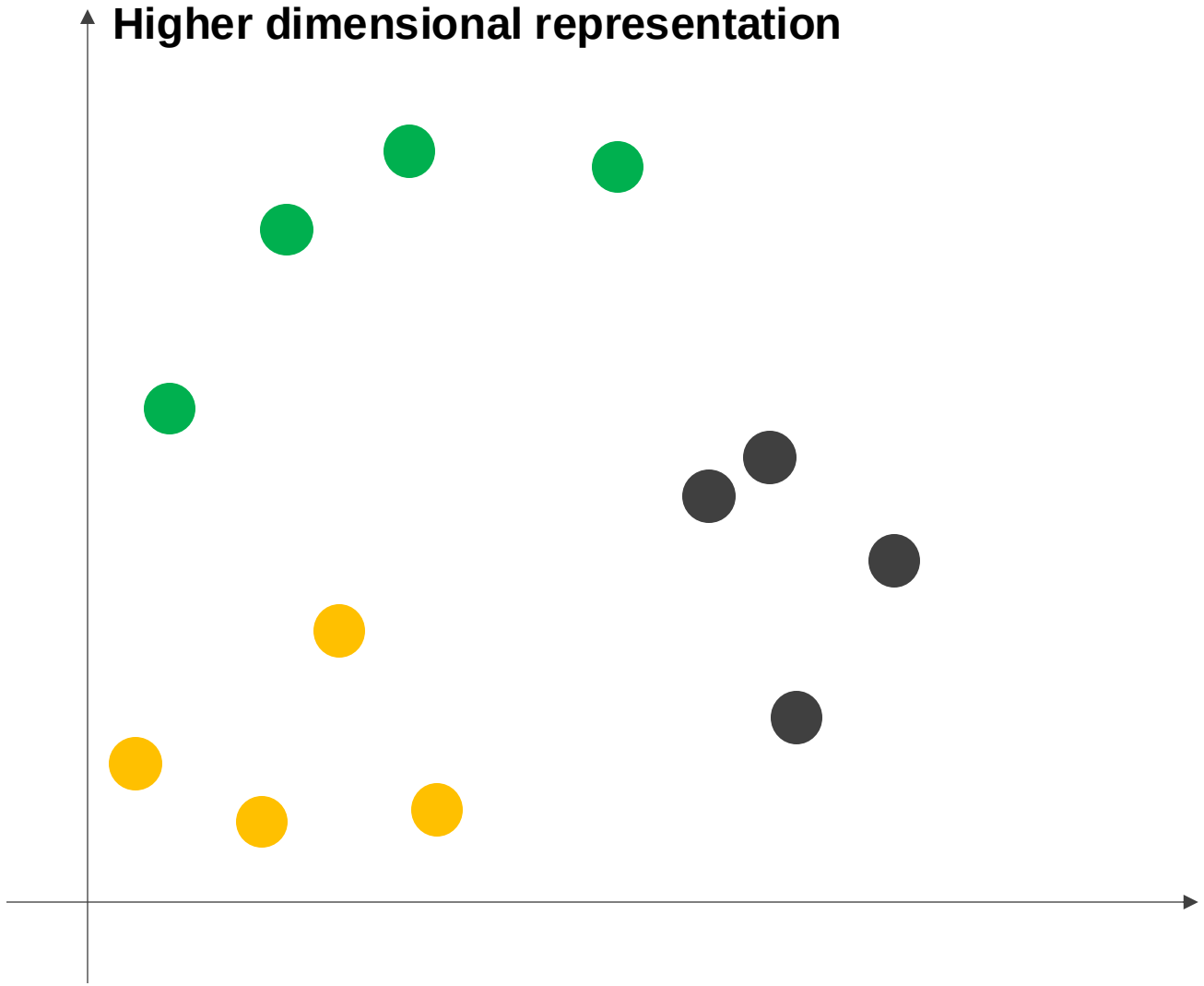
(perfect reconstruction)

Images from Bishop, Pattern Recognition, 2006

IMPORTANT APPLICATION NOTE

If you use PCA as preprocessing for supervised learning, learn the PCA transform from the training set and apply that learned transform to your test set

UMAP



Finds a low-dimensional embedding (e.g., 2 or 3D) that preserves the high-dimensional relationships

Minimizing the difference between the high-dimensional and low-dimensional probability distributions

Lower dimensional representation

Unlike t-SNE, which primarily preserves local structure, UMAP also maintains some global structure

Learns a transformation that allows for new points to be added without rerunning the optimization

UMAP Example

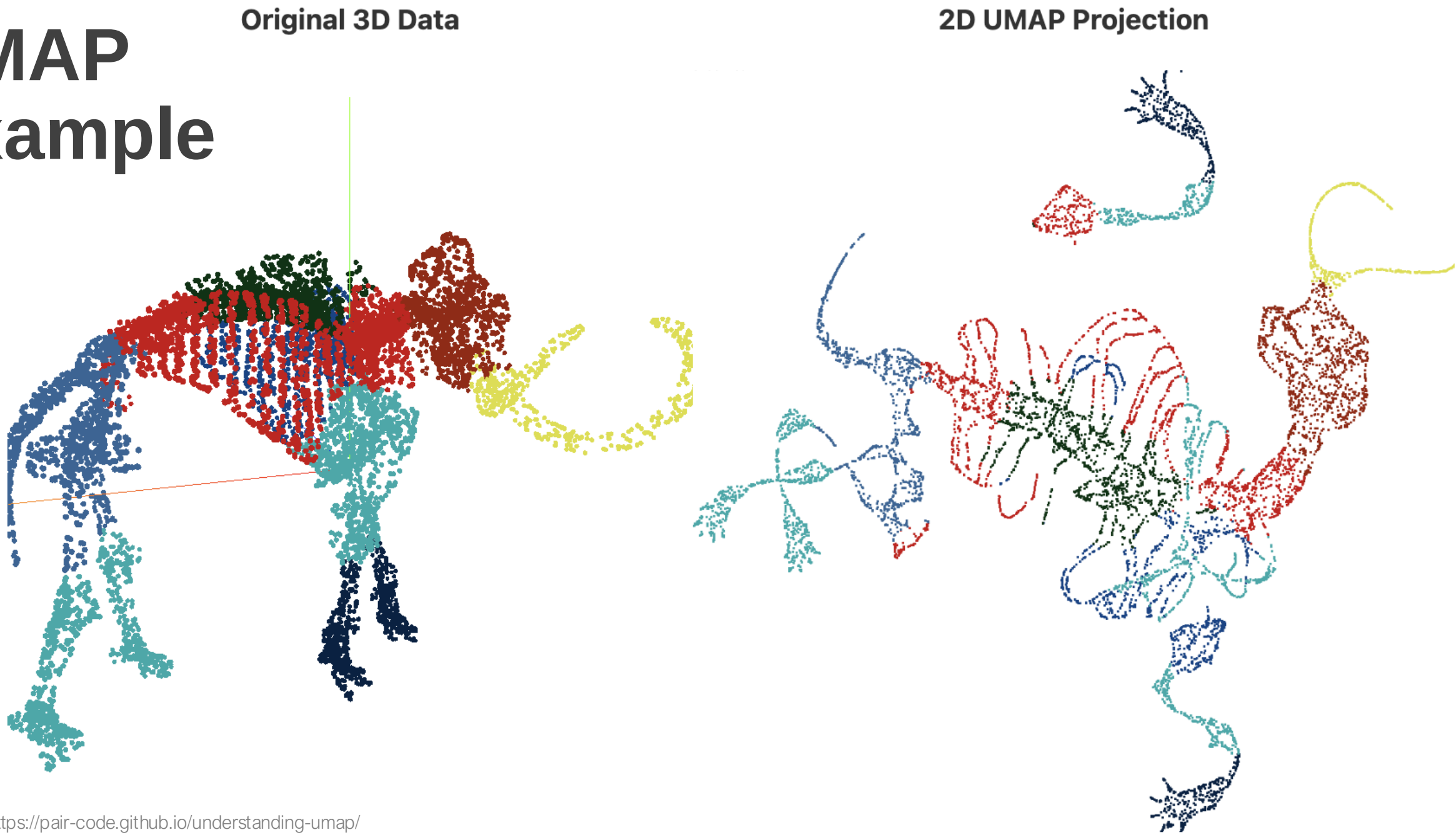


Image from https://pair-code.github.io/understanding-umap/

Autoencoders

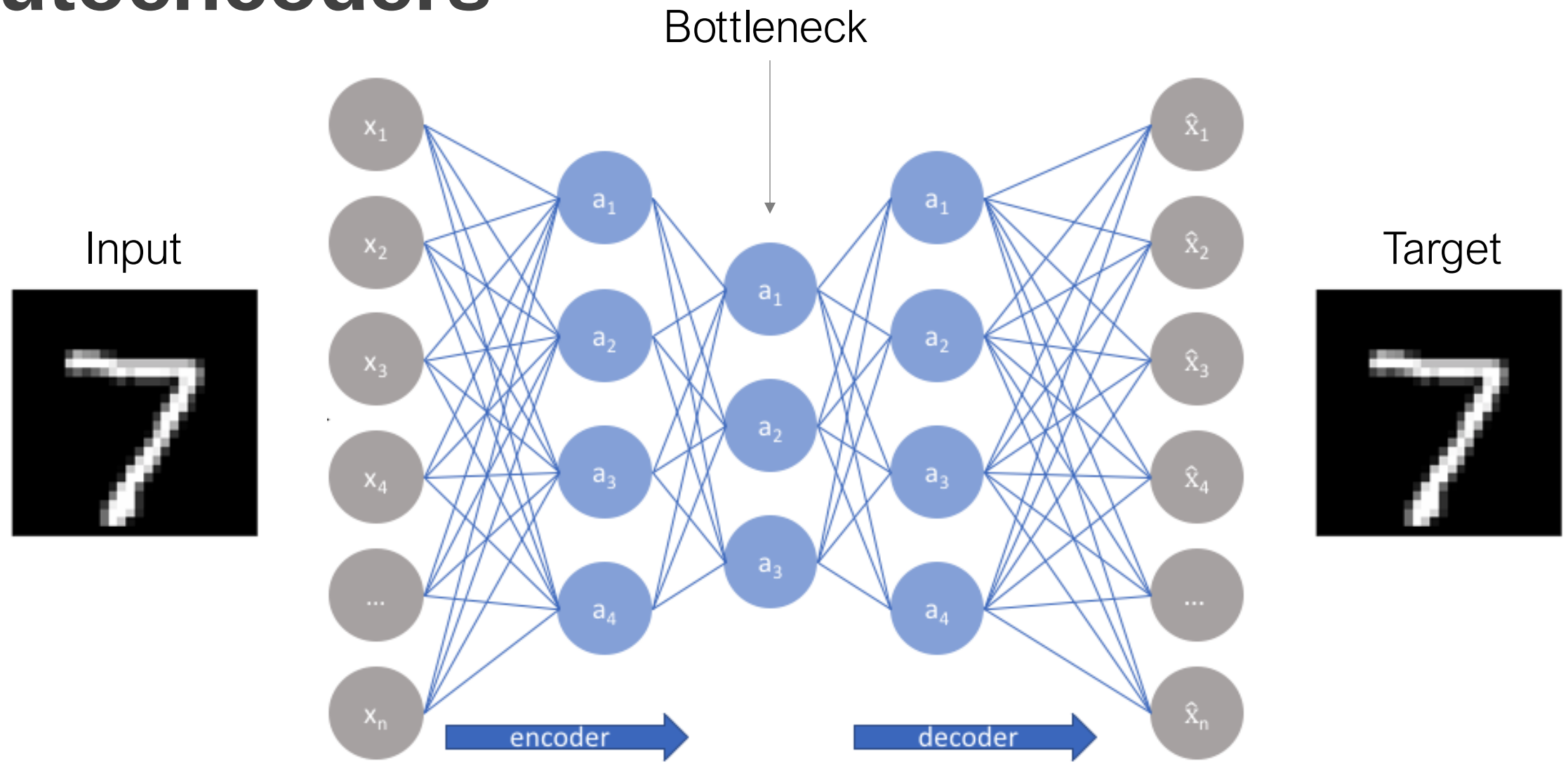


Image from: https://www.jeremyjordan.me/autoencoders/

Masked Autoencoders

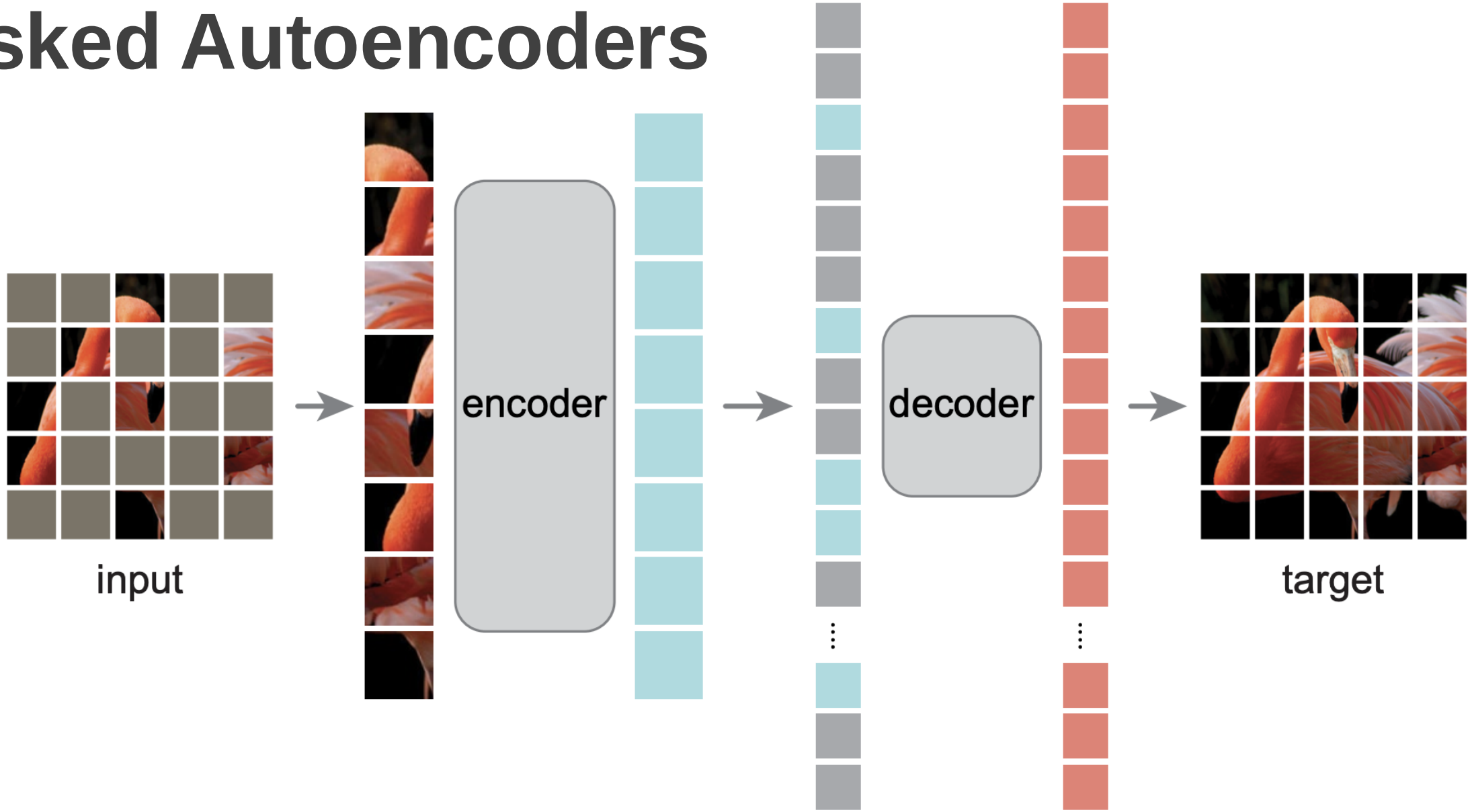


Image from: He, K., Chen, X., Xie, S., Li, Y., Dollár, P. and Girshick, R., 2022. Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 16000-16009).

Masked Autoencoders

Unsupervised learning: describing data

1

Dimensionality Reduction

Developing new data representations

- Feature subset selection
- Feature projections
- Supervised approaches

2

Density Estimation

Quantifying data distributions

- Histograms
- Nonparametric density estimation
- Parametric models

3

Clustering

Grouping similar data

- Hierarchical
- Centroid-based
- Distribution-based
- Density-based

4

Other Unsupervised Learning Tools

- Anomaly detection
- Representation learning
- Generative models