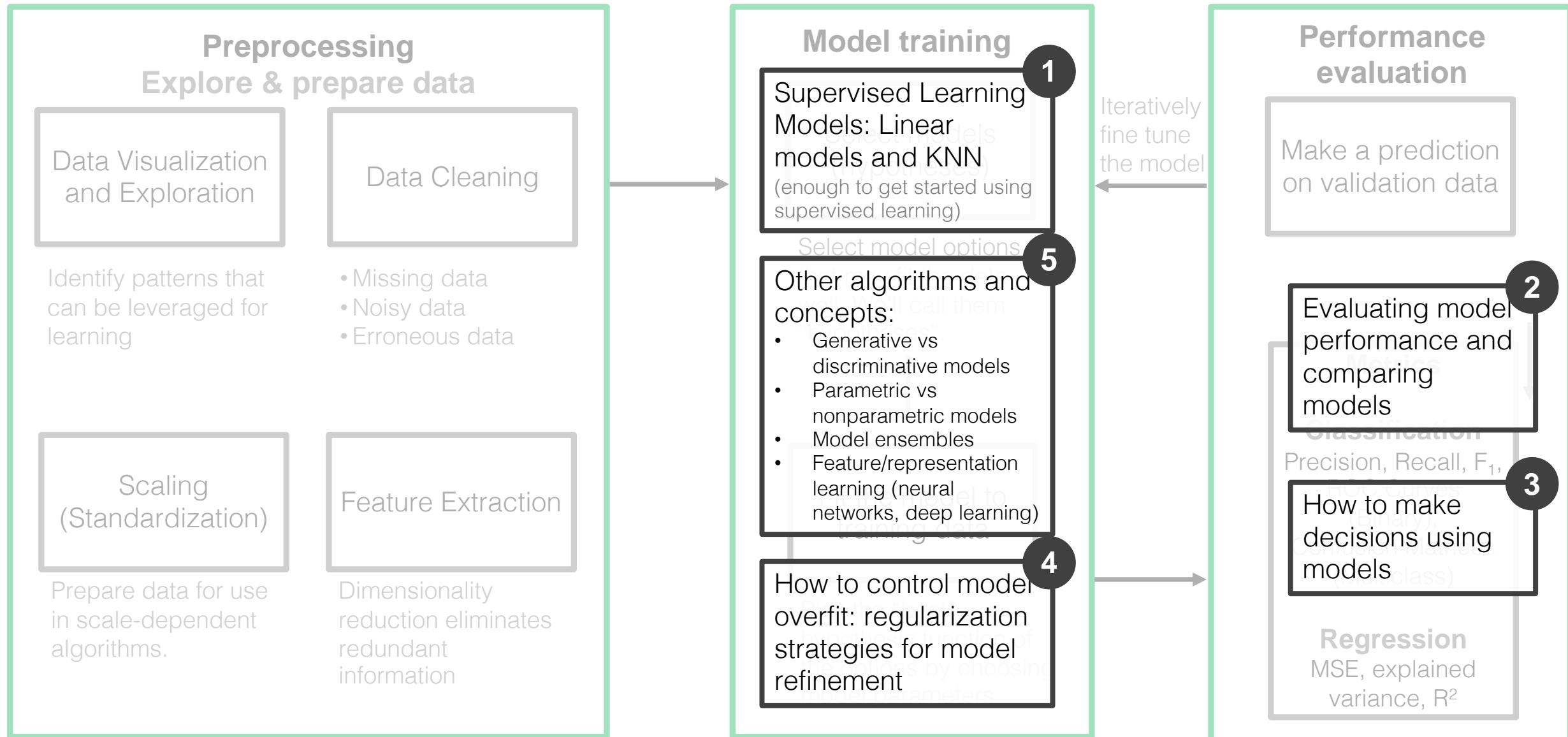


# Unsupervised Learning

# Supervised learning in practice



# Types of machine learning

	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Goal	<b>Predict</b> ...from examples	<b>Describe</b> ...structure in data	<b>Strategize</b> learn by trial and error
Data	$(x, y)$	$x$	delayed feedback
Types	<ul style="list-style-type: none"><li>Classification</li><li>Regression</li></ul>	<ul style="list-style-type: none"><li>Density estimation</li><li>Clustering</li><li>Dimensionality reduction</li><li>Anomaly detection</li></ul>	<ul style="list-style-type: none"><li>Model-free learning</li><li>Model-based learning</li></ul>

# 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

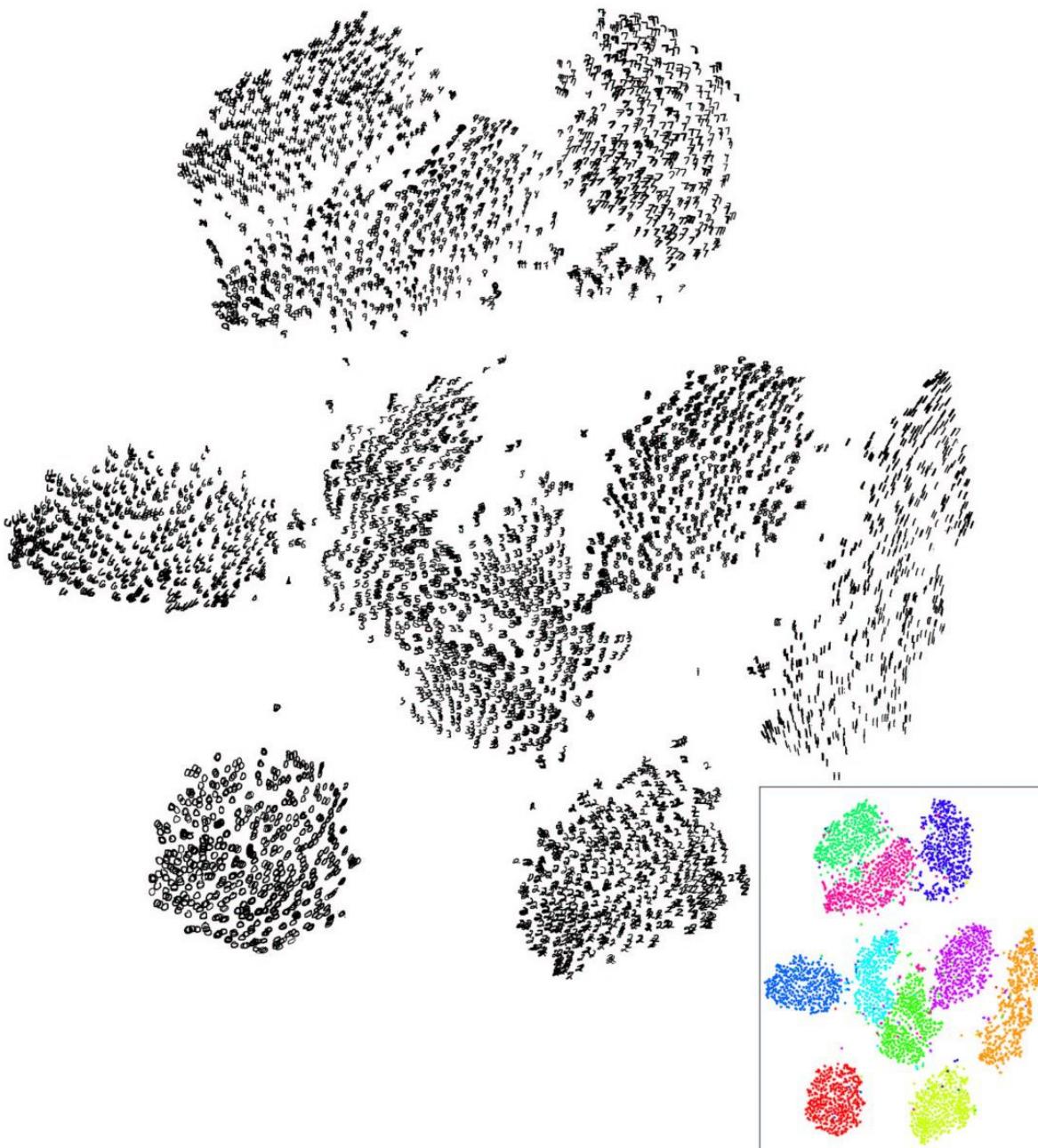
## Anomaly detection

Identifying anomalies in data

- Probabilistic approaches
- Cluster-based
- Supervised approaches

# Dimensionality Reduction

# Reducing MNIST dimensionality into 2 dimensions



## 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).



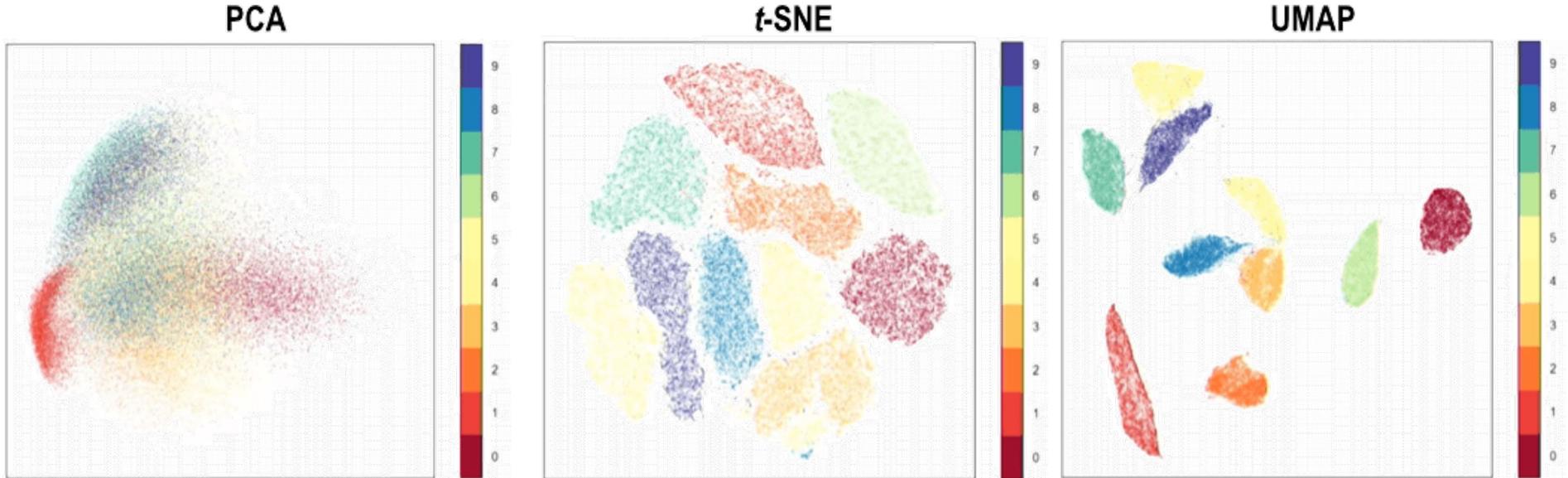
Van der Maaten, L. and Hinton, G., 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).



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:  
 $(28 \times 28 = 784)$



Fashion-MNIST  
Dataset  
reduced to 2  
dimensions

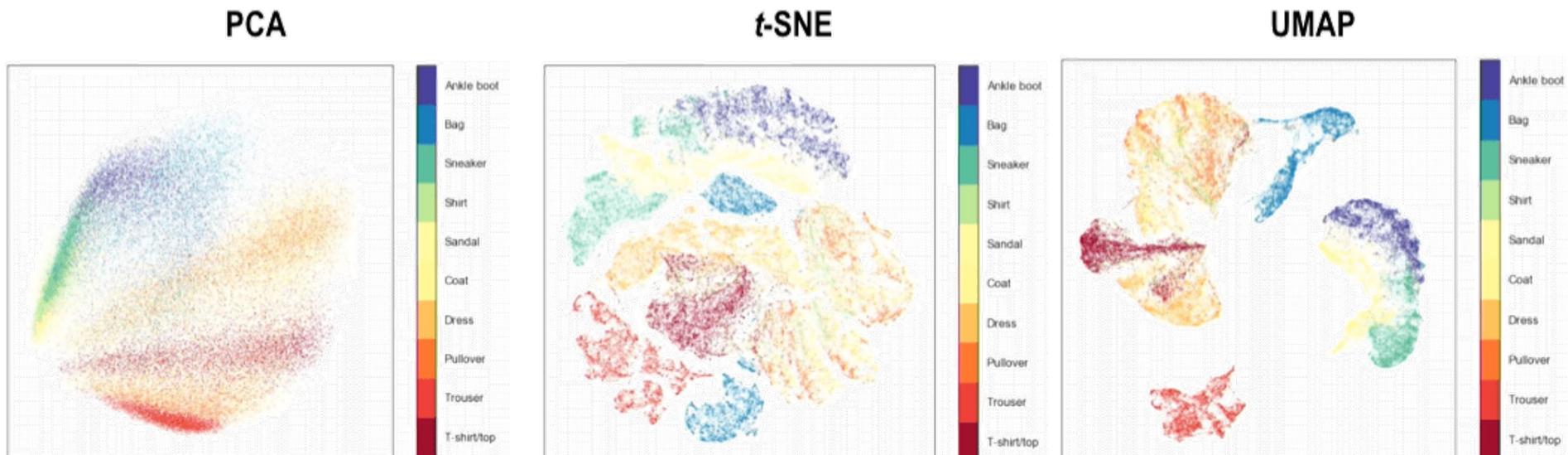


Image from Aizan Fahri: <https://meta.caspershire.net/umap/>

# The Curse of Dimensionality

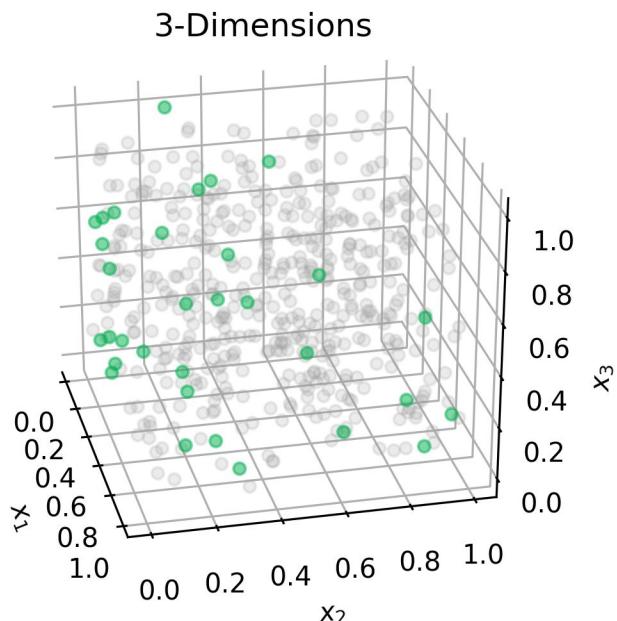
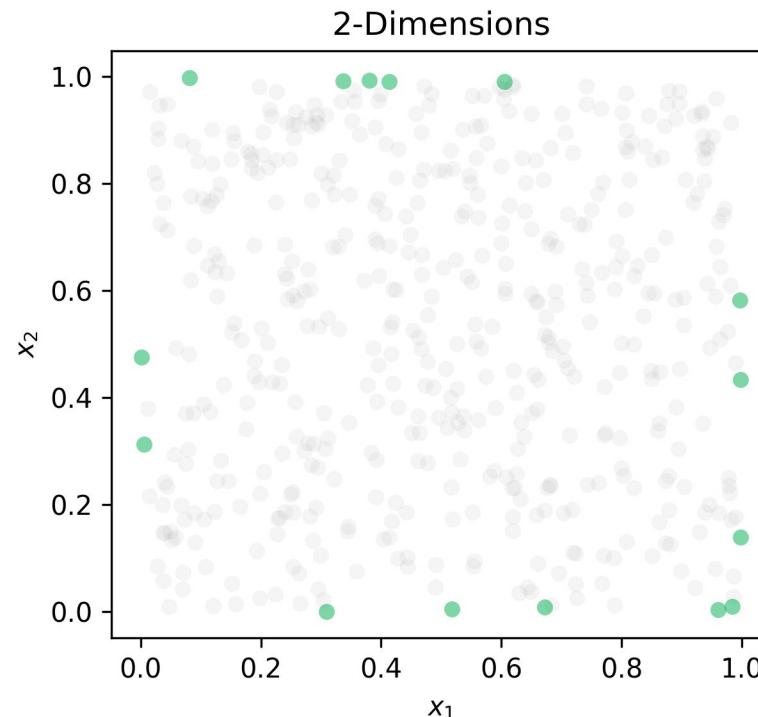
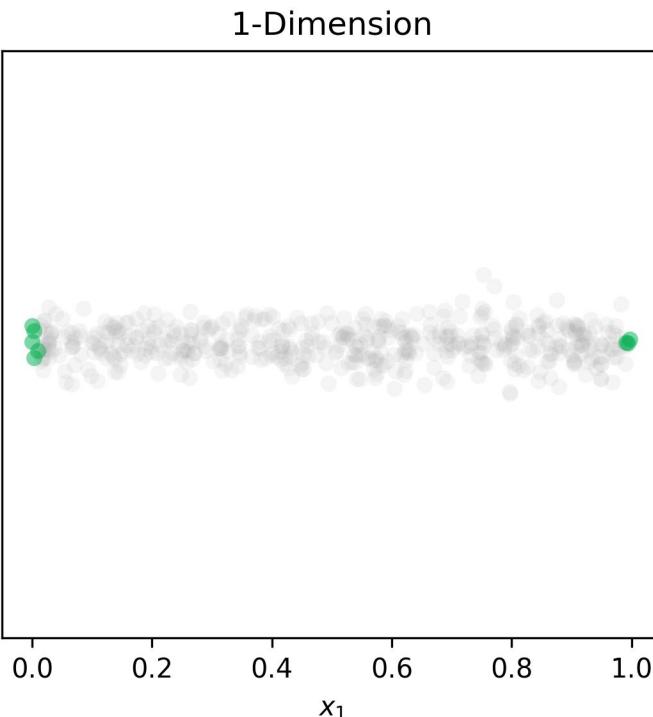
**High dimensions = lots of features**

# 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

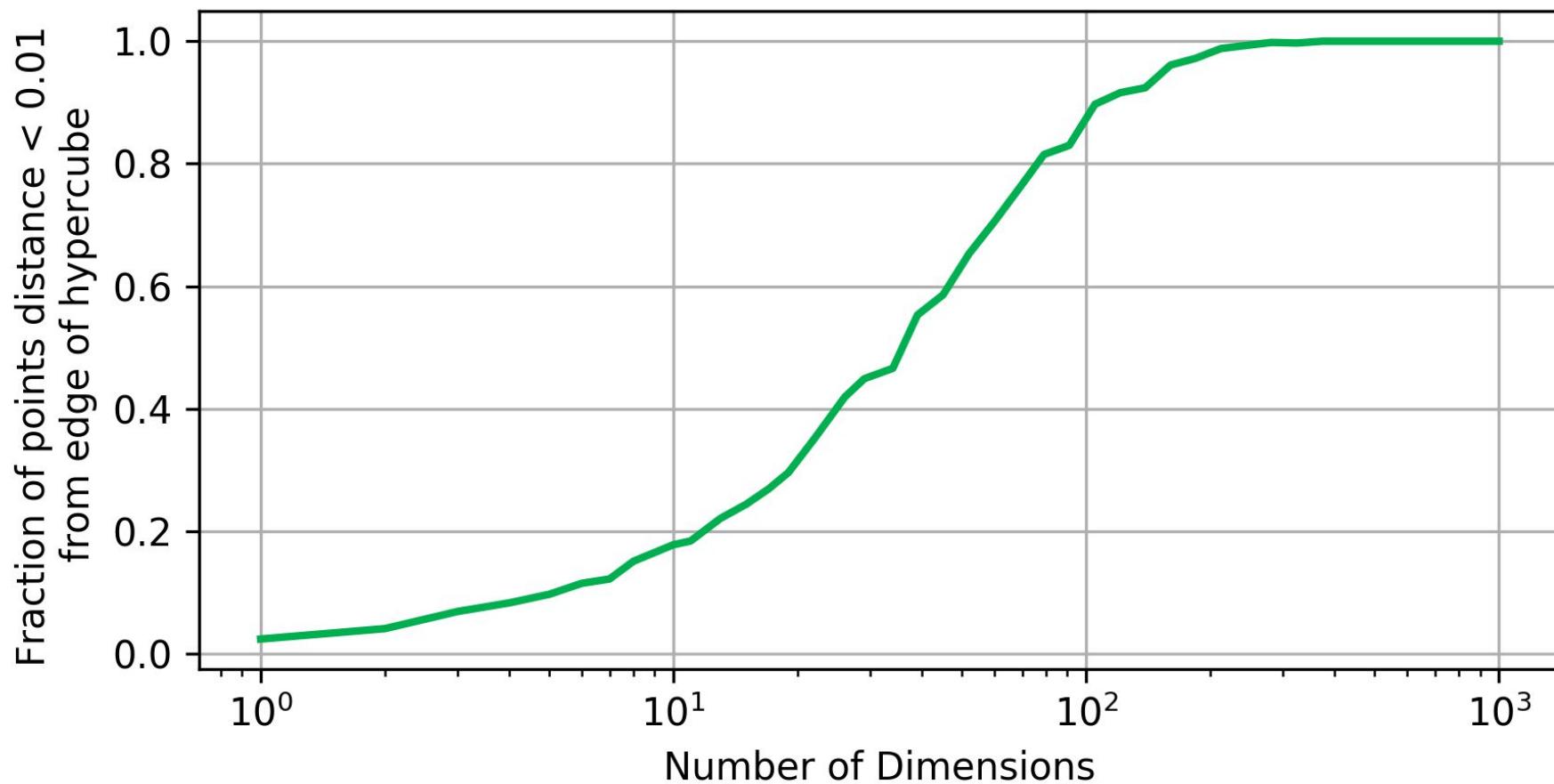
$$\frac{\text{●}}{\text{●} + \text{○}} =$$

0.016

0.030

0.064

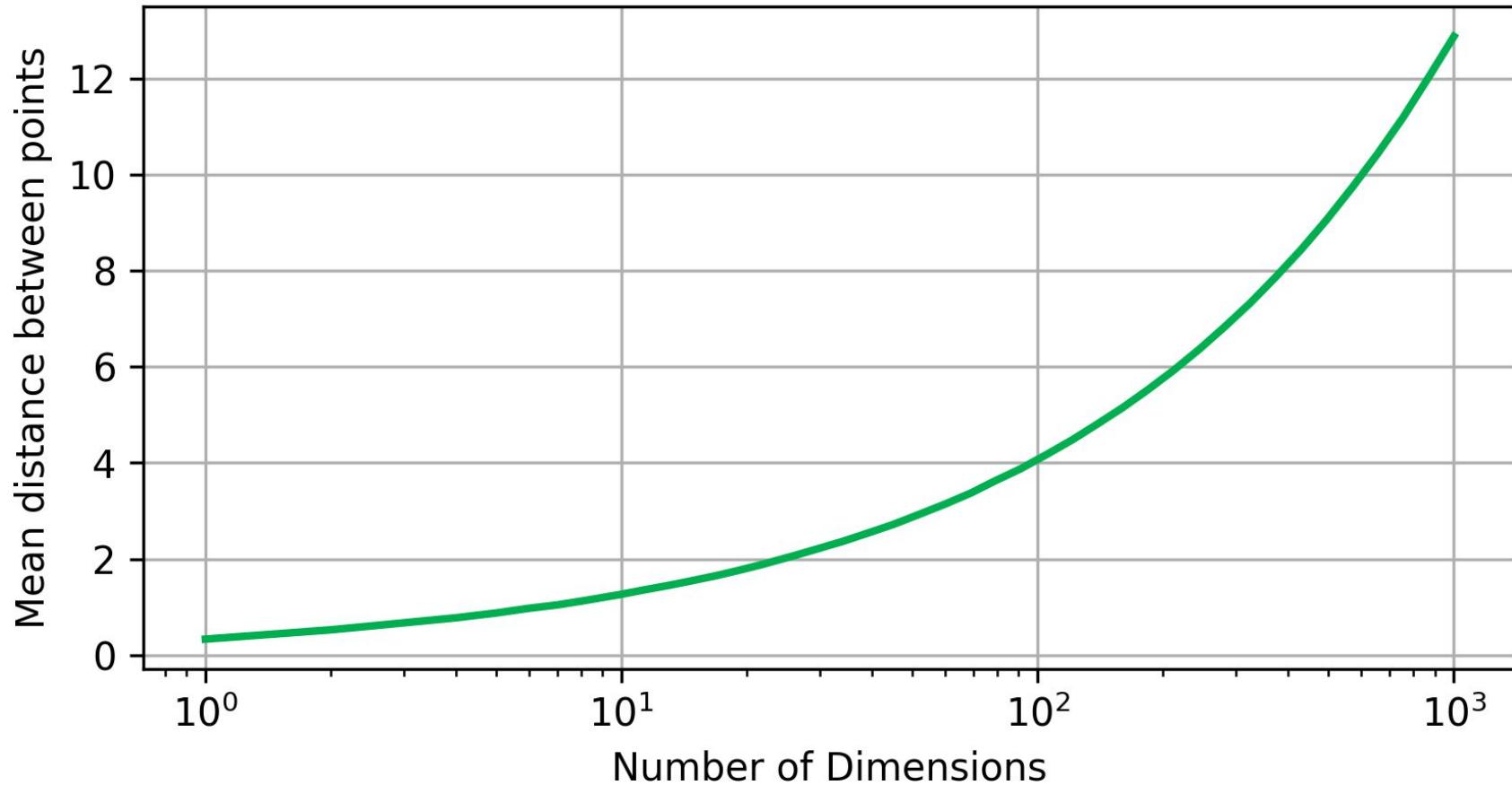
# In high dimensions...



...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

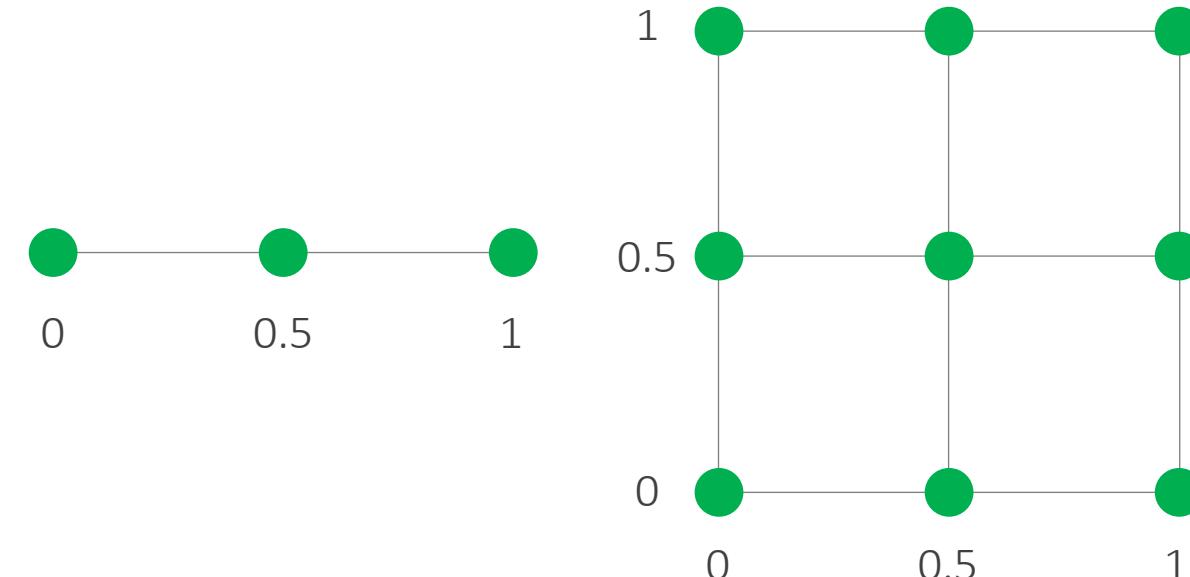
Note: figures constructed using 1,000 random points

# Challenge 2

**Much more data are needed for sampling higher dimensional spaces**

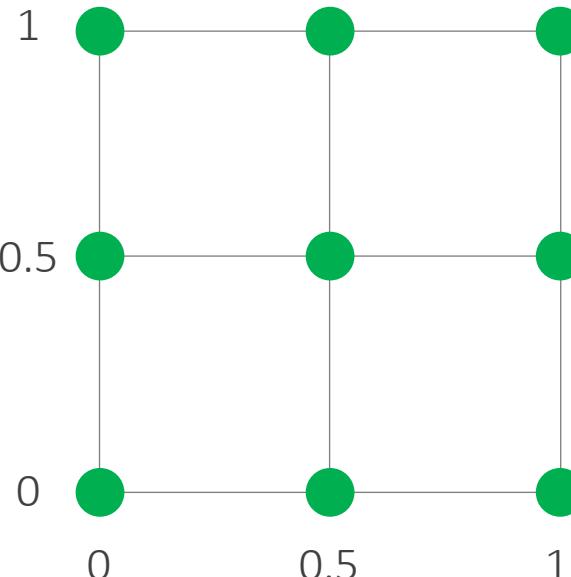
Sample a unit hypercube on a grid spaced at intervals of 0.5

**D = 1**



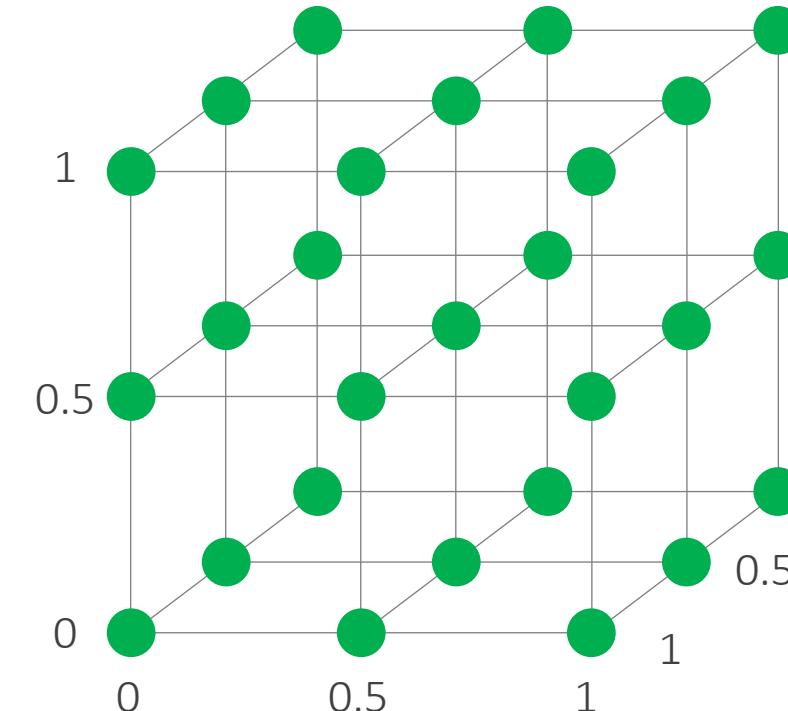
**N = 3**

**D = 2**



**N = 9**

**D = 3**



**N = 27**

**Dim (D)**

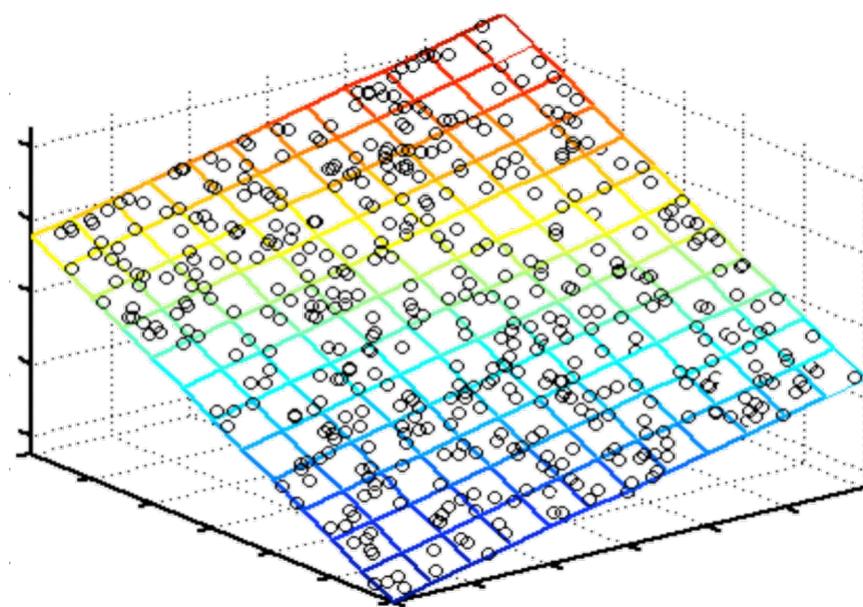
**Samples (N)**

4	81
5	243
10	59,049
100	$5.2 \times 10^{47}$
300	$1.4 \times 10^{143}$

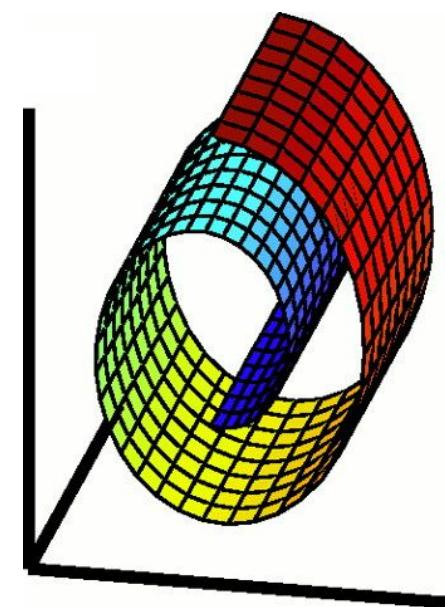
1 million Googles,  
each with 20 exabytes  
of data would only be  
 $10^{25}$  bytes

**...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

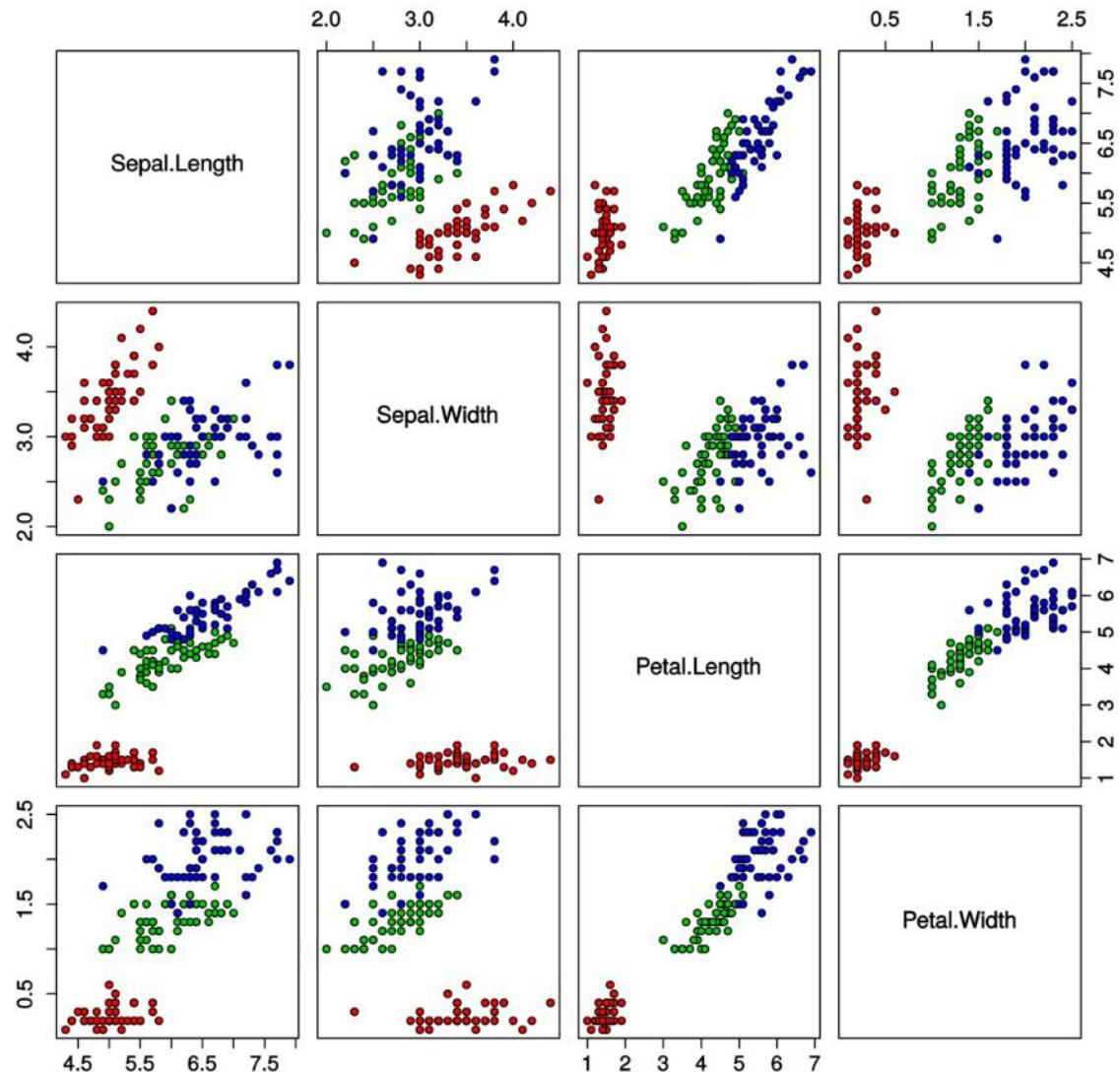
Image Left: Torki, M. and June, Dissertation, 2011. Learning the manifolds of local features and their spatial arrangements. Rutgers University.

Image Right: Roweis, S.T. and Saul, L.K., 2000. Nonlinear dimensionality reduction by locally linear embedding. science, 290(5500), pp.2323-2326.

# Challenge 3

**Visualization of data becomes challenging  
in higher dimensions**

# Scatterplot Matrix



# Parallel Coordinates Plot

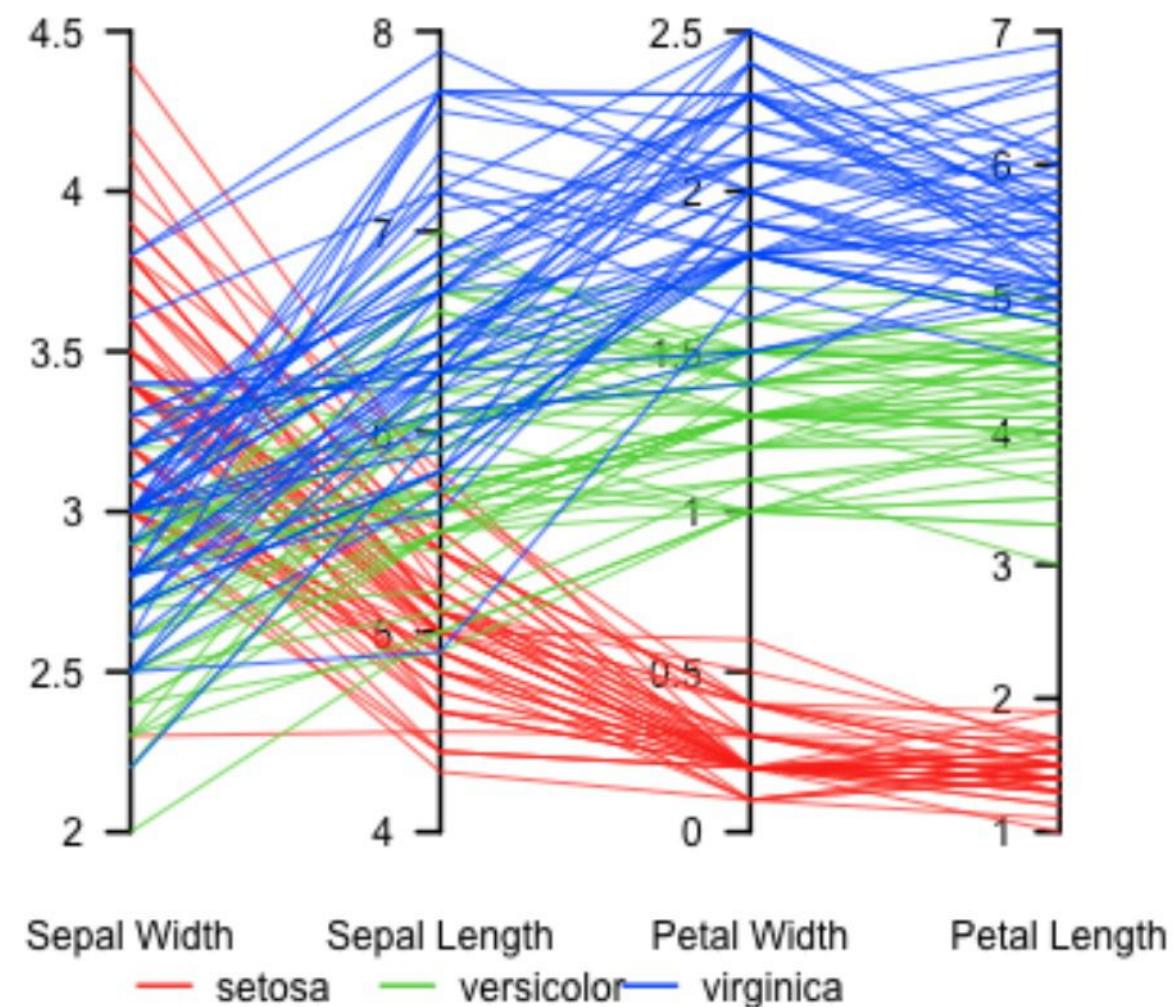


Image: Pezzotti, N., 2019. Dimensionality-Reduction Algorithms for Progressive Visual Analytics.

# Dimensionality Reduction

## Benefits:

Simplified computation

Data are easier to visualize

Reduced redundancy of features

## Drawbacks:

Results in data/information loss

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

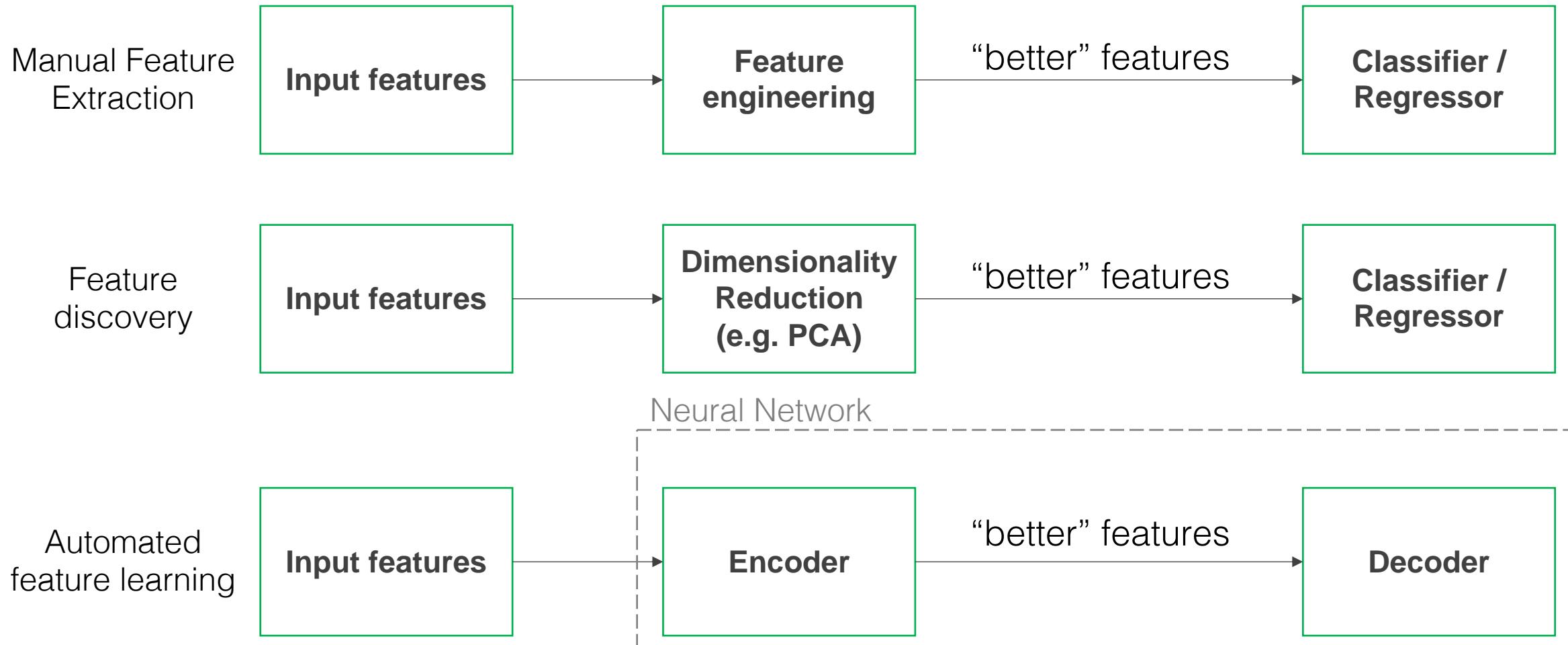
## Feature projection

- **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)

# Another perspective: Representation Learning



# 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

$$\text{Normalized feature} \quad \downarrow \quad \text{Raw, unscaled feature} \quad \downarrow \\ x = \frac{x' - \bar{x}'}{\sigma_{x'}}$$

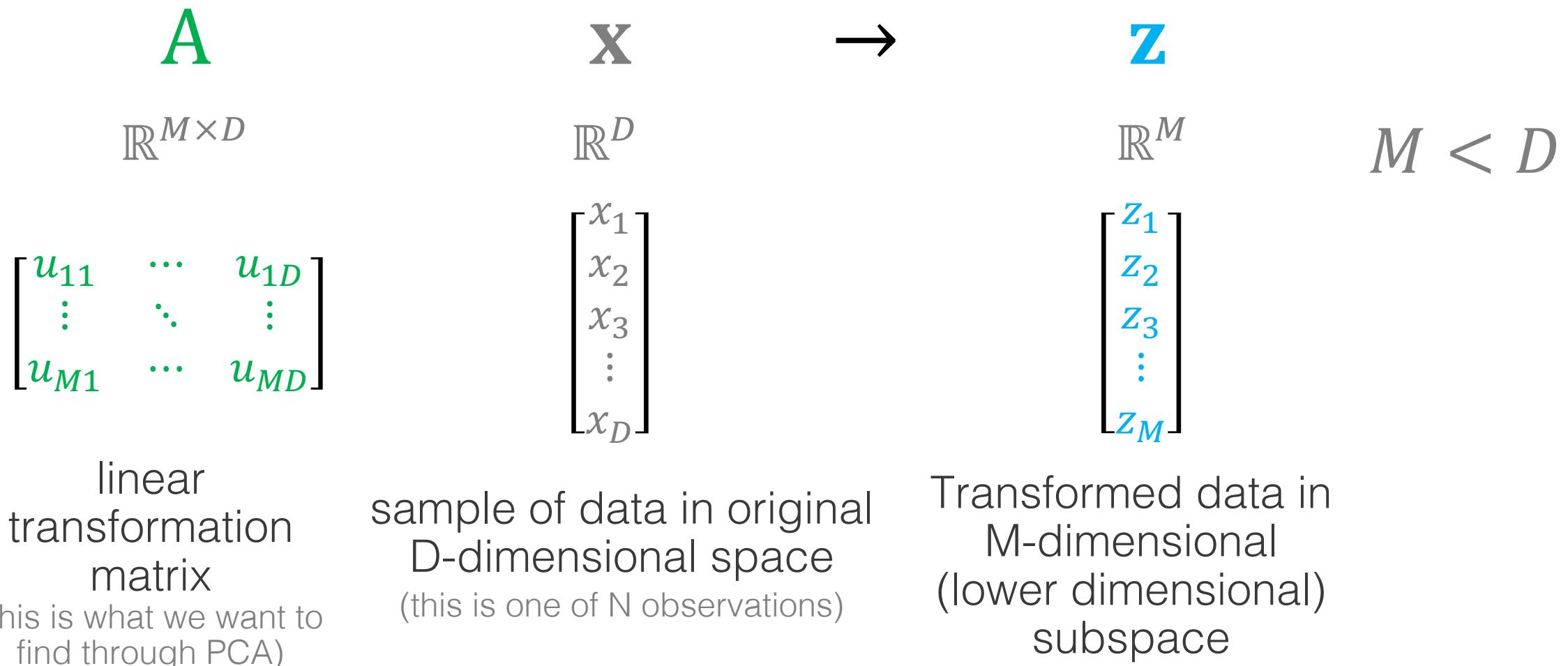
$$\mathbf{X} = \begin{bmatrix} x_{11} & \cdots & x_{1D} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{ND} \end{bmatrix}$$

columns = features  
rows = observations

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

# Principal components analysis

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



# PCA Features are linear combinations of the input features

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

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

⋮

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

# 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  $i^{\text{th}}$  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.  $z_i = \mathbf{u}_1^T \mathbf{x}_i$

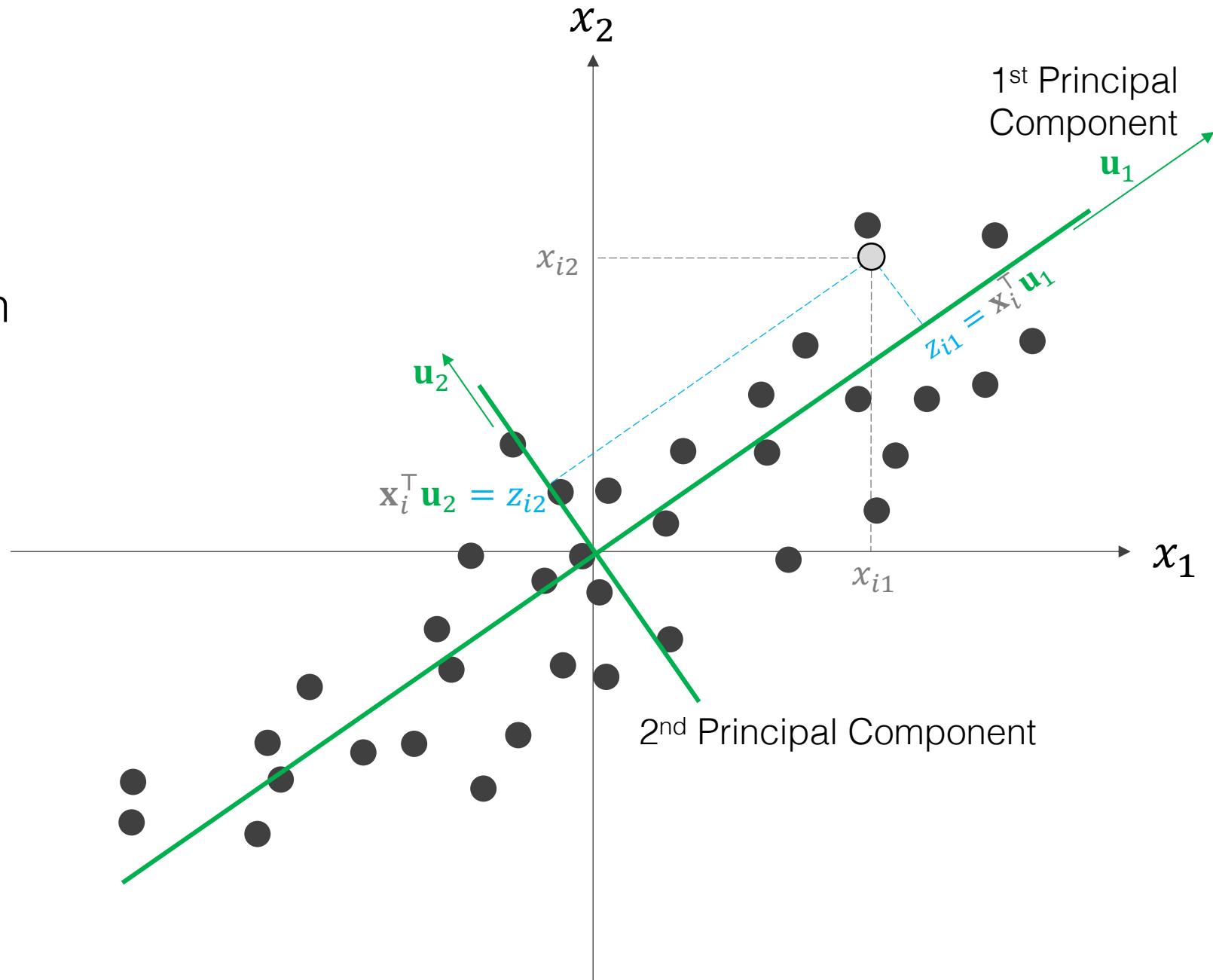
# Principal Components

Maximum variance formulation

Length of a projection  
onto a unit vector:

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

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

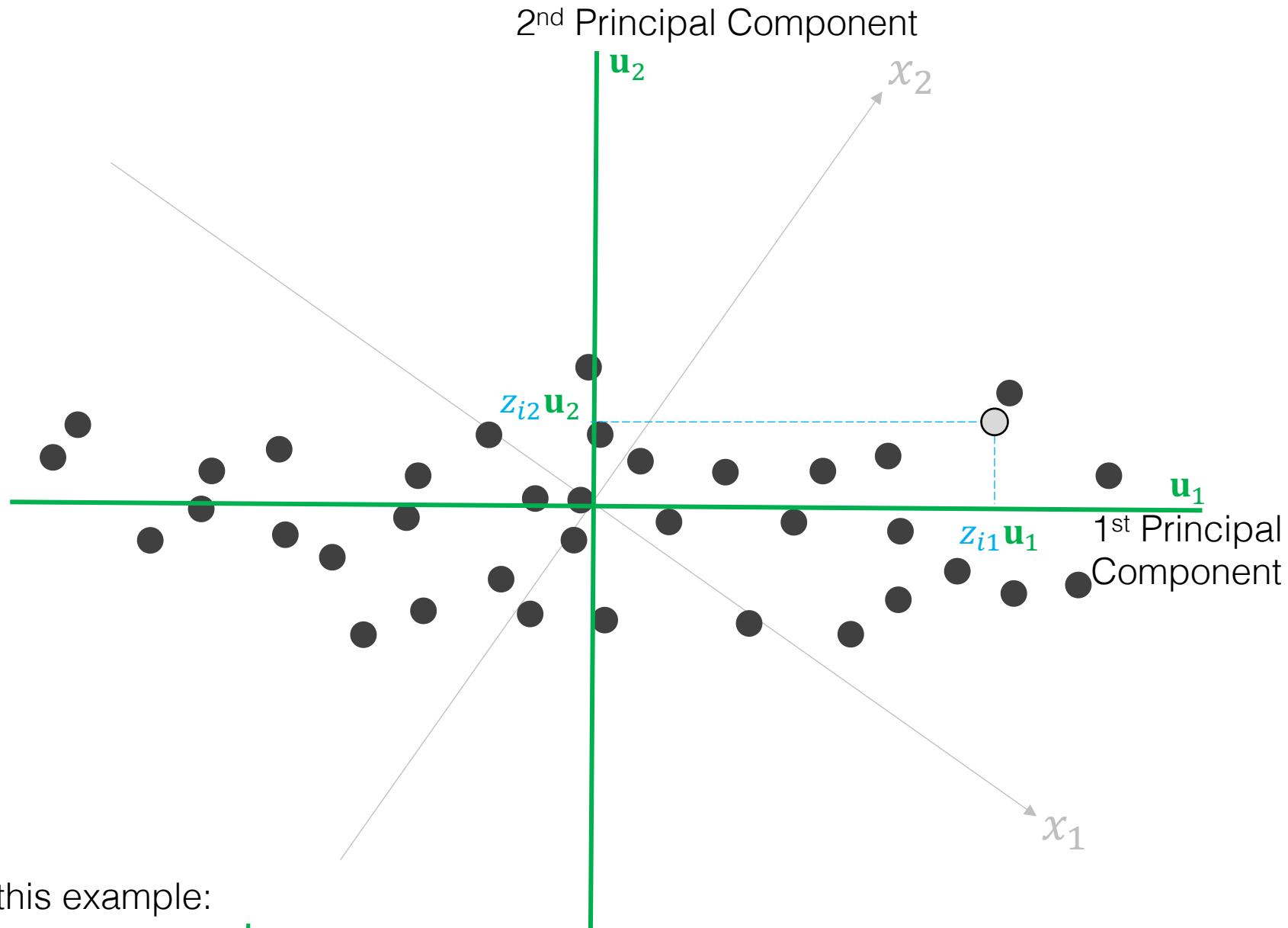


# Reprojected Data onto Principal Components

Any point  $\mathbf{x}_i$  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:

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

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:

$$\sigma_z^2 = \frac{1}{N} \sum_{i=1}^N (z_i - \bar{z})^2$$

[scalar]

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

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

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

# PCA: Compute the variance of the transformed data

We can compute the variance as:

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

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

$$= \mathbf{u}_1^T \Sigma \mathbf{u}_1 \quad \text{Variance of the projected data}$$

Define:

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

Covariance matrix of our data

# Covariance matrix

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

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

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

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

$$\begin{aligned} \Sigma_{jk} &= \frac{1}{N} \sum_{i=1}^N (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k) \\ &= \text{cov}(X_j, X_k) \\ &= E[(X_j - \mu_j)(X_k - \mu_k)] \end{aligned}$$

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

Vector of observation  $i$

$x_{ij}$

Observation index      Predictor index

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

Predictors →  
Observations ↓

Mean of each predictor  
 $\downarrow$   
 If  $\bar{x}_j = 0$  for all  $j$

This will be the case IF the data are standardized

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

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

$$= E[X_j X_k]$$

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

# 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:

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

# Covariance matrix properties

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

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

All eigenvalues are positive

Eigenvectors are orthogonal

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

# PCA: Maximize the variance

We want to **maximize variance**  $\sigma_z^2 = \mathbf{u}_1^T \Sigma \mathbf{u}_1$   
subject to  $\mathbf{u}_1^T \mathbf{u}_1 = 1$  (unit vectors)

We can use **Lagrange multipliers**:

Maximize  $f(x)$

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

subject to the constraint  $g(x)$

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

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

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

We take the derivative and set it equal to zero

# PCA

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

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

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

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

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

$$\sigma_z^2 = \mathbf{u}_1^T \Sigma \mathbf{u}_1$$

To do that this must be true: (shown on last slide)

$$\Sigma \mathbf{u}_1 = \lambda \mathbf{u}_1$$

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

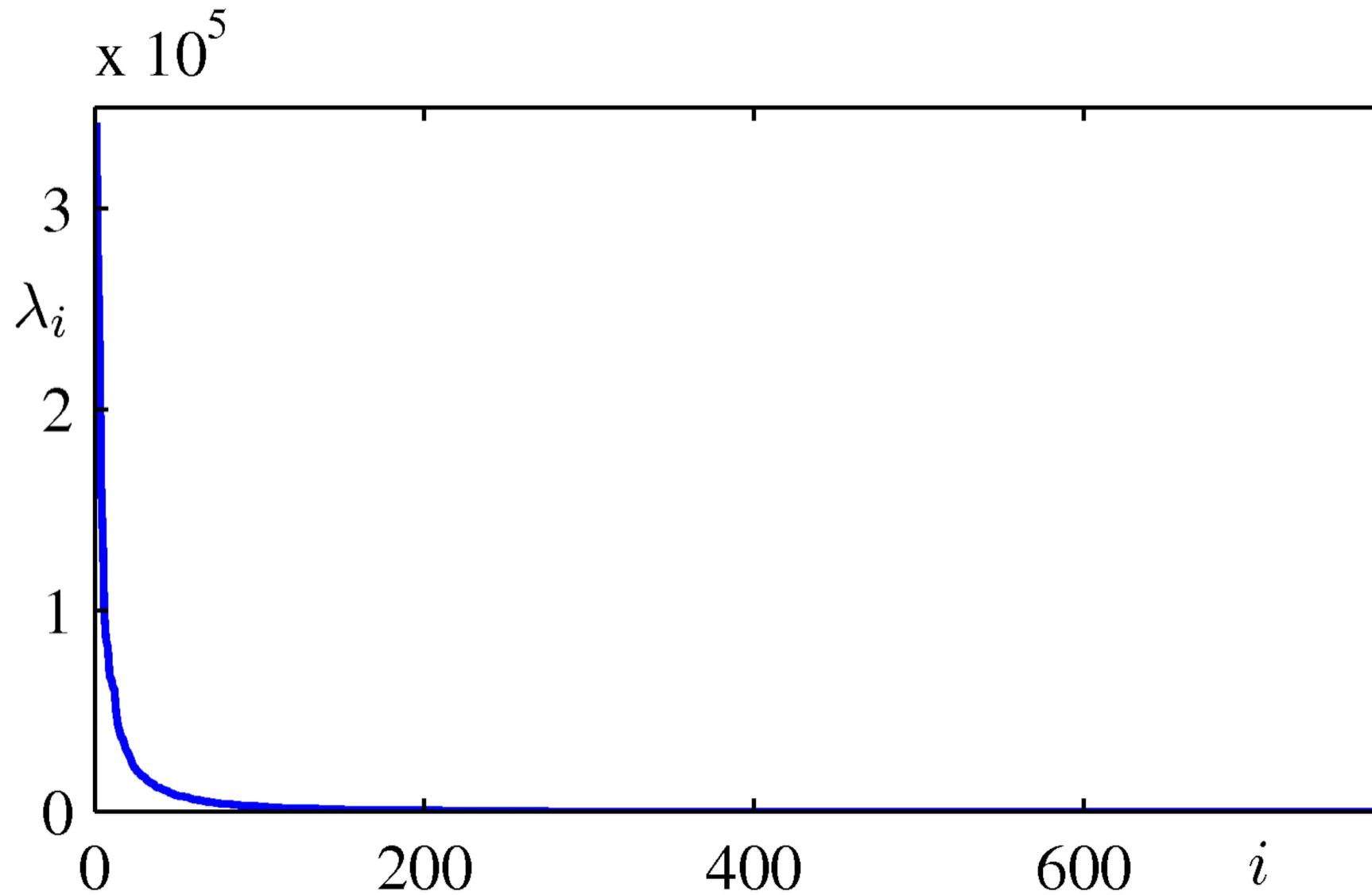
$$\text{The fraction of variance explained} = \frac{\sum_{i=1}^M \lambda_i}{\sum_{i=1}^D \lambda_i}$$

$M$  = 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

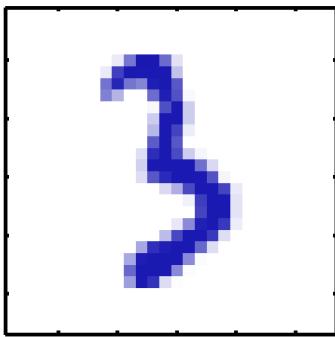
# Eigenvalues by principal component $i$



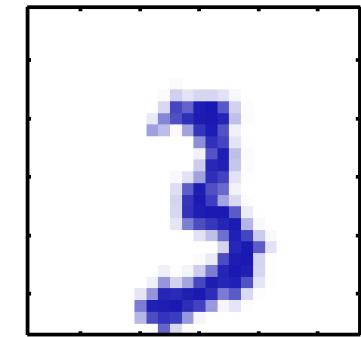
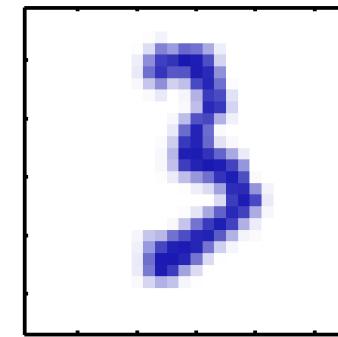
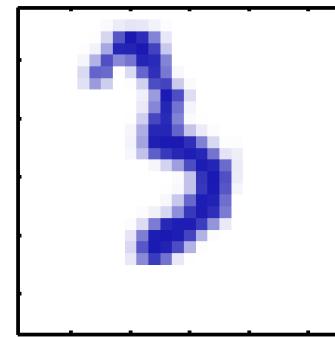
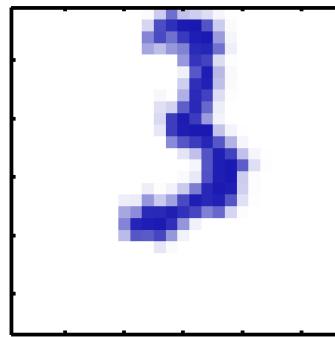
Bishop, Pattern Recognition, 2006

# Example: translated digits

Original digit



Translated digits



**Types of translation:**

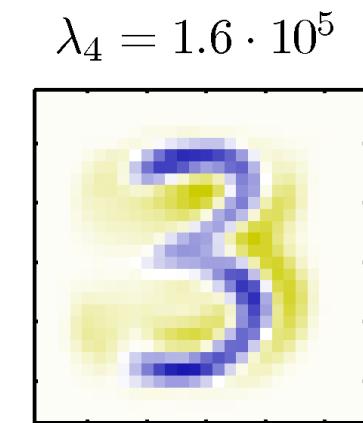
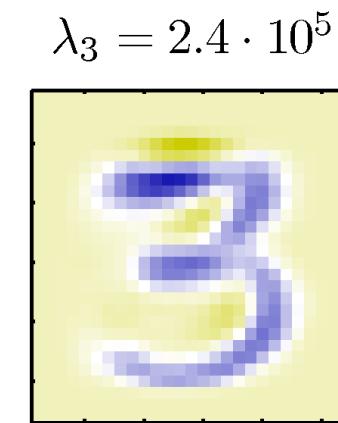
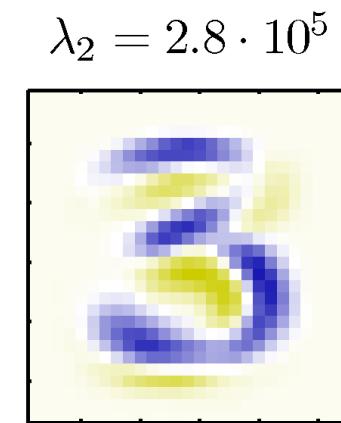
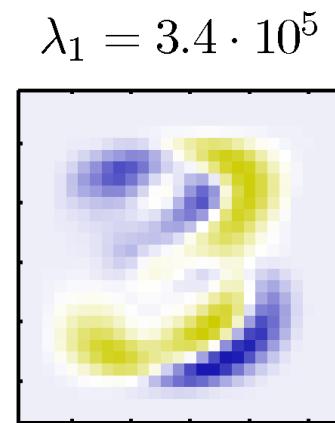
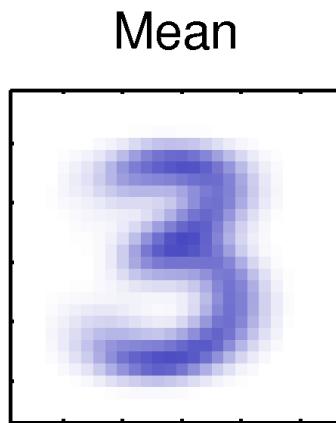
1. Horizontal translation
2. Vertical translation
3. Rotation

Size: 100 x 100 pixels

# Example: translated digits

PCA identifies a new way to represent the data

Examples of first four principle component eigenvectors and eigenvalues:



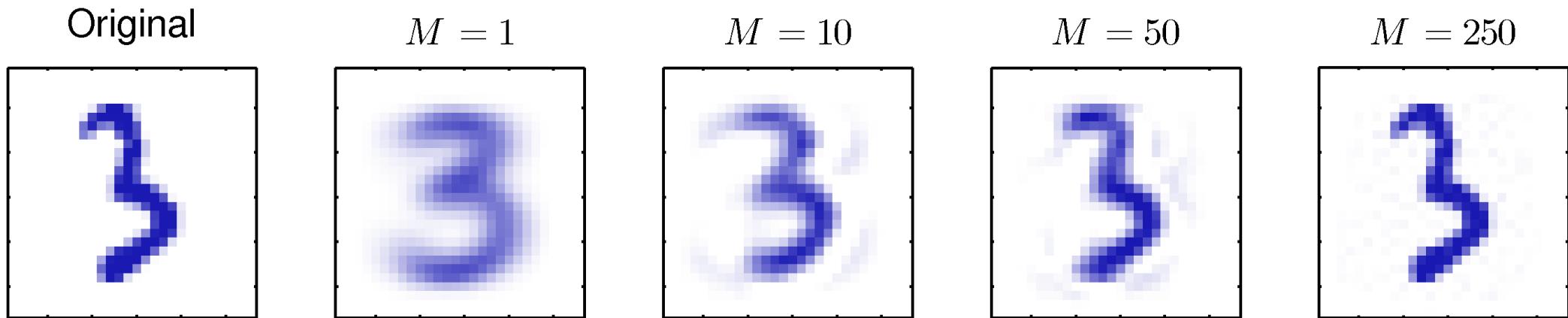
## Latent Space DEMO

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:



# Extracting principal components

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

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

**2** Determine the principal components

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

Eigenvectors in descending order of their eigenvalues are the principal components

**3** Project the data features on the principal components

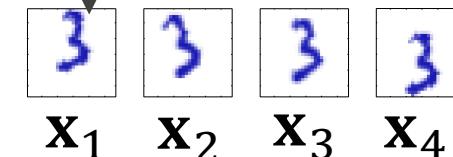
**4** Keep the top  $M$  principal components to reduce into a lower dimension

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

Each observation as a vector:  
 $\mathbf{x}_i \quad i = 1, \dots, D$

[ $N \times D$ ]

Each pixel represents a feature

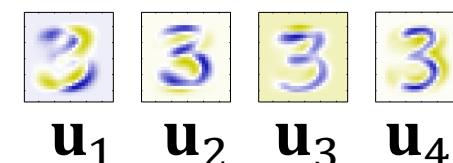


[ $D \times 1$ ]

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

$\lambda_i$  eigenvalues (how much of the variance is explained)  
 $i = 1, \dots, D$

[ $D \times 1$ ]



[scalar]

$$z_{ij} = \mathbf{u}_j^T \mathbf{x}_i \quad j = 1, \dots, D \\ i = 1, \dots, N$$

[scalar]

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

[ $D \times M$ ]

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

$$\mathbf{u}_1 \cdot \mathbf{x}_1 = z_{11} \\ \text{---} \cdot \text{---} = 2.43$$

[ $D \times 1$ ] [scalar]

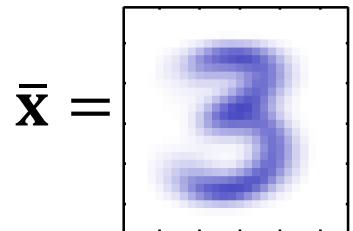
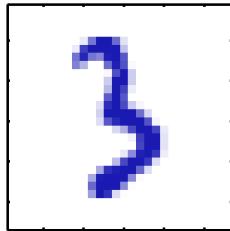
Images from Bishop, Pattern Recognition, 2006

# 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  $i^{\text{th}}$  observation:  $\mathbf{x}_i =$



$M = 1$        $\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + z_{i1} \cdot \mathbf{u}_1$

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

$=$

$M = 250$      $\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + z_{i1} \cdot \mathbf{u}_1 + z_{i2} \cdot \mathbf{u}_2 + \sum_{j=3}^{250} z_{ij} \mathbf{u}_j$

$=$

$M = 10,000$     $\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + z_{i1} \cdot \mathbf{u}_1 + z_{i2} \cdot \mathbf{u}_2 + \sum_{j=3}^{10,000} z_{ij} \mathbf{u}_j$

$=$

Images from Bishop, Pattern Recognition, 2006

(perfect reconstruction)

# Why PCA?

- Dimensionality reduction
- Feature extraction
- Data visualization
- Reducing feature correlation (whitening)
- (Lossy data compression)

## 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

# Autoencoders

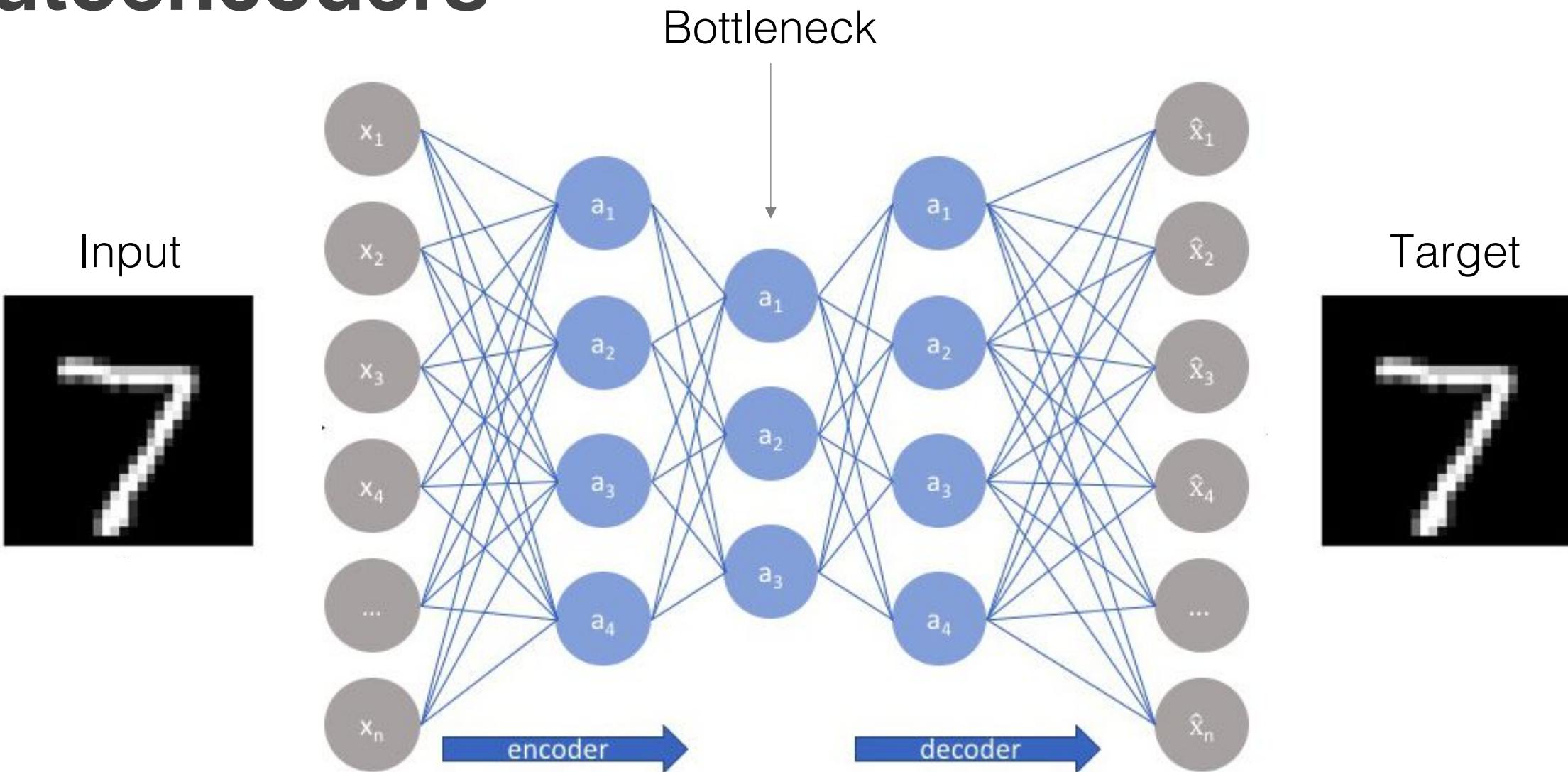
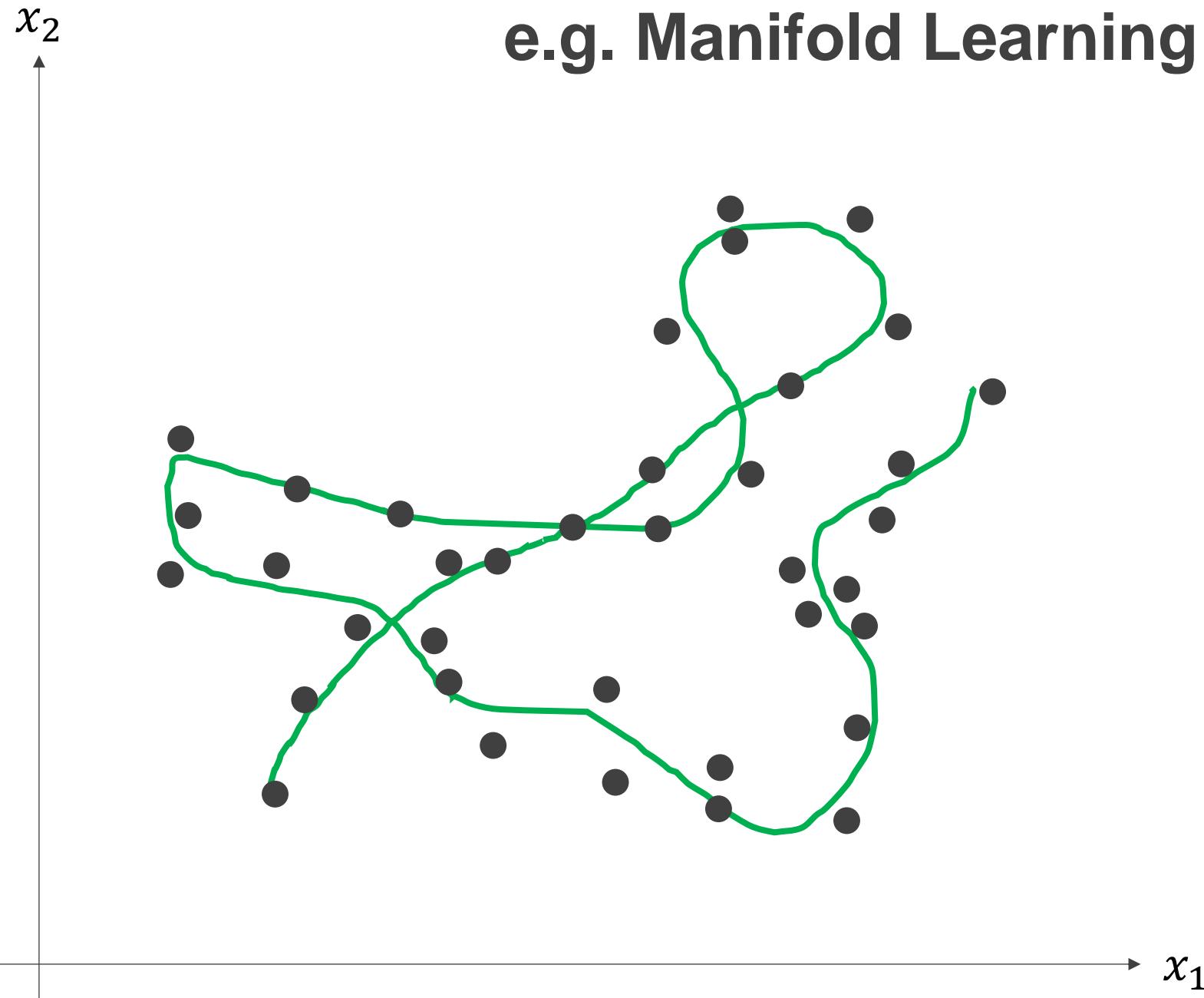


Image from: <https://www.jeremyjordan.me/autoencoders/>

# Other dimensionality reduction techniques

- Kernel PCA
- Random projections
- Multidimensional scaling
- Locality sensitive hashing
- Isomap
- t-SNE
- UMAP

e.g. Manifold Learning



# Appendix: Eigenvector resources

# Eigenanalysis and PCA

Eigenvector Demo:

<http://setosa.io/ev/eigenvectors-and-eigenvalues/>

PCA Demo:

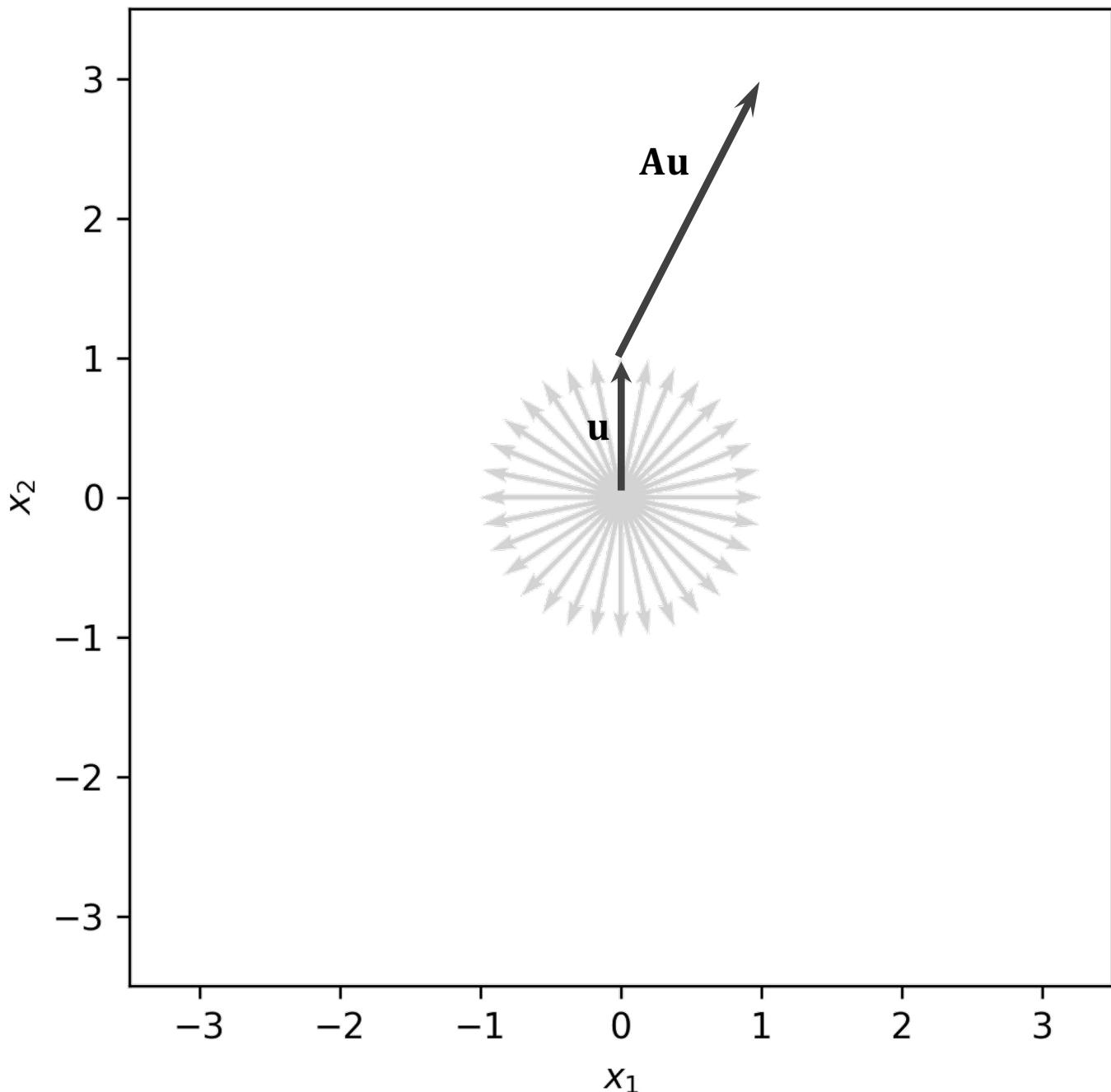
<http://setosa.io/ev/principal-component-analysis/>

# Eigenvalues/Eigenvectors

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$$

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

$$\mathbf{u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{A}\mathbf{u} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



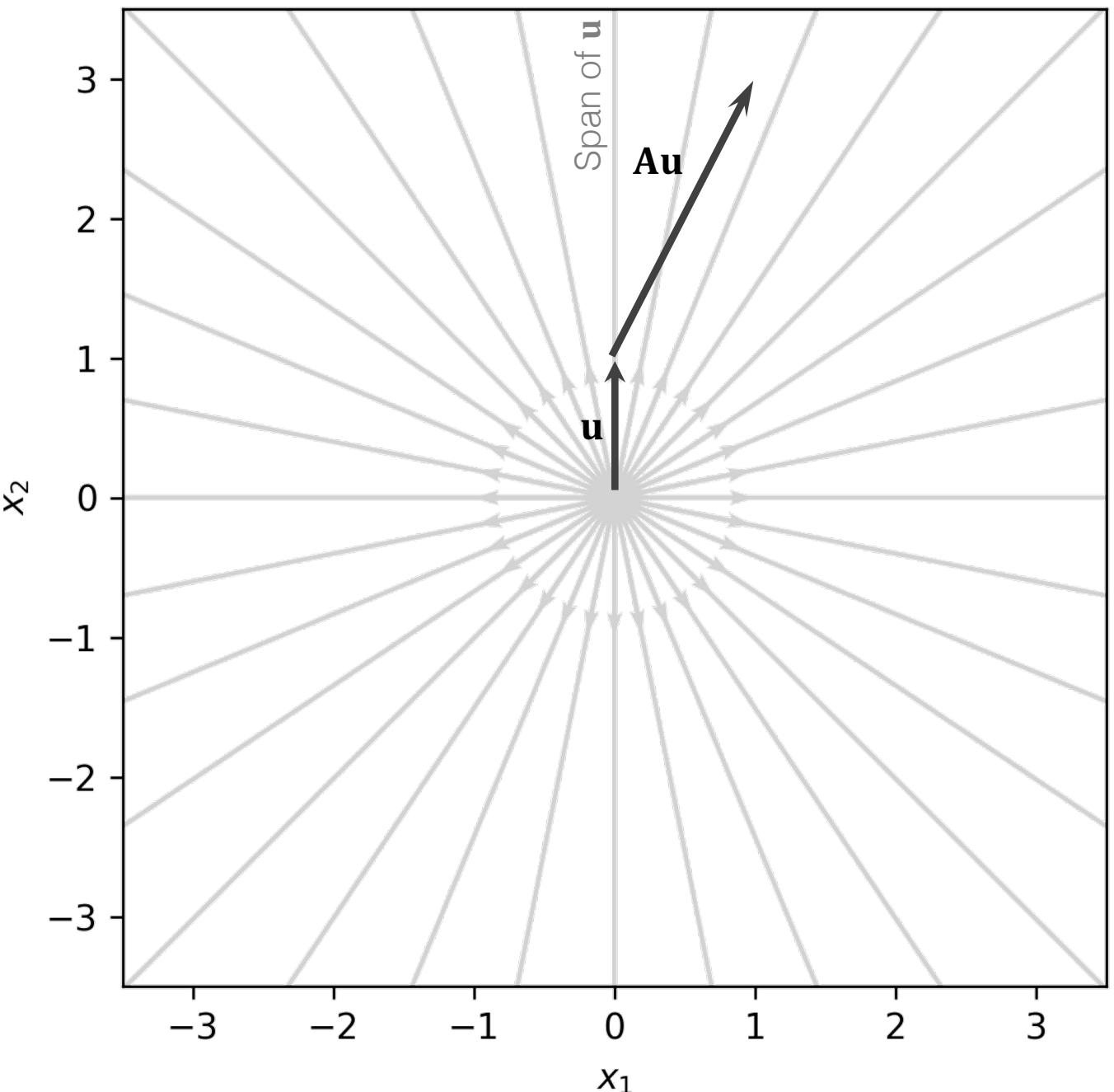
# Eigenvalues/Eigenvectors

$$\mathbf{Au} = \lambda \mathbf{u}$$

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

$$\mathbf{u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

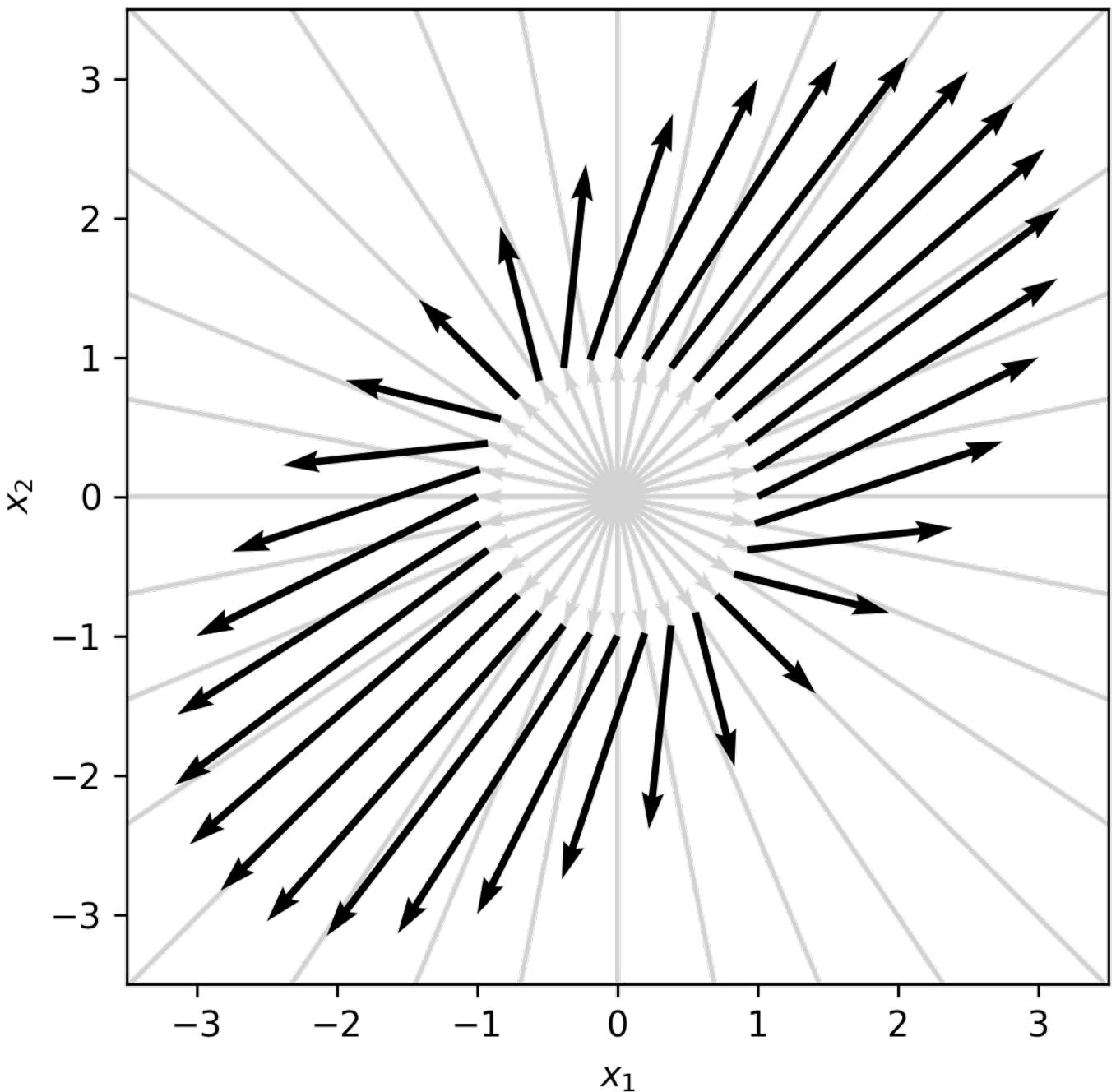
$$\begin{aligned}\mathbf{Au} &= \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 \\ 1 \end{bmatrix}\end{aligned}$$



# Eigenvalues/Eigenvectors

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$$

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



# Eigenvalues/Eigenvectors

$$\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$$

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

$$\mathbf{u}_1 = \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix} \quad \lambda_1 = 3$$

$$\mathbf{A}\mathbf{u}_1 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}$$

$$= \begin{bmatrix} 2.12 \\ 2.12 \end{bmatrix}$$

$$= \lambda_1 \mathbf{u}_1 = 3 \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}$$

