Unsupervised Learning 2
Representation Learning

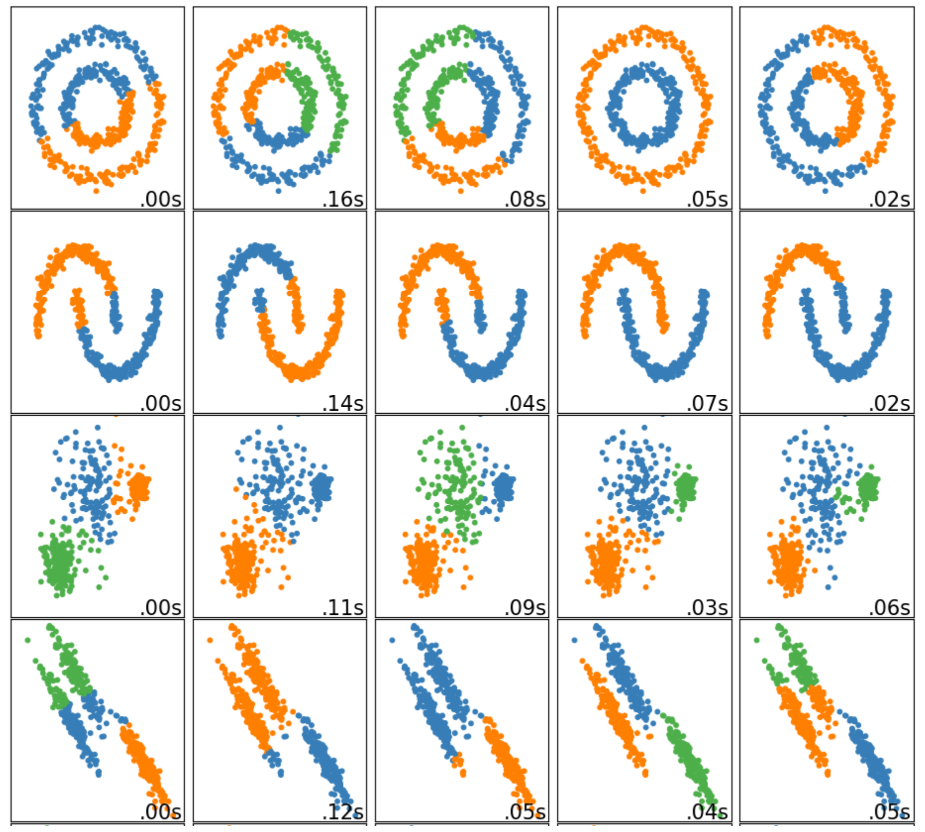
DSCC 251/451: Machine Learning with Limited Data

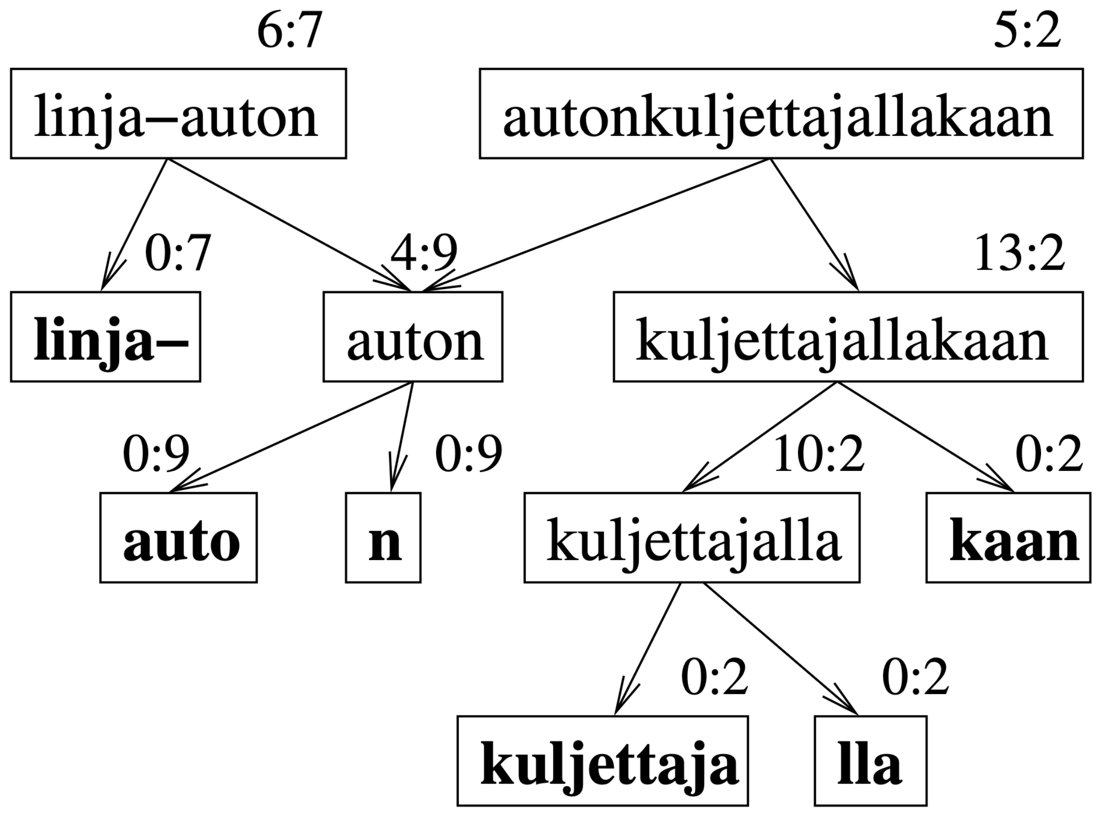
C.M. Downey

Spring 2026

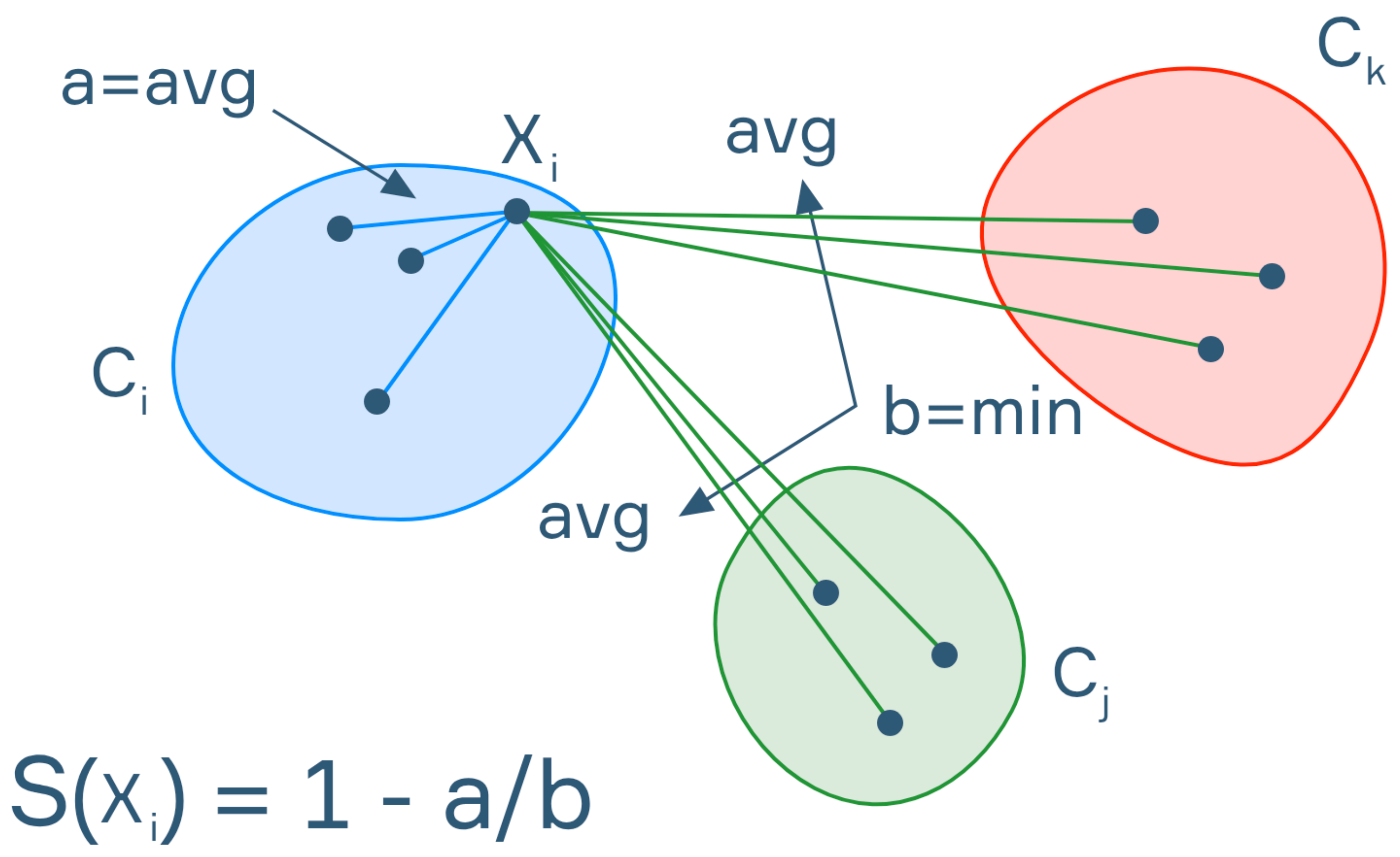
Recap and Roadmap

Recap





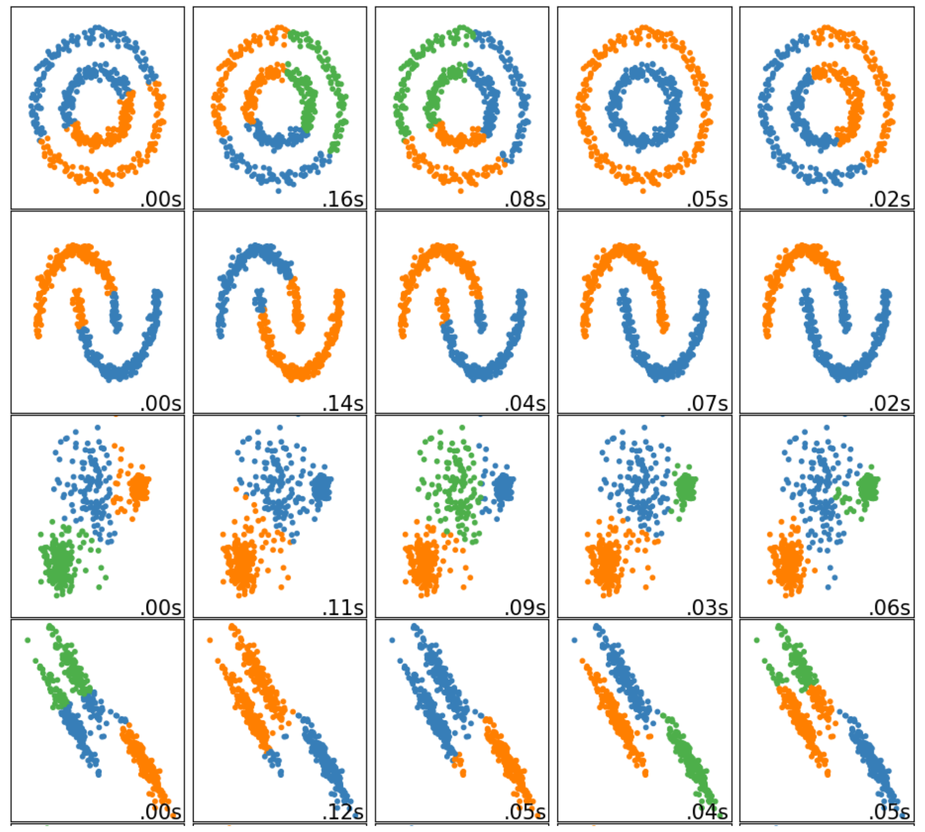
6:7
linja–auton
5:2
autonkuljettajallakaan
0:7
4:9
13:2
linja–
auton
kuljettajallakaan
0:9
0:9
10:2
0:2
auto
n
kuljettajalla
kaan
0:2
0:2
kuljettaja
lla

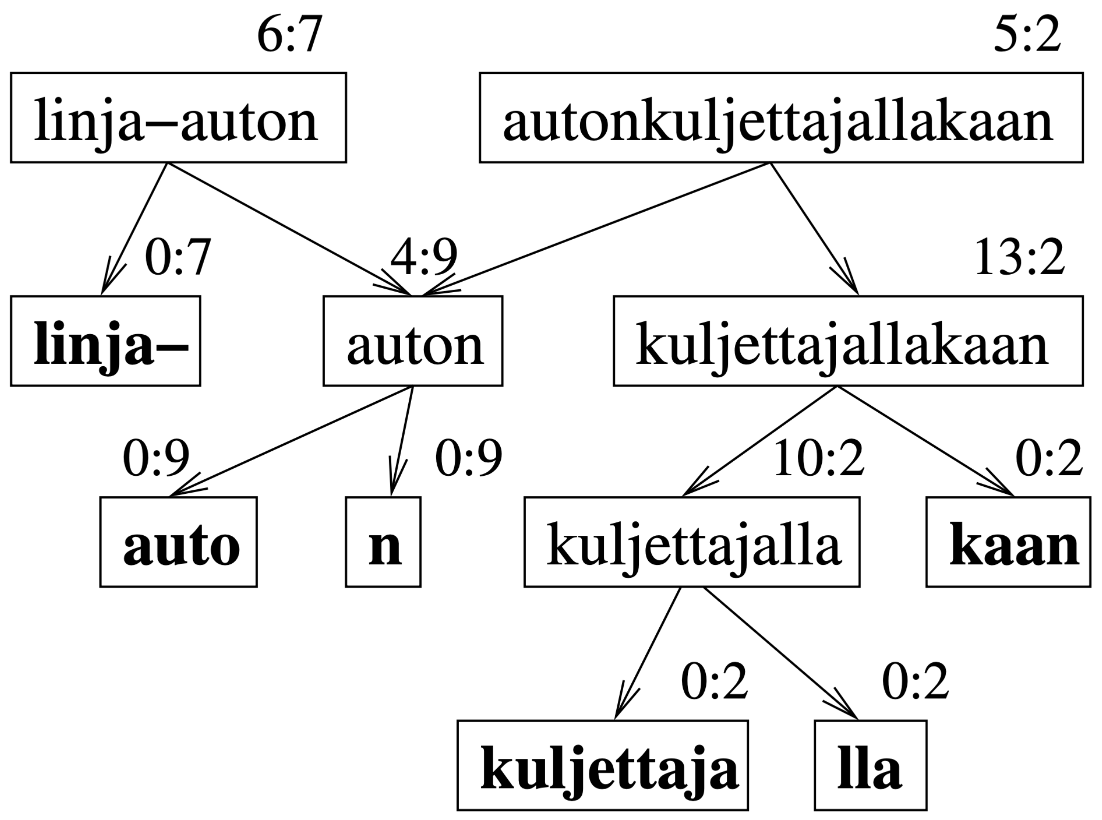


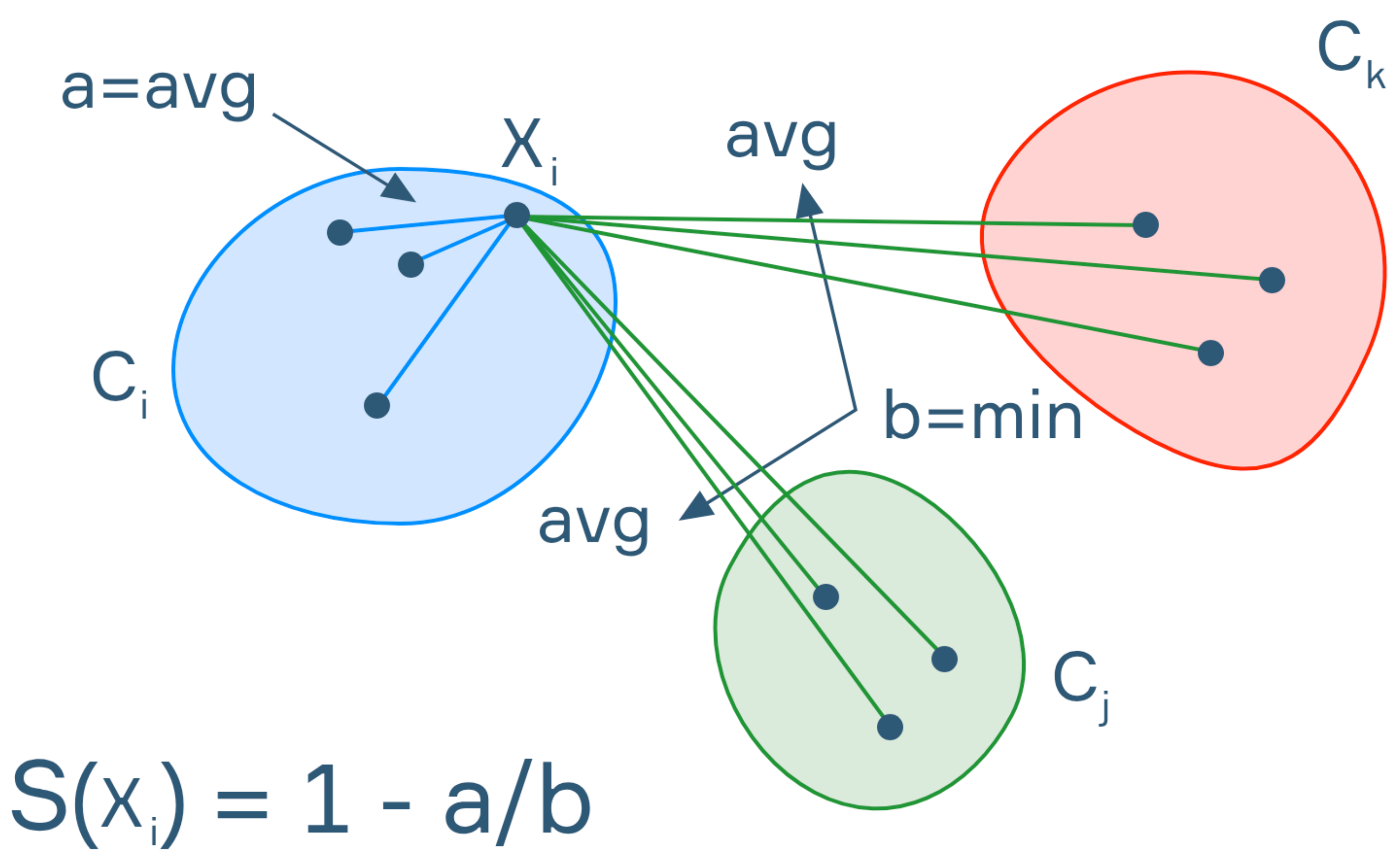
a=avg
X_i
avg
C_k
C_i
b=min
avg
C_j
$S(x_i) = 1 - a/b$

Recap

- Unsupervised Learning = **discovering structure in (unlabeled) data**

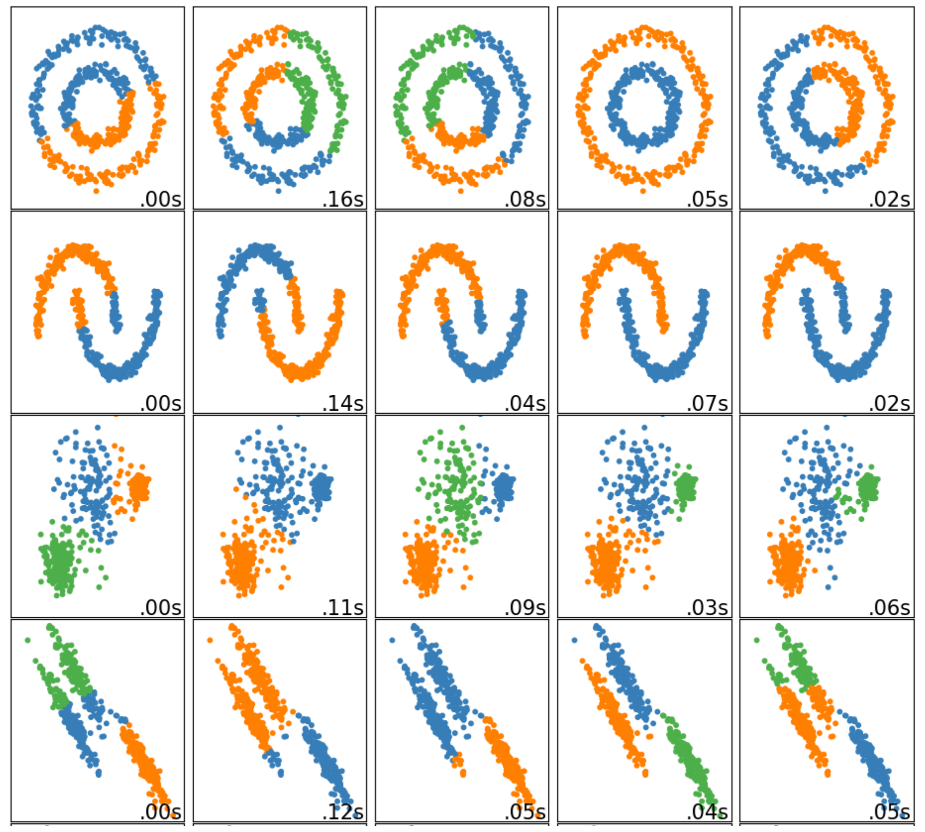


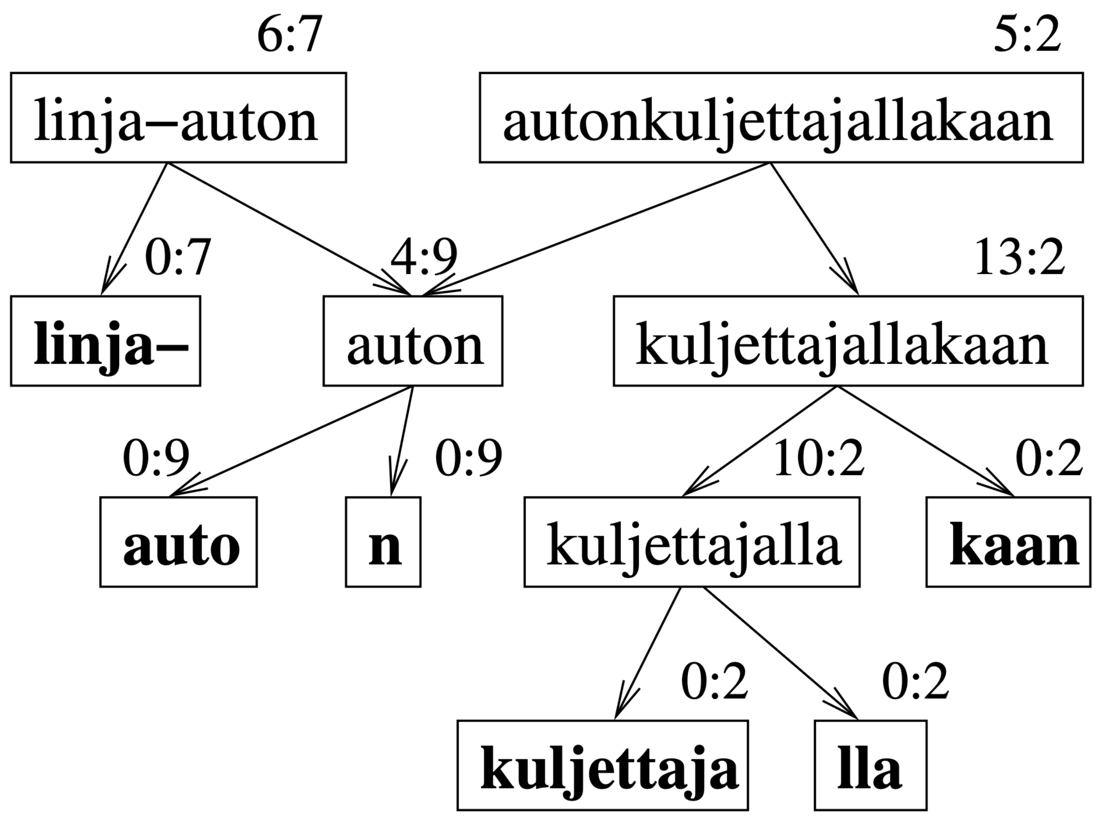


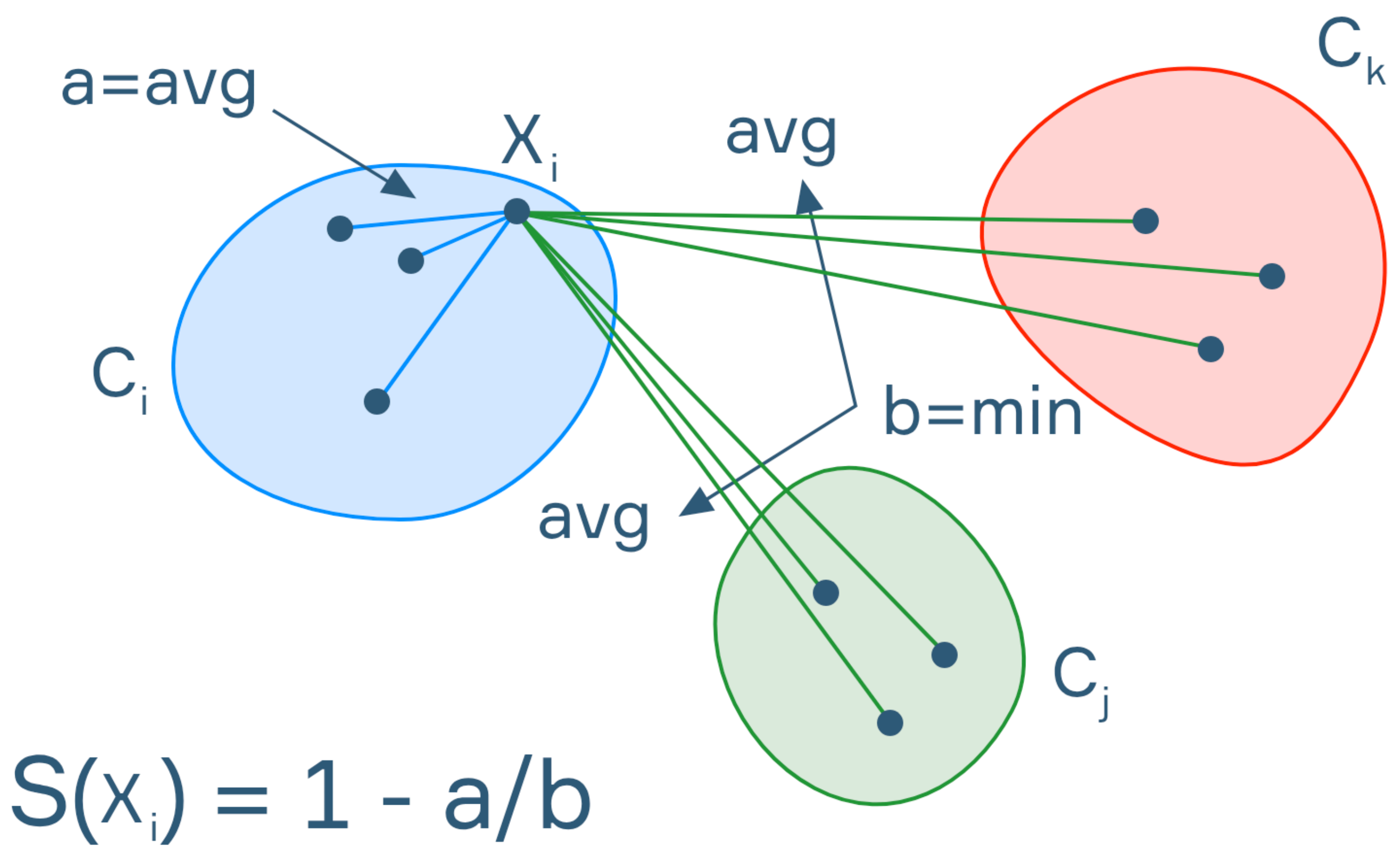


Recap

- Unsupervised Learning = **discovering structure in (unlabeled) data**
- Usually have to optimize a **surrogate objective**, since you can't optimize a supervised one
    - Hope/hypothesis: this **correlates** with something you care about

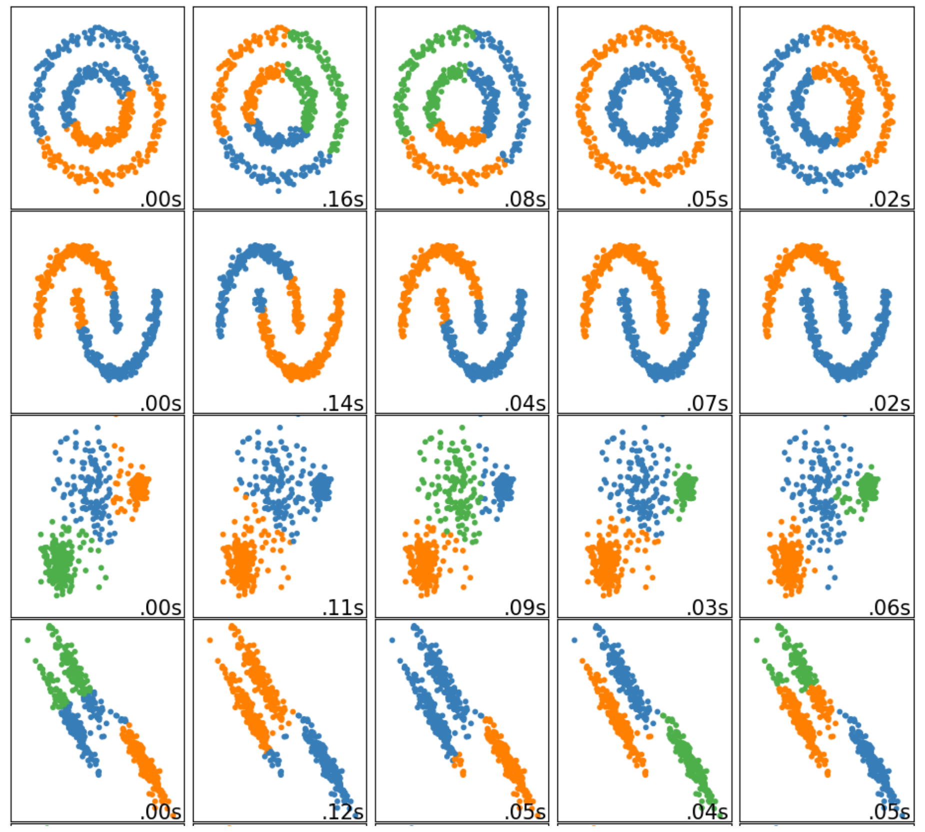


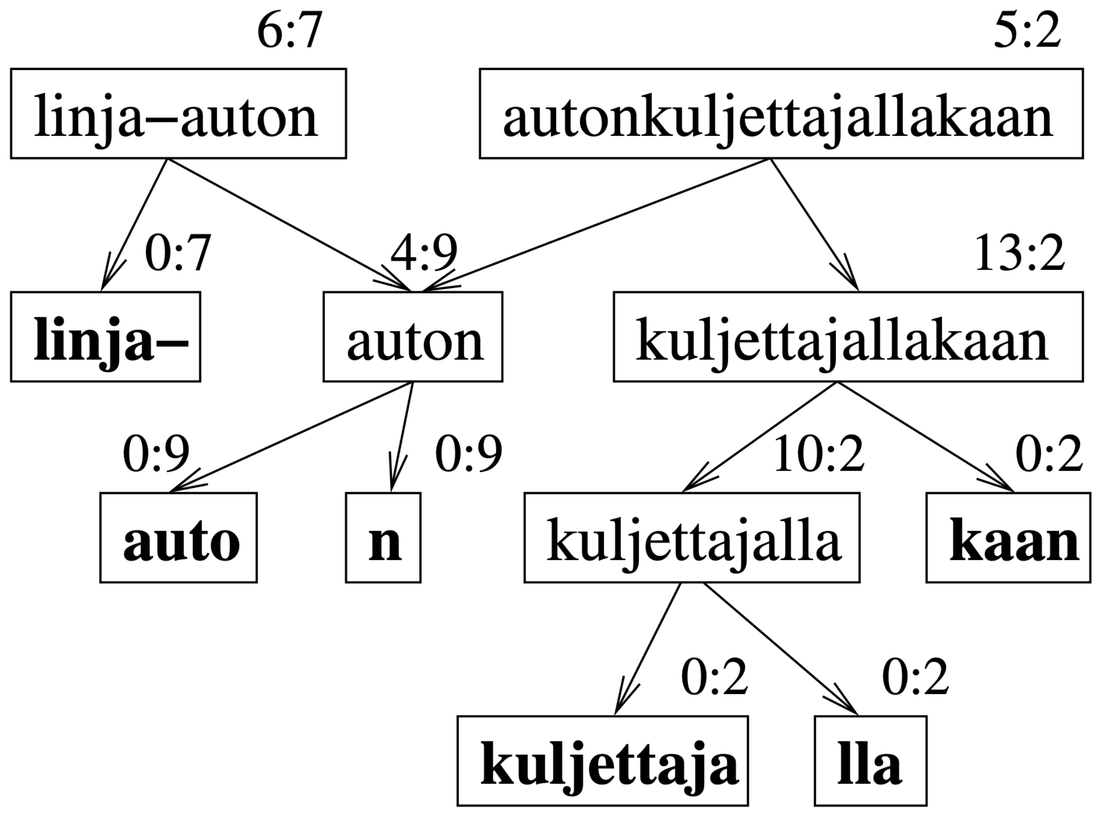


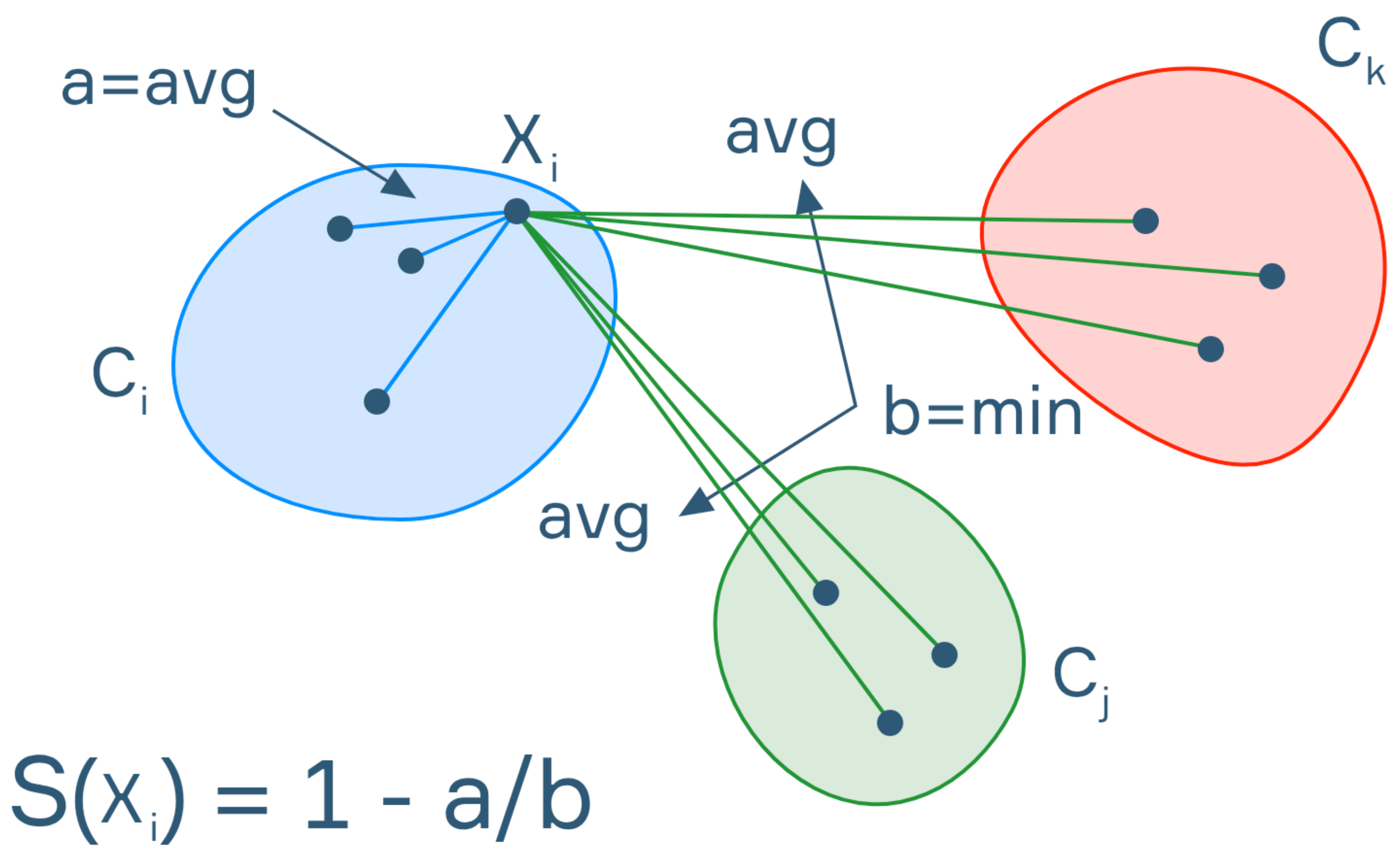


Recap

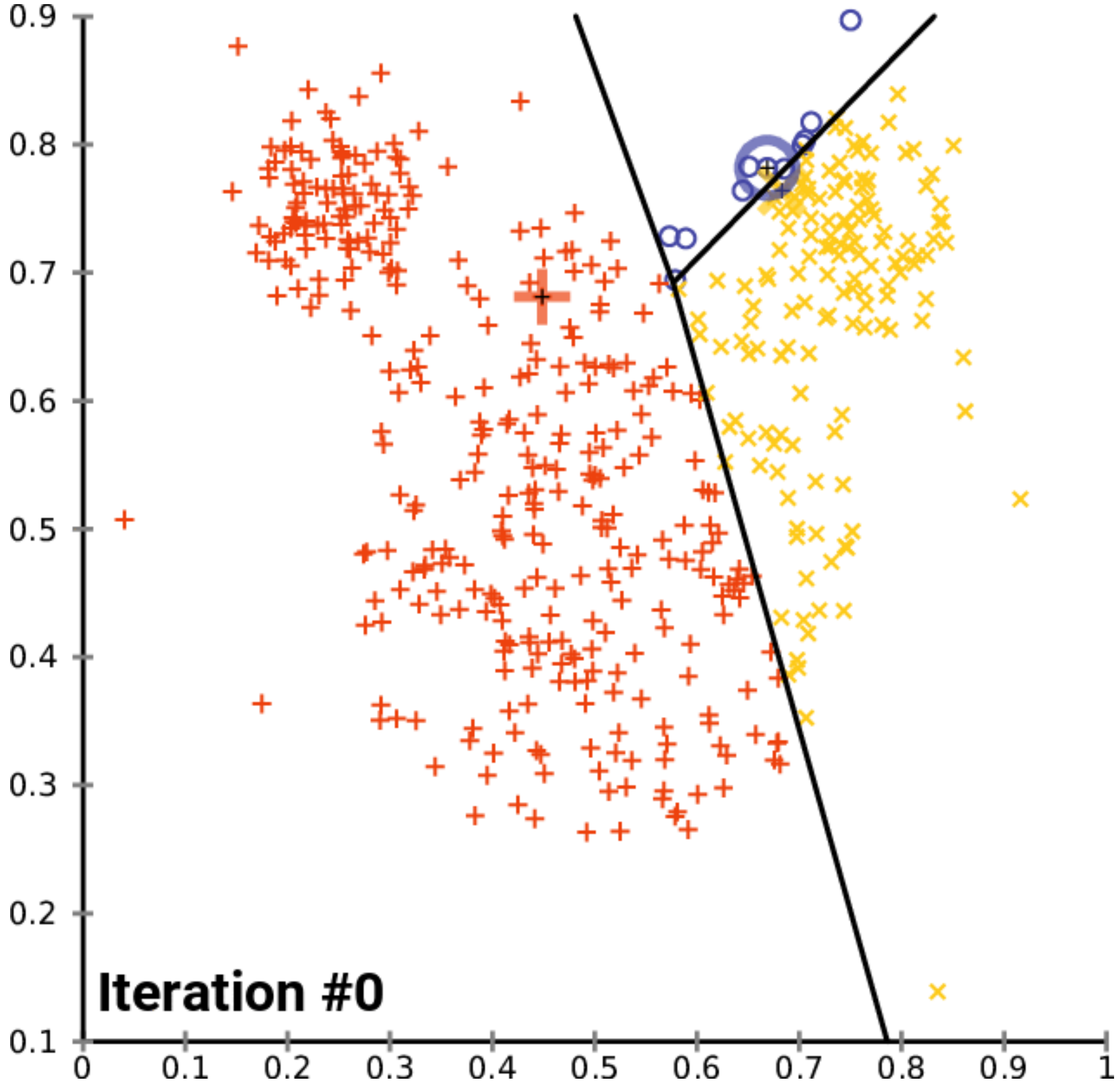
- Unsupervised Learning = **discovering structure in (unlabeled) data**
- Usually have to optimize a **surrogate objective**, since you can't optimize a supervised one
  - Hope/hypothesis: this **correlates** with something you care about
- Last time: discovering **discrete structure** (clusters, segmentation)
  - Hope these stand in for **supervised outputs/labels**





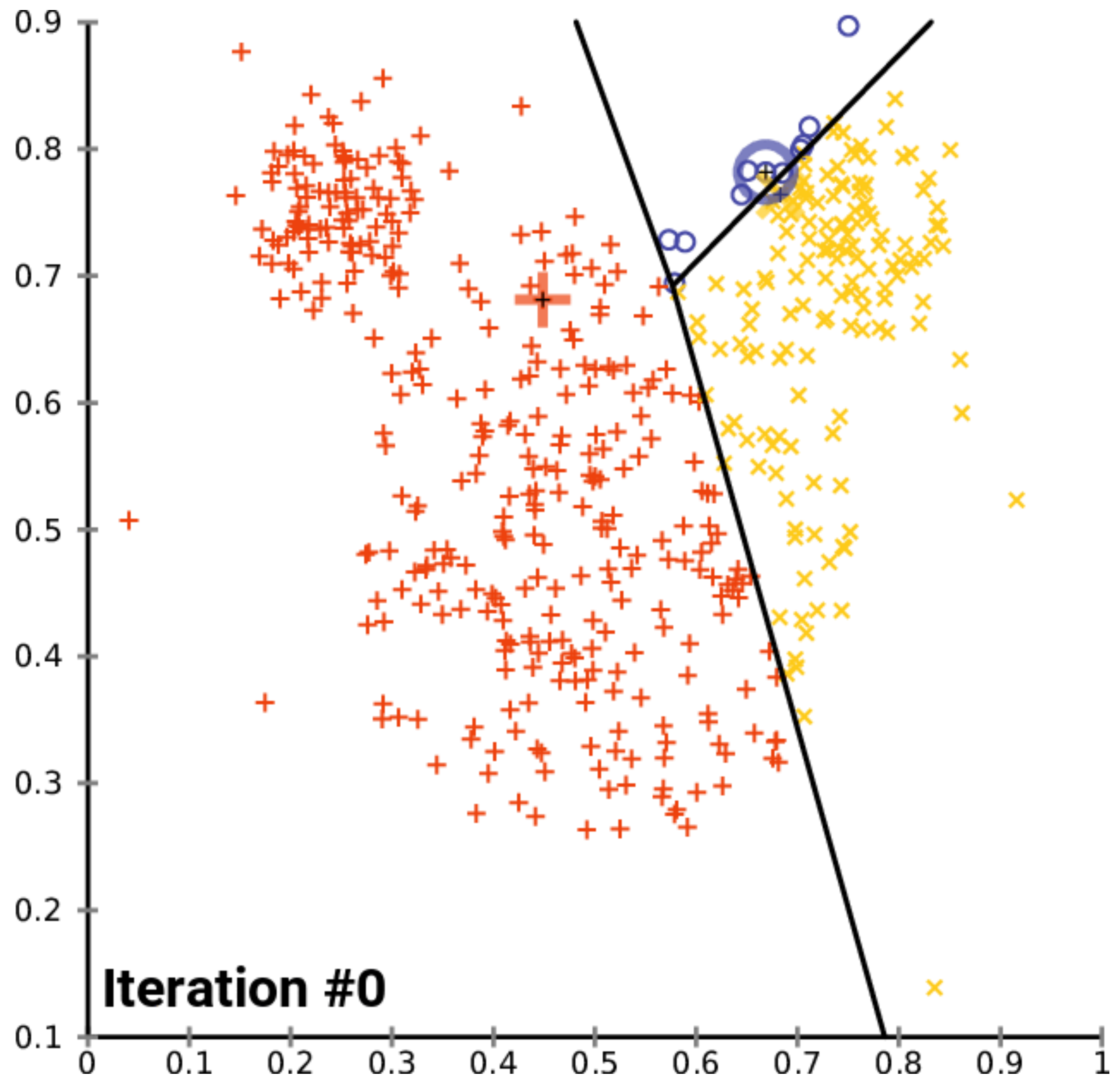


Clustering



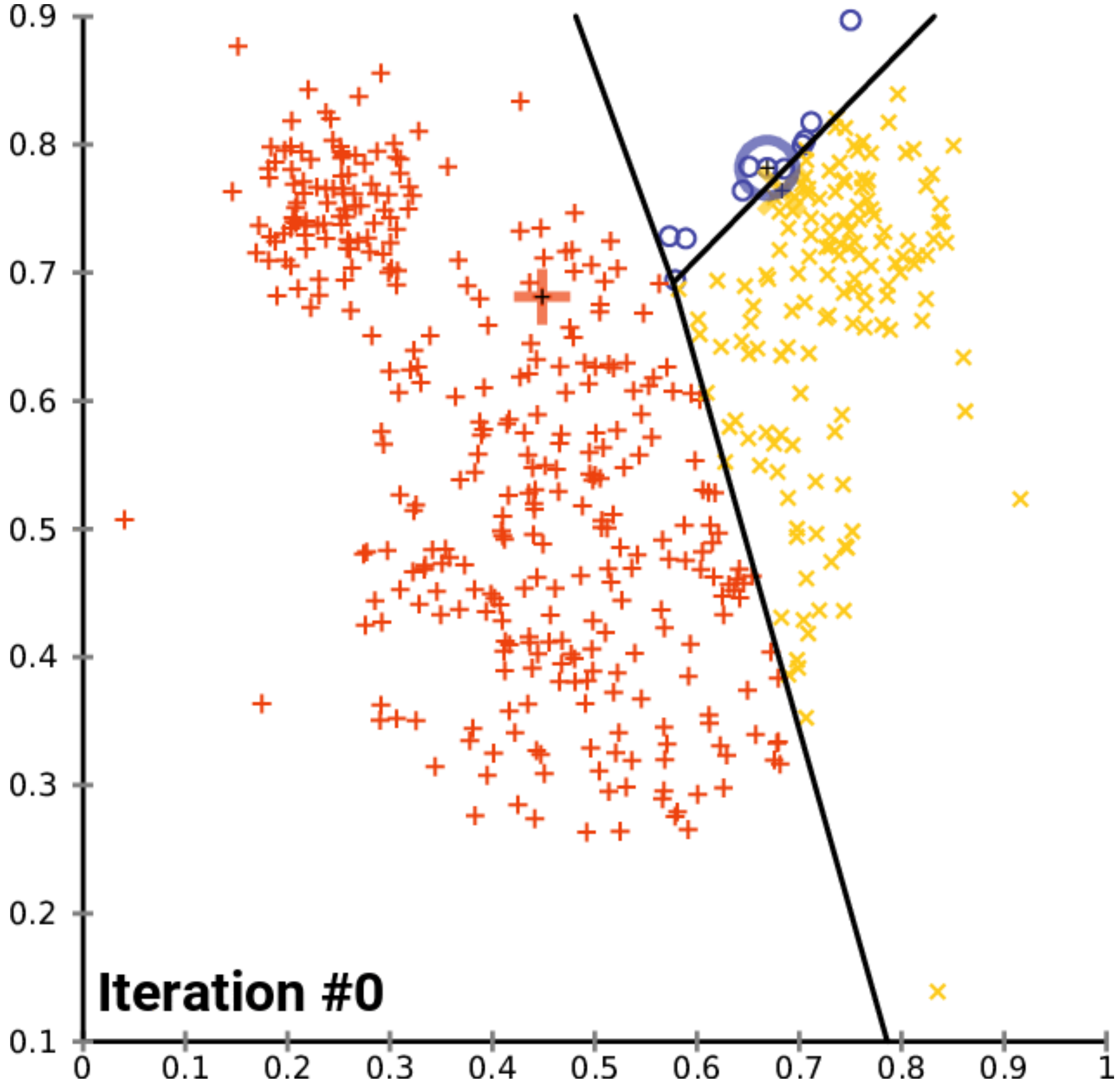
Clustering

- Goal: find **groupings** in data



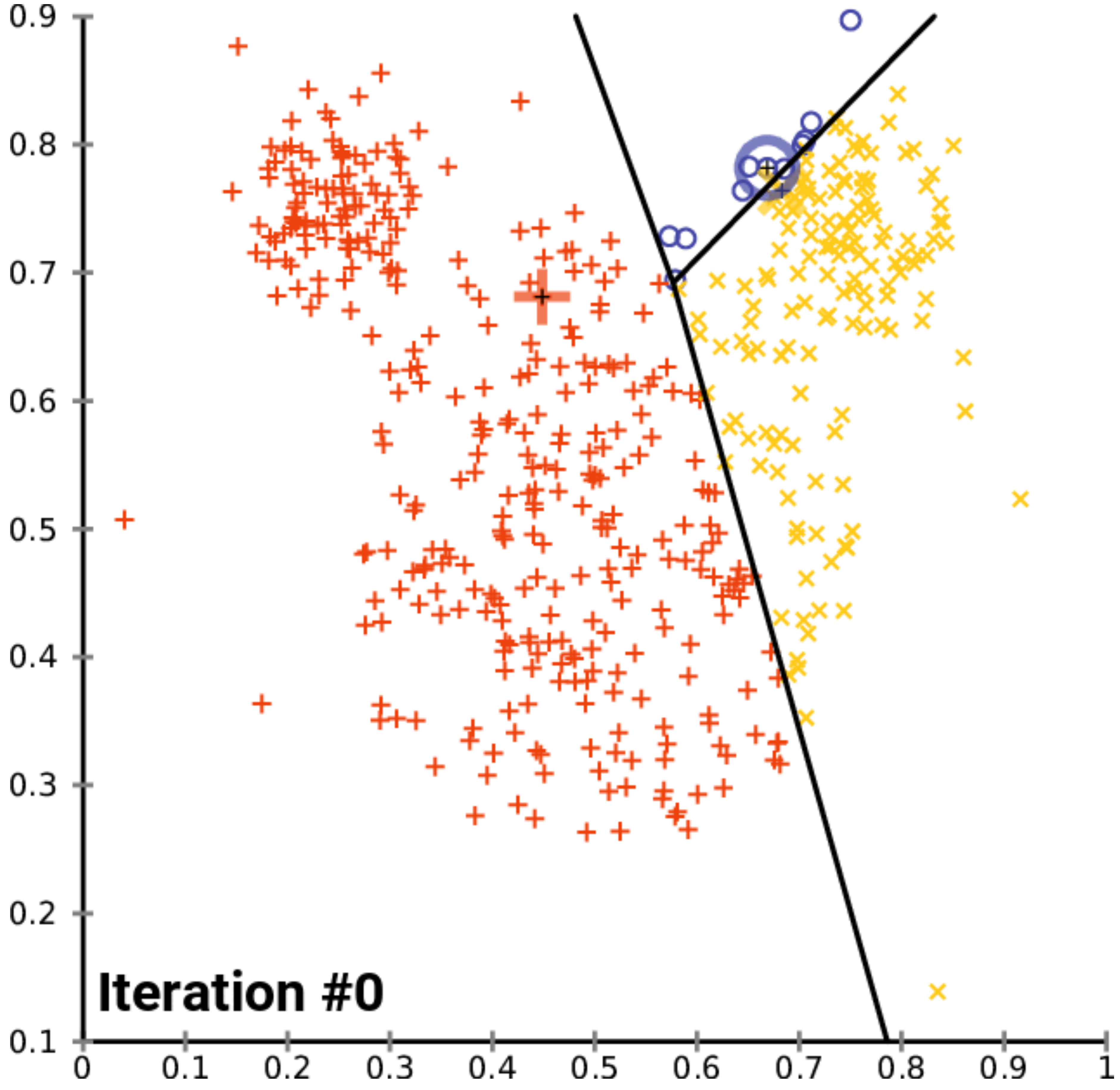
Clustering

- Goal: find **groupings** in data
- Surrogate metric: **cluster quality**
  - K-means: minimize **Within-Cluster Sum of Squares** (WCSS, "compactness")
  - Silhouette Score: measures **coherence** and **distinctness**



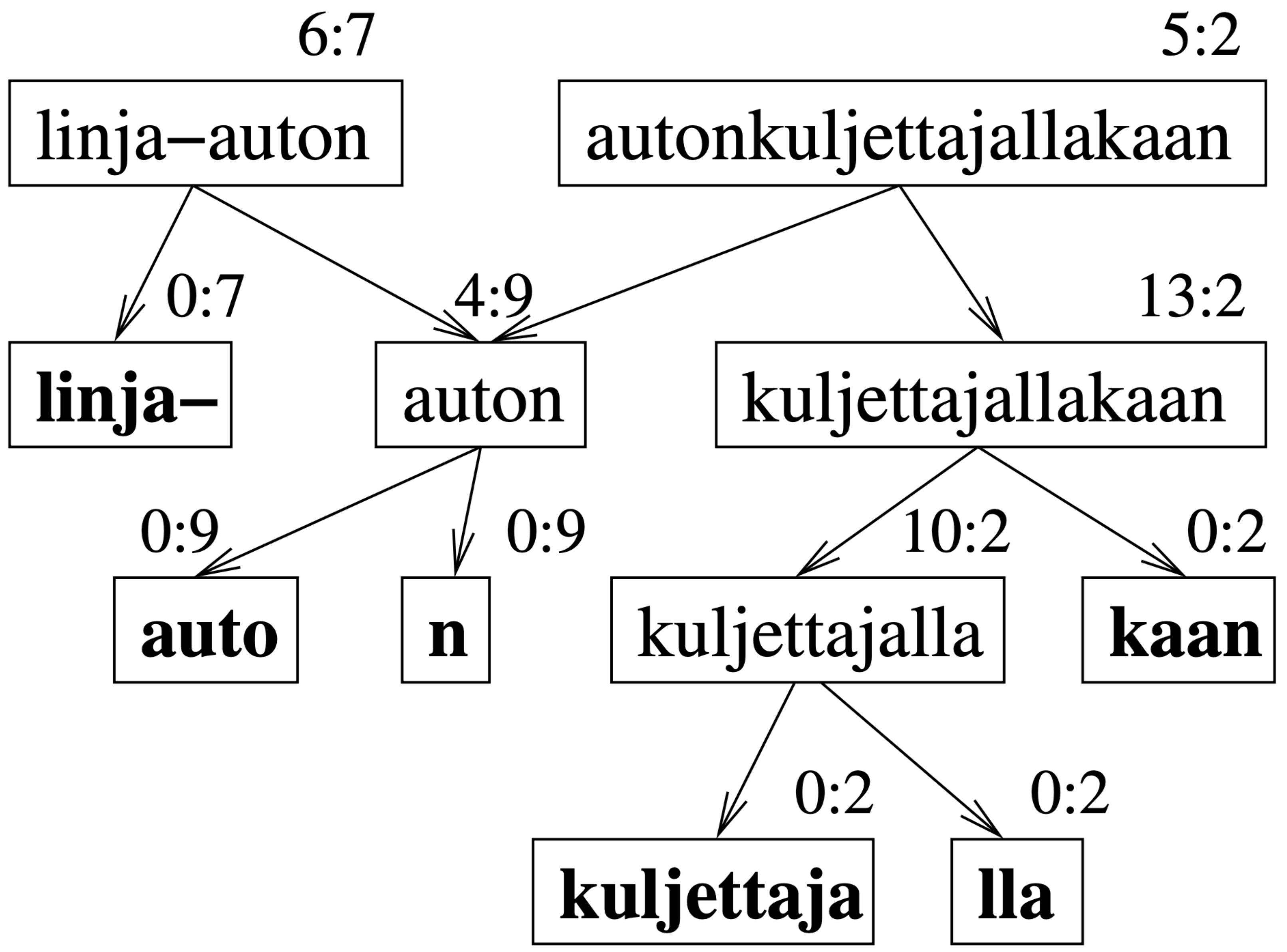
Clustering

- Goal: find **groupings** in data
- Surrogate metric: **cluster quality**
  - K-means: minimize **Within-Cluster Sum of Squares** (WCSS, "compactness")
  - Silhouette Score: measures **coherence** and **distinctness**
- Inductive biases: all kinds!
  - K-means biased towards **spherical clusters**
  - Other algorithms have different biases



Sequence Segmentation

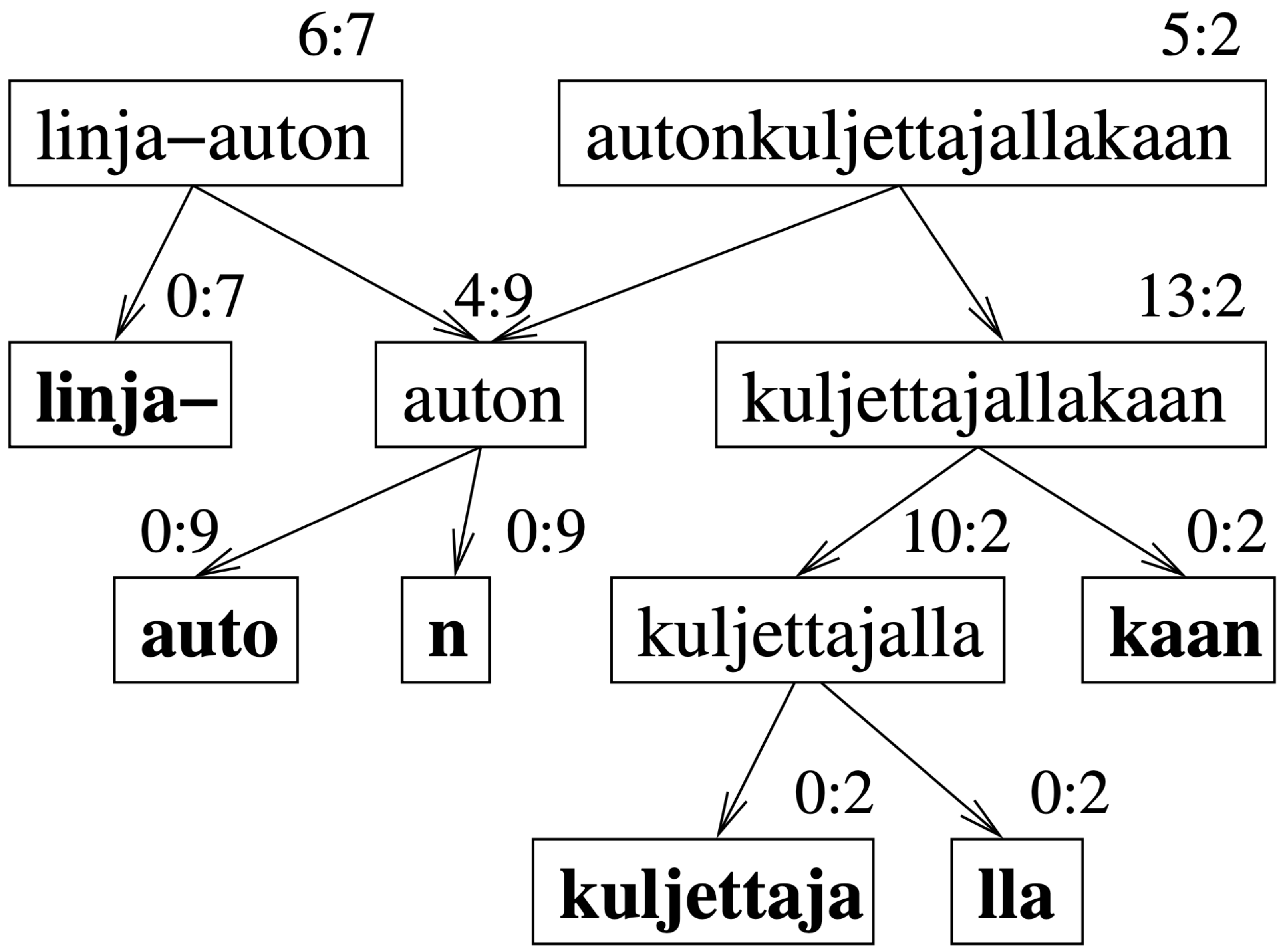
$$
\begin{aligned}
C &= \mathrm{Cost}(\text{Source text}) + \mathrm{Cost}(\text{Codebook}) \\
&= \sum_{\text{tokens}} -\log p(m_i) + \sum_{\text{types}} k * l(m_j)
\end{aligned}
$$



Sequence Segmentation

- Goal: find **meaningful sub-sequences** in language

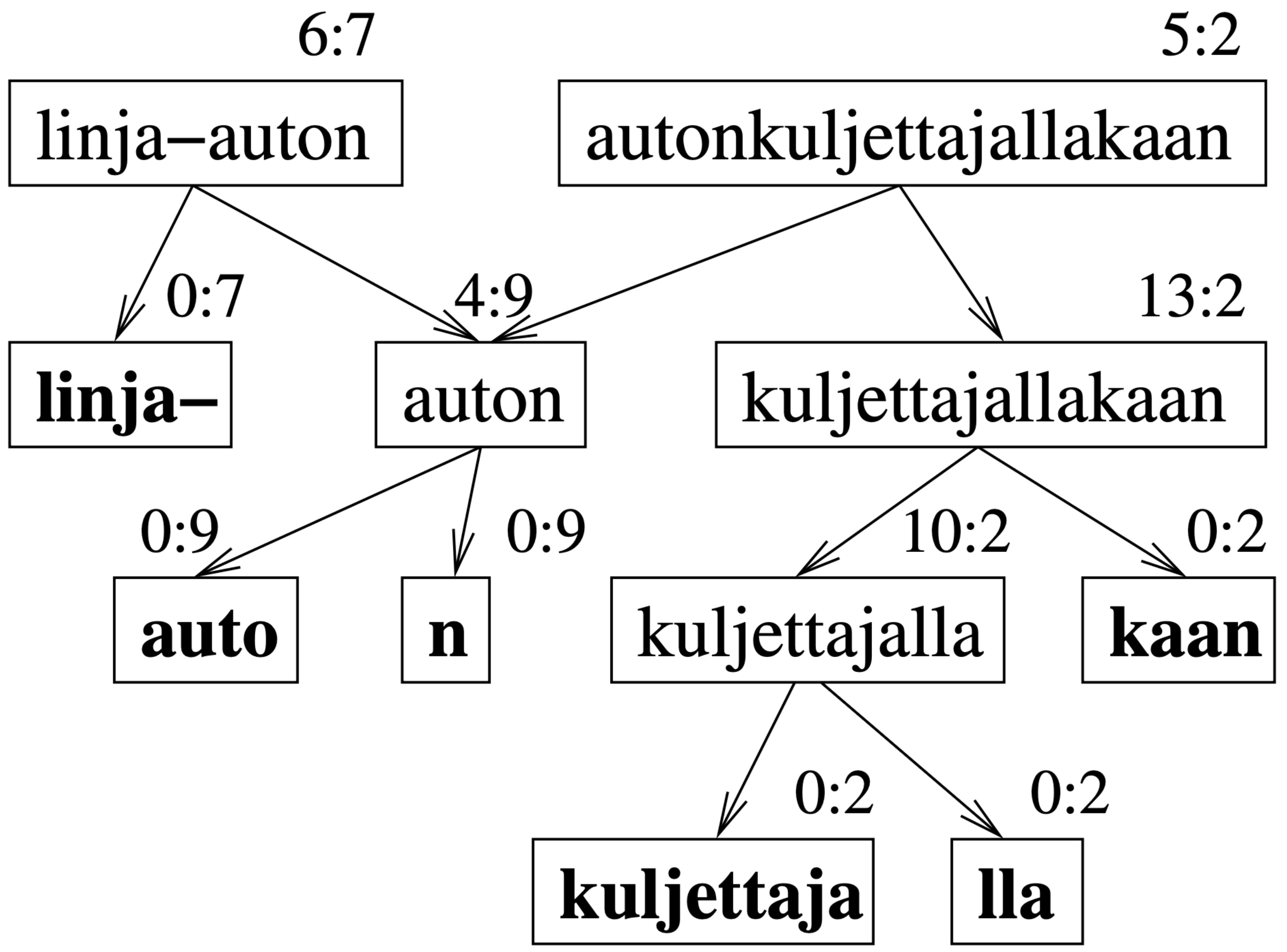
$$
\begin{aligned}
C &= \text{Cost}(\text{Source text}) + \text{Cost}(\text{Codebook}) \\
&= \sum_{\text{tokens}} -\log p(m_i) + \sum_{\text{types}} k * l(m_j)
\end{aligned}
$$



Sequence Segmentation

- Goal: find **meaningful sub-sequences** in language
- Surrogate metric: **Minimum Description Length (MDL)**
  - Simultaneously minimize the **data complexity** and **description length**
  - More unique, long units → **longer description** → under-segmentation

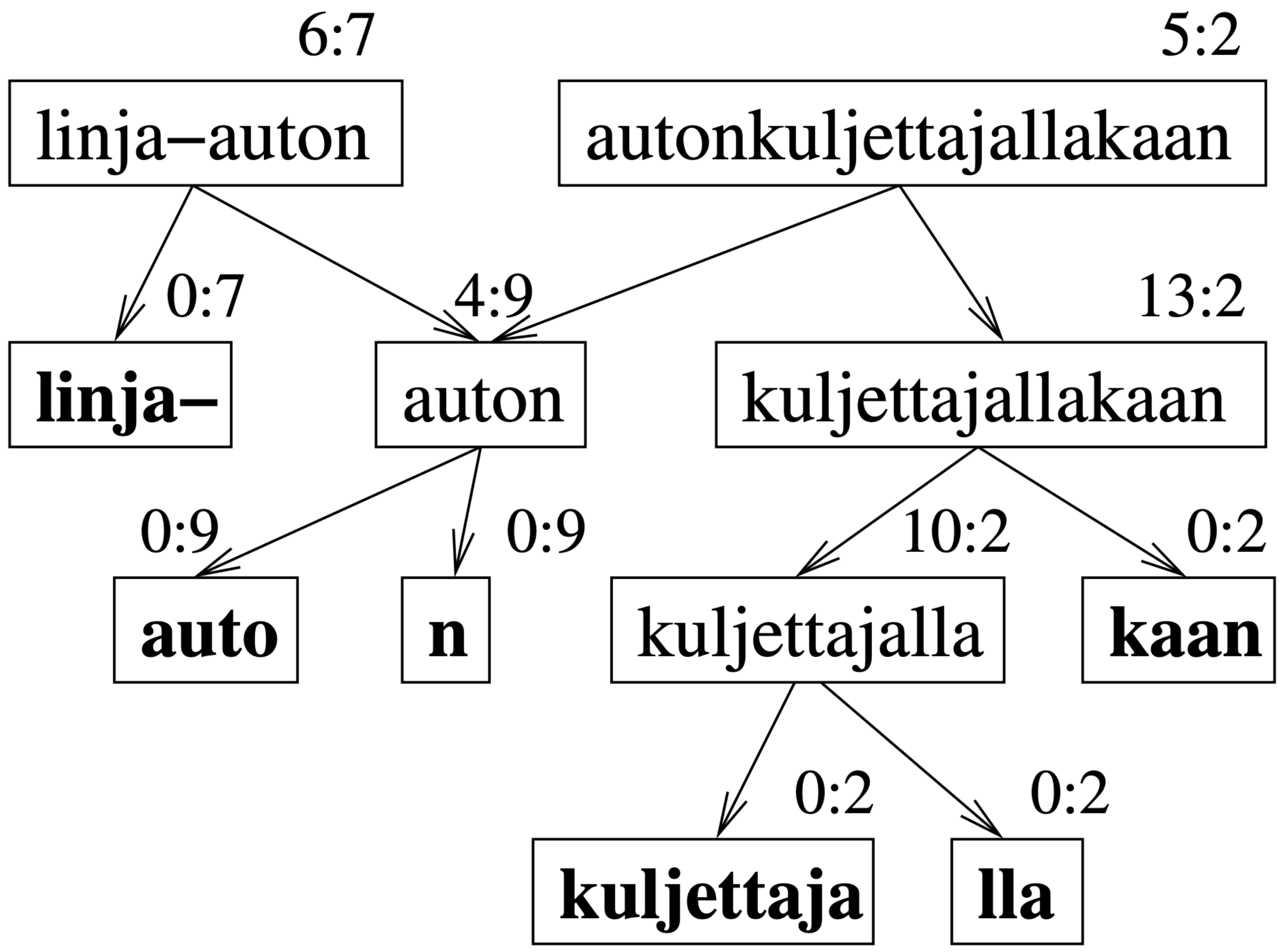
$$
\begin{aligned}
C &= \mathrm{Cost}(\text{Source text}) + \mathrm{Cost}(\text{Codebook}) \\
&= \sum_{\text{tokens}} -\log p(m_i) + \sum_{\text{types}} k * l(m_j)
\end{aligned}
$$



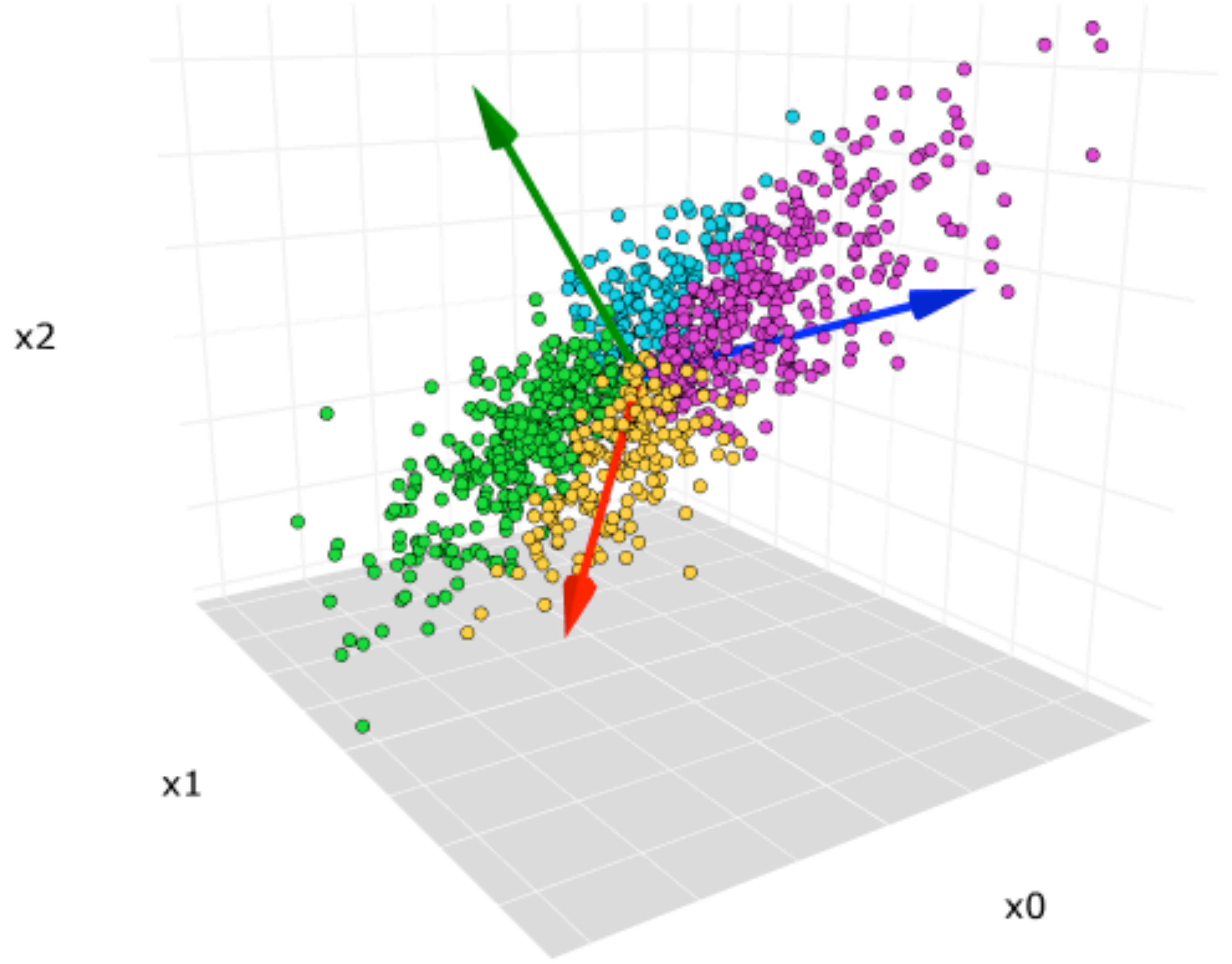
Sequence Segmentation

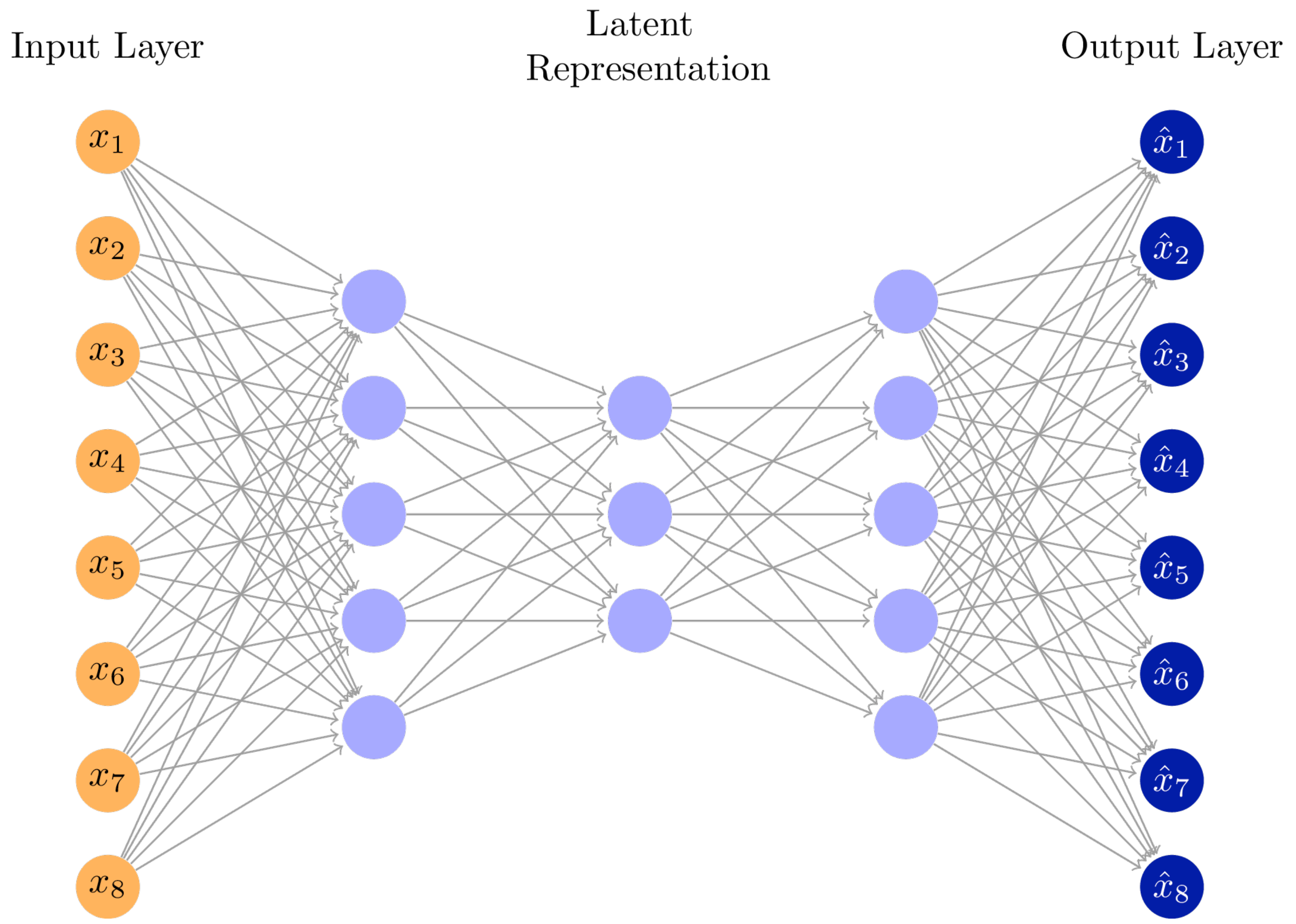
- Goal: find **meaningful sub-sequences** in language
- Surrogate metric: **Minimum Description Length (MDL)**
  - Simultaneously minimize the **data complexity** and **description length**
  - More unique, long units → **longer description** → under-segmentation
- Inductive bias: assumes language **prefers efficiency** (not guaranteed)

$$
\begin{aligned}
C &= \mathrm{Cost}(\text{Source text}) + \mathrm{Cost}(\text{Codebook}) \\
&= \sum_{\text{tokens}} -\log p(m_i) + \sum_{\text{types}} k * l(m_j)
\end{aligned}
$$



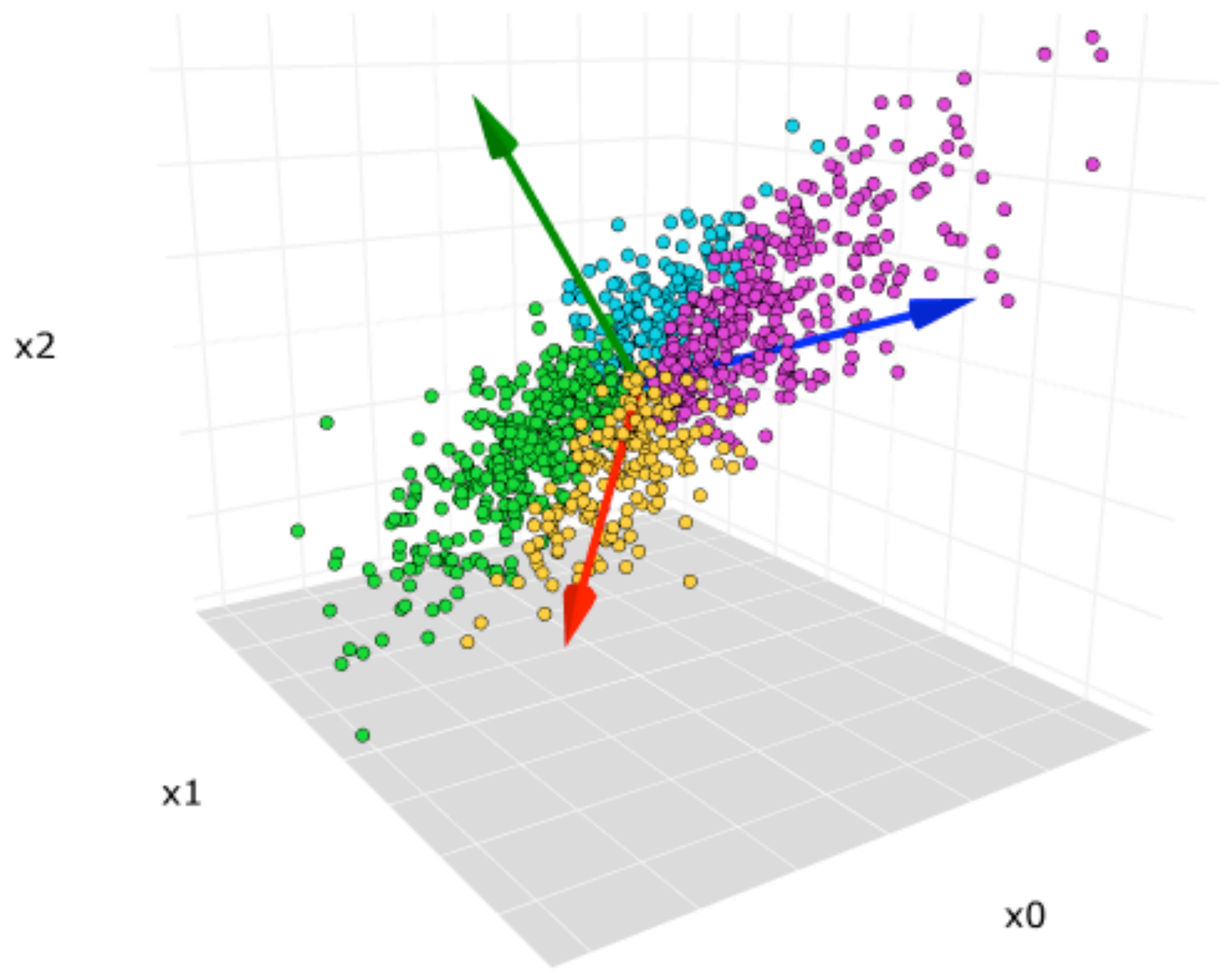
Today: Continuous Structure

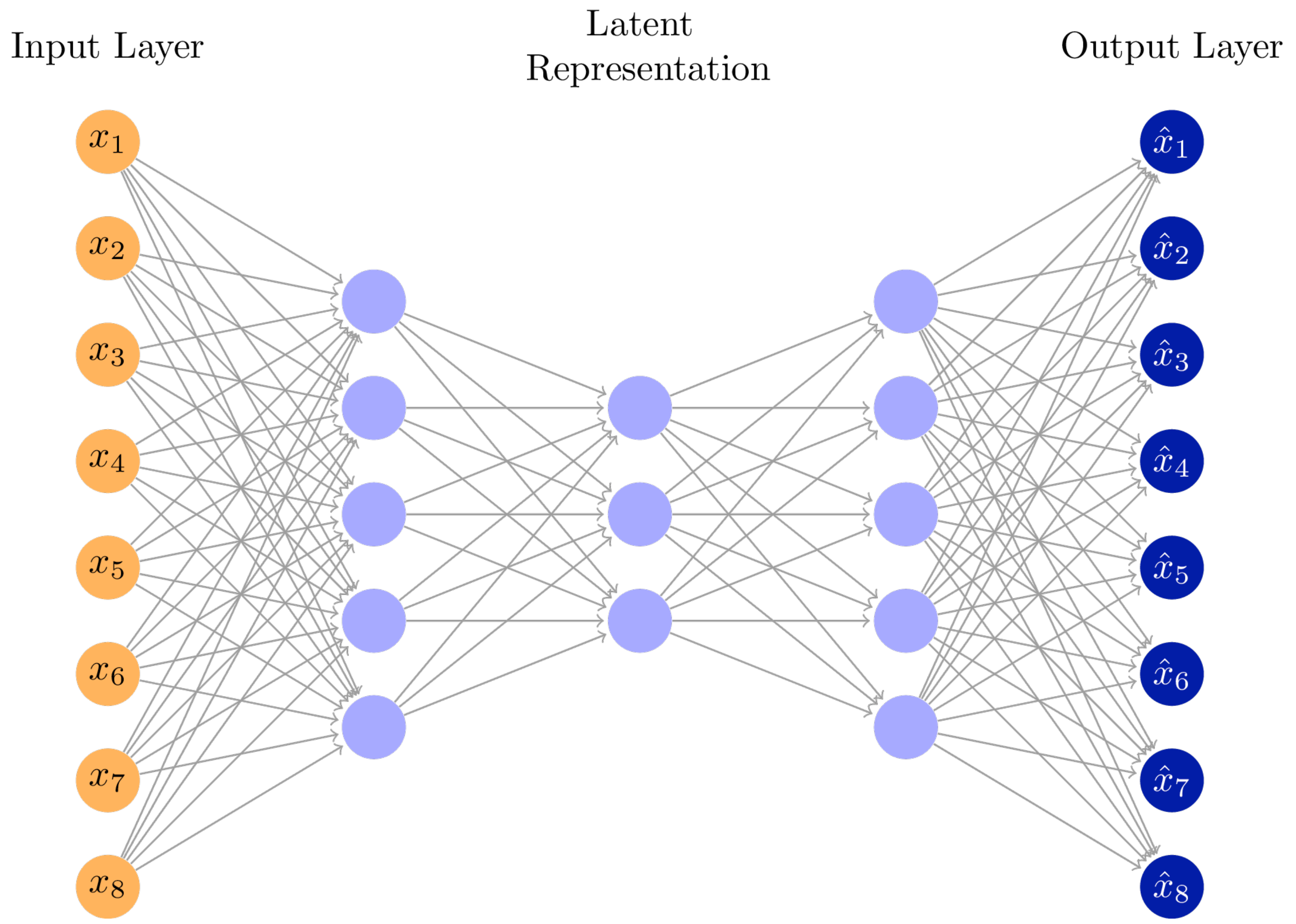




Today: Continuous Structure

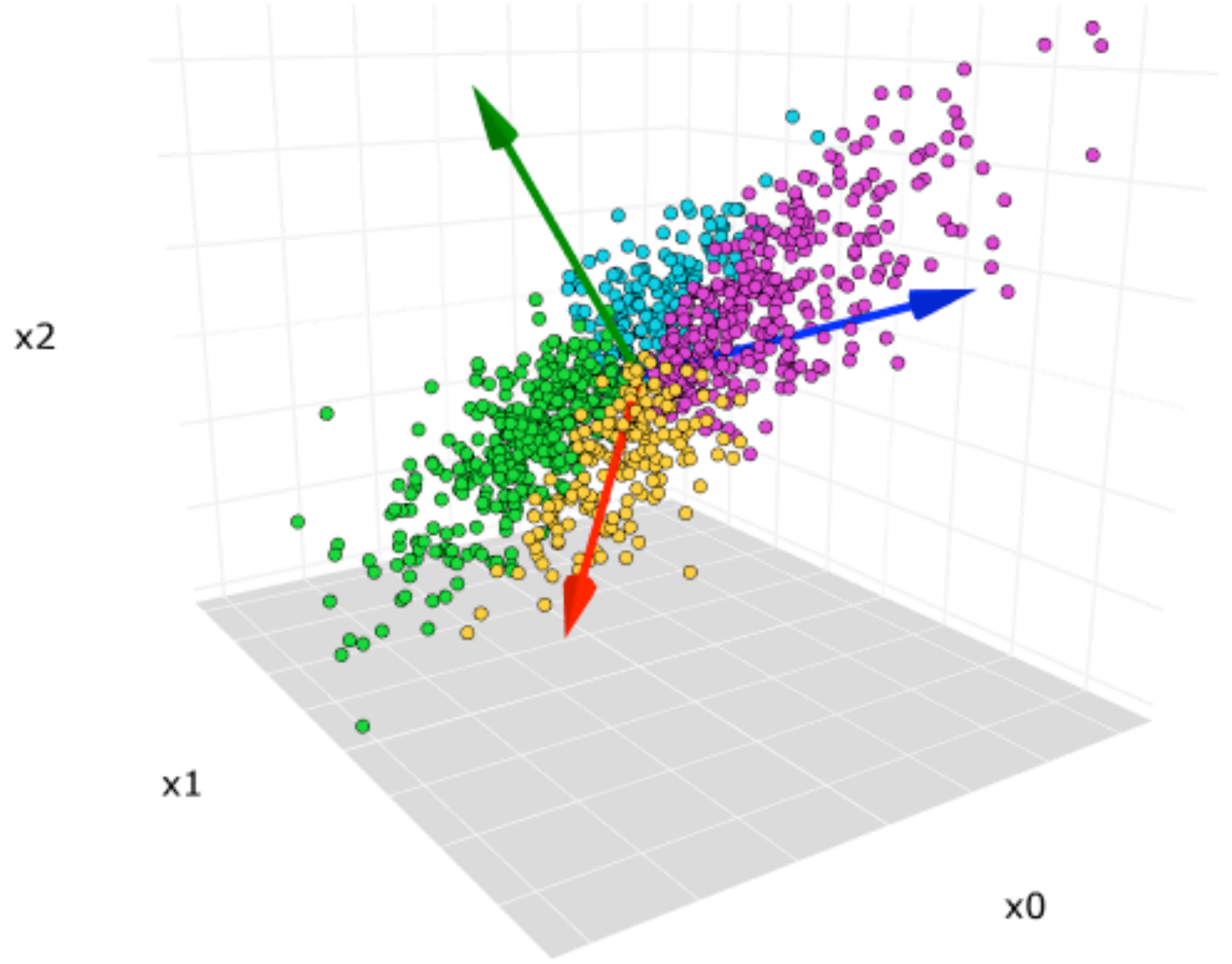
- Instead of categorizing data, find **useful transformations**

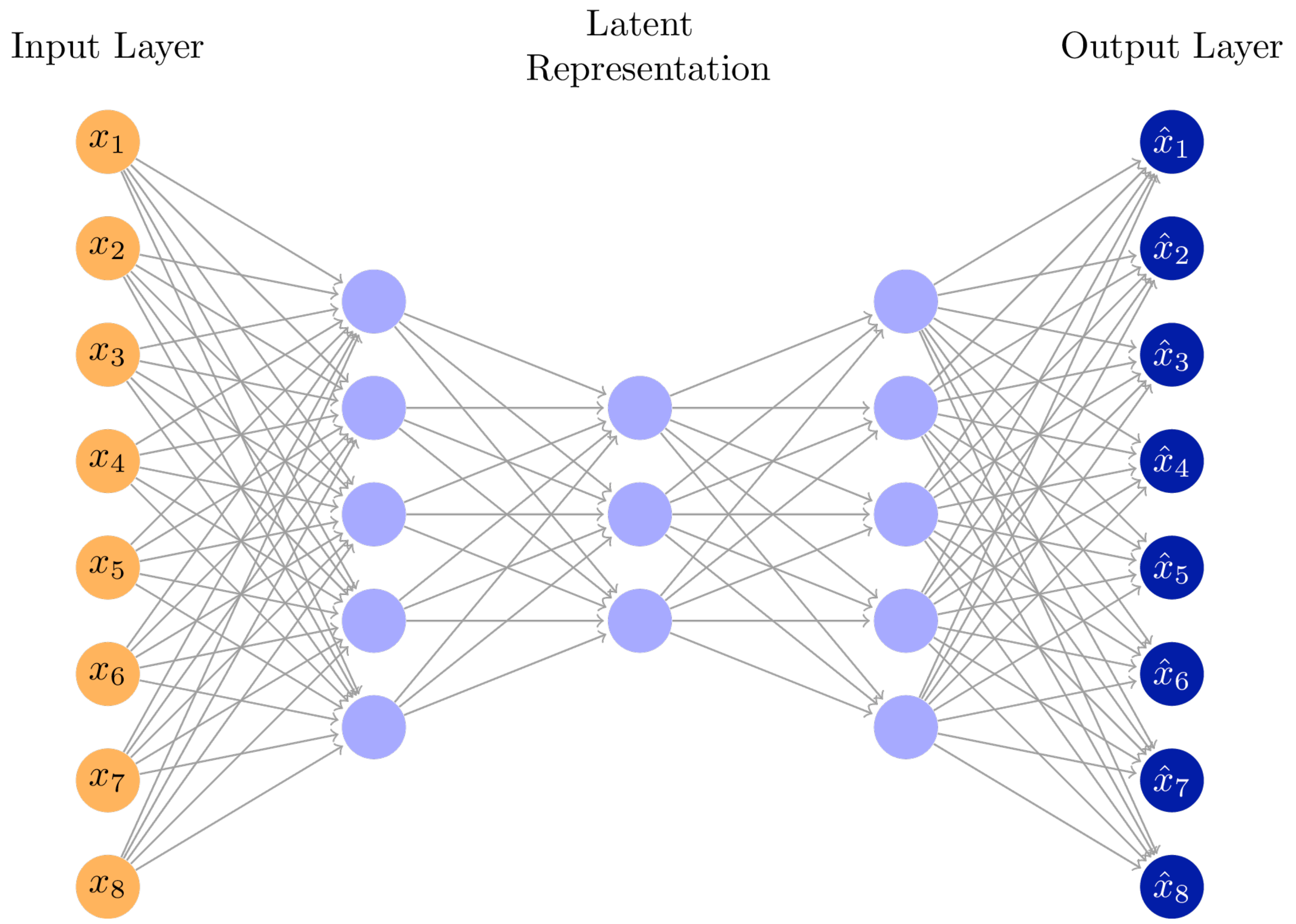




Today: Continuous Structure

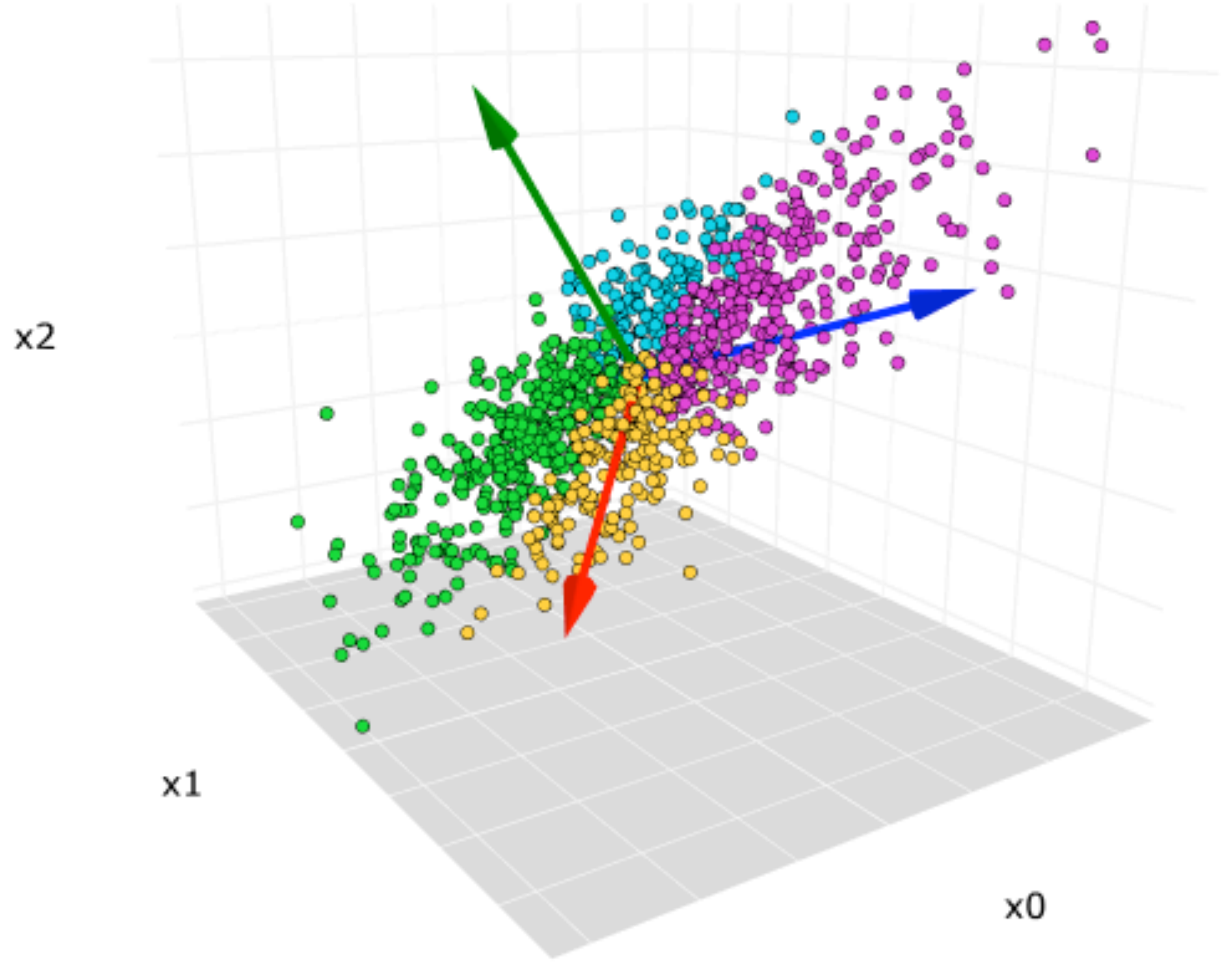
- Instead of categorizing data, find **useful transformations**
- Surrogate objectives: **dimensionality reduction**
  - Either via **explaining variance** or minimizing **reconstruction error**

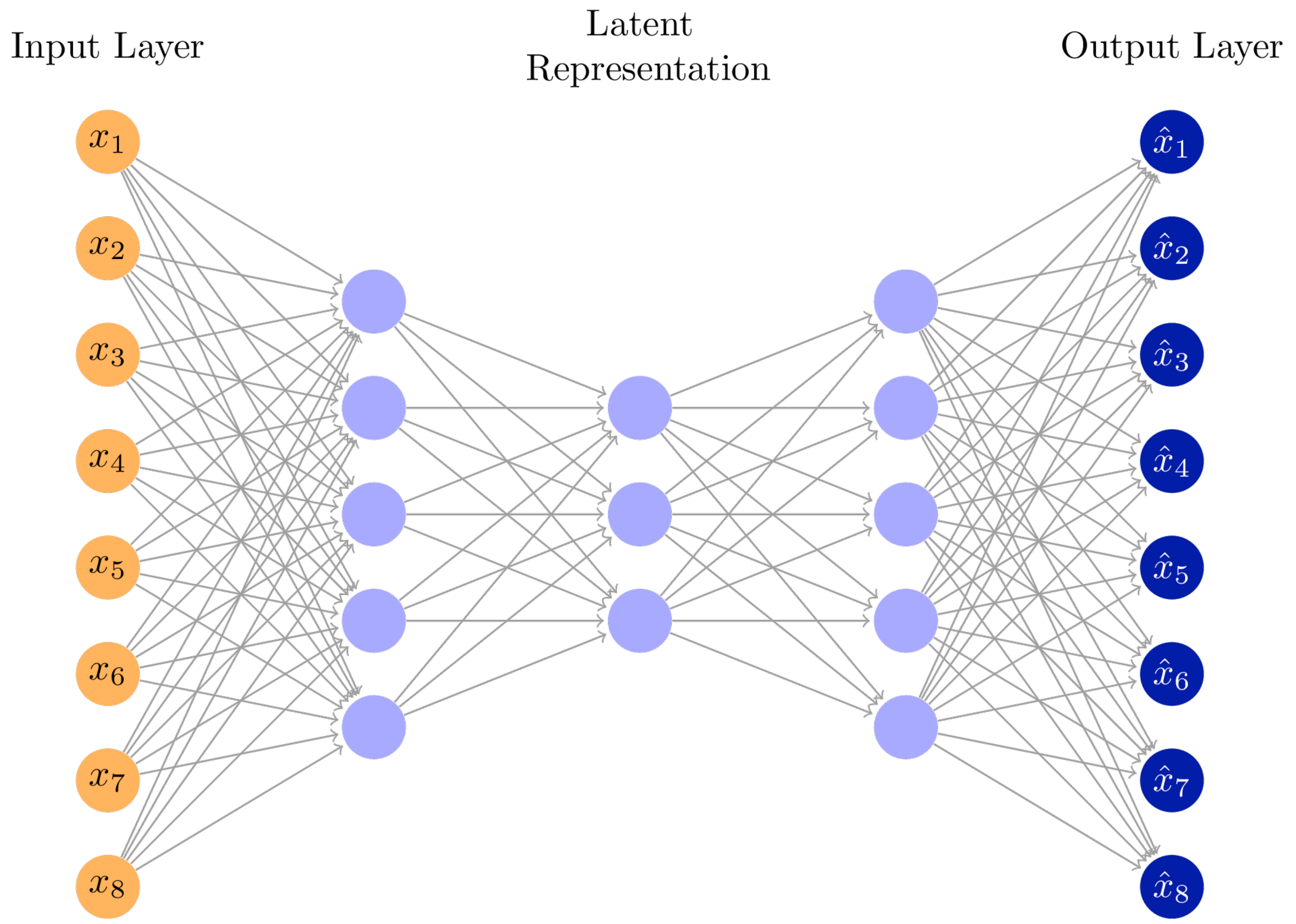




Today: Continuous Structure

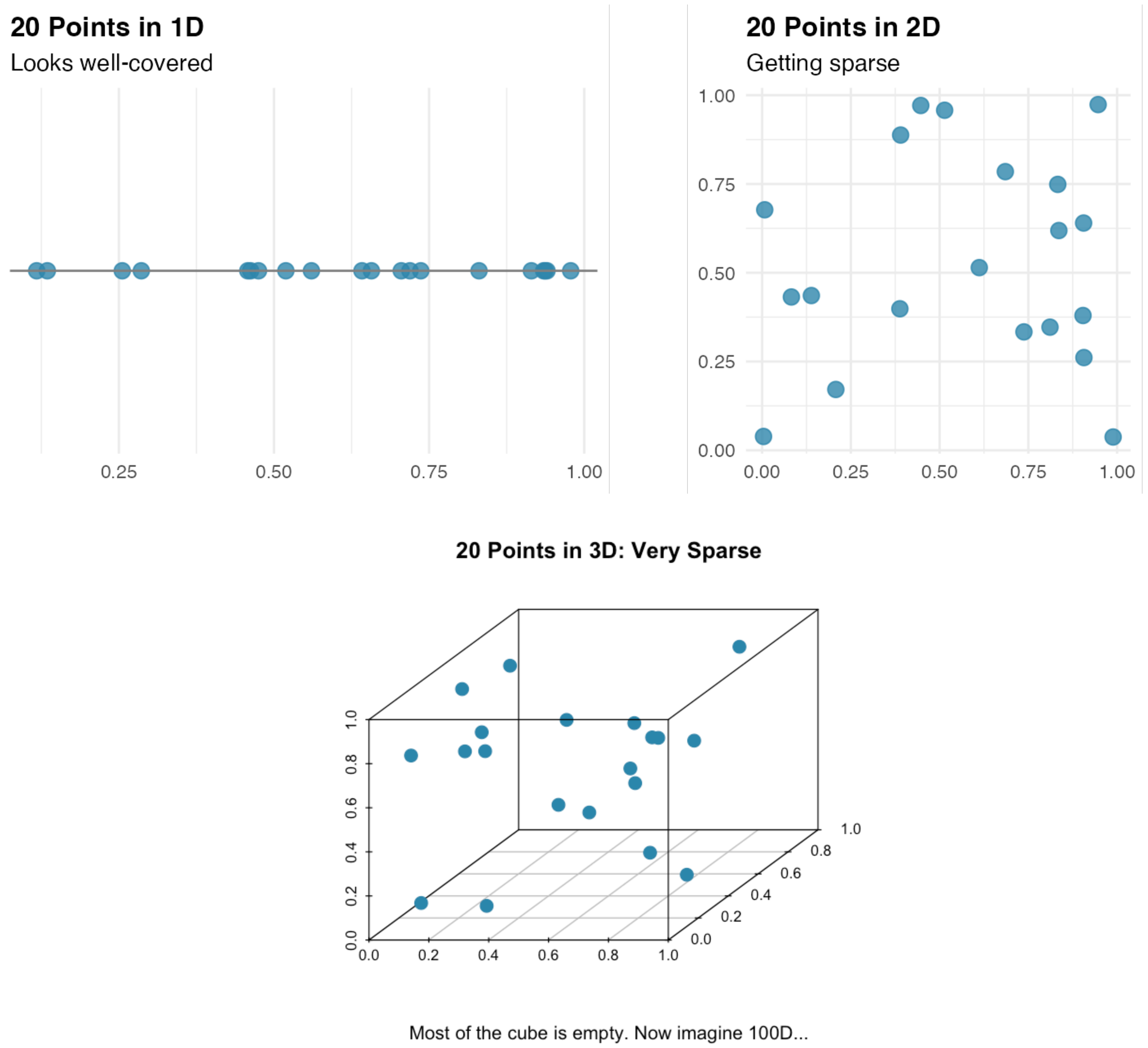
- Instead of categorizing data, find **useful transformations**
- Surrogate objectives: **dimensionality reduction**
  - Either via **explaining variance** or minimizing **reconstruction error**
- The hope: remaining dims are **more useful for downstream tasks**
  - Inductive bias that **most dimensions are redundant/noise**





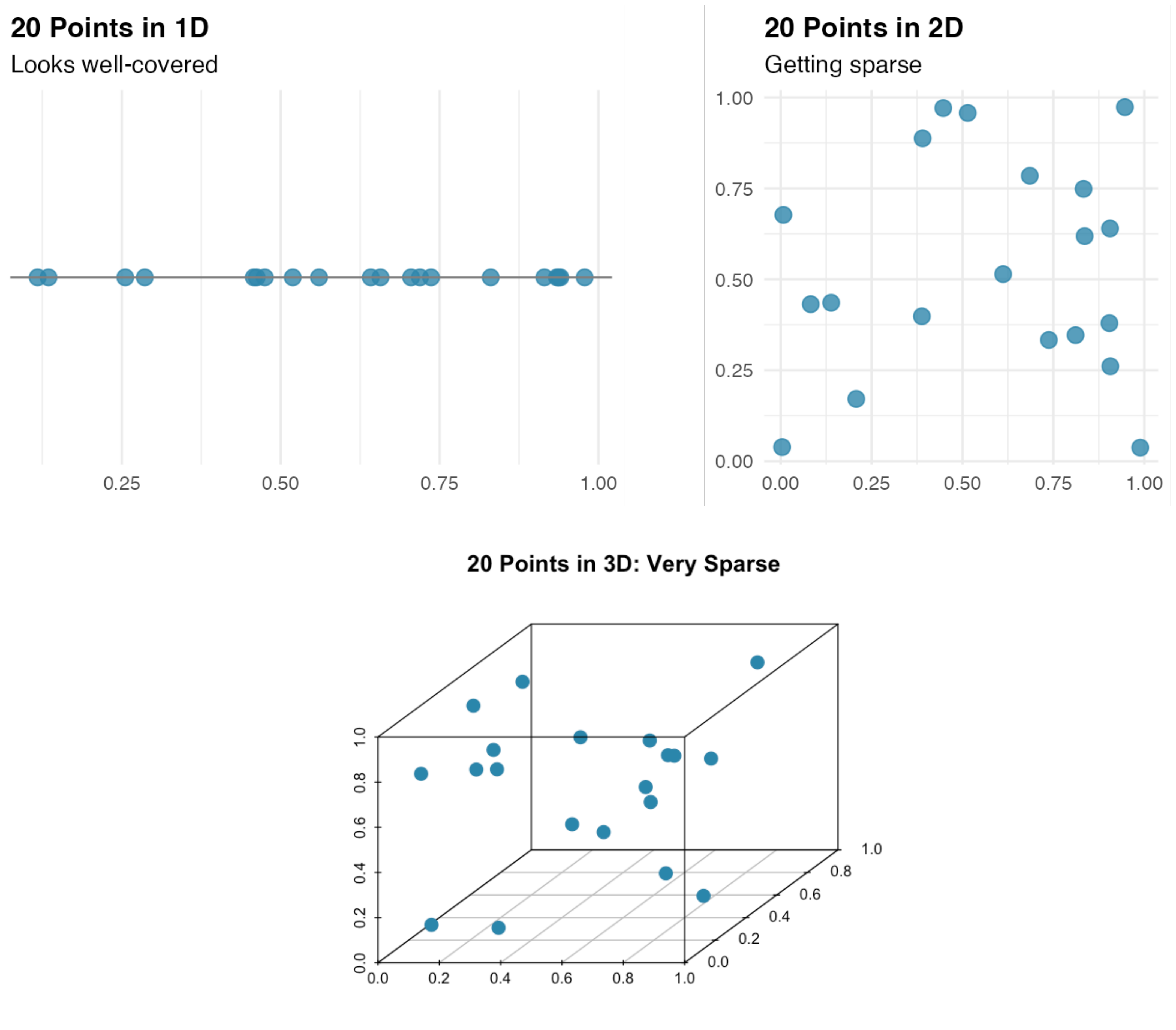
Dimensionality Reduction

The "Curse of Dimensionality"



The "Curse of Dimensionality"

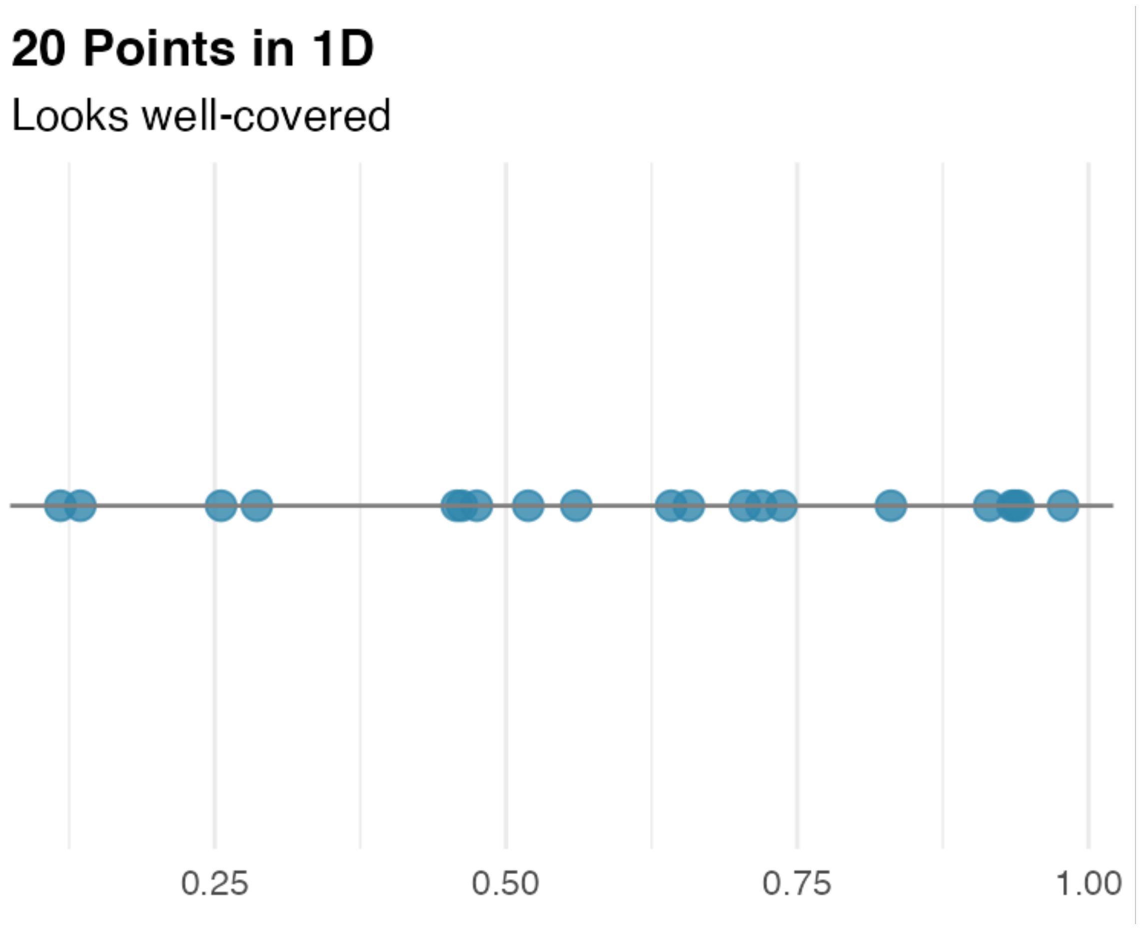
- High-dimensional data is very **sparse** (i.e. empty)

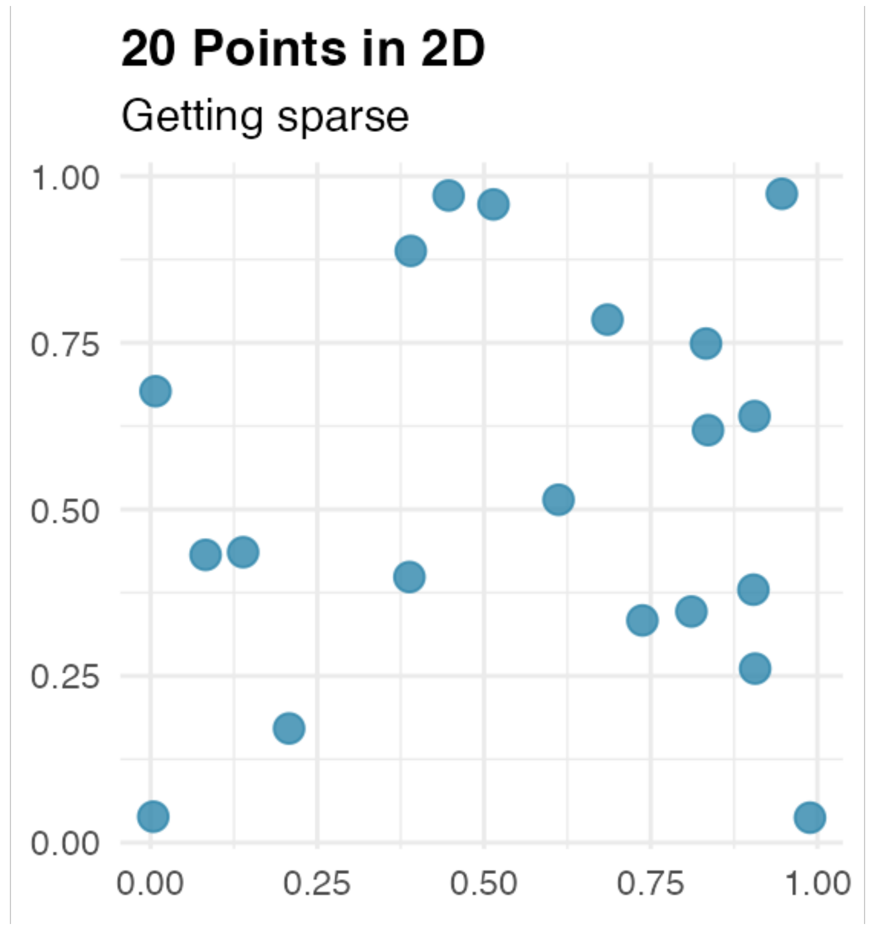


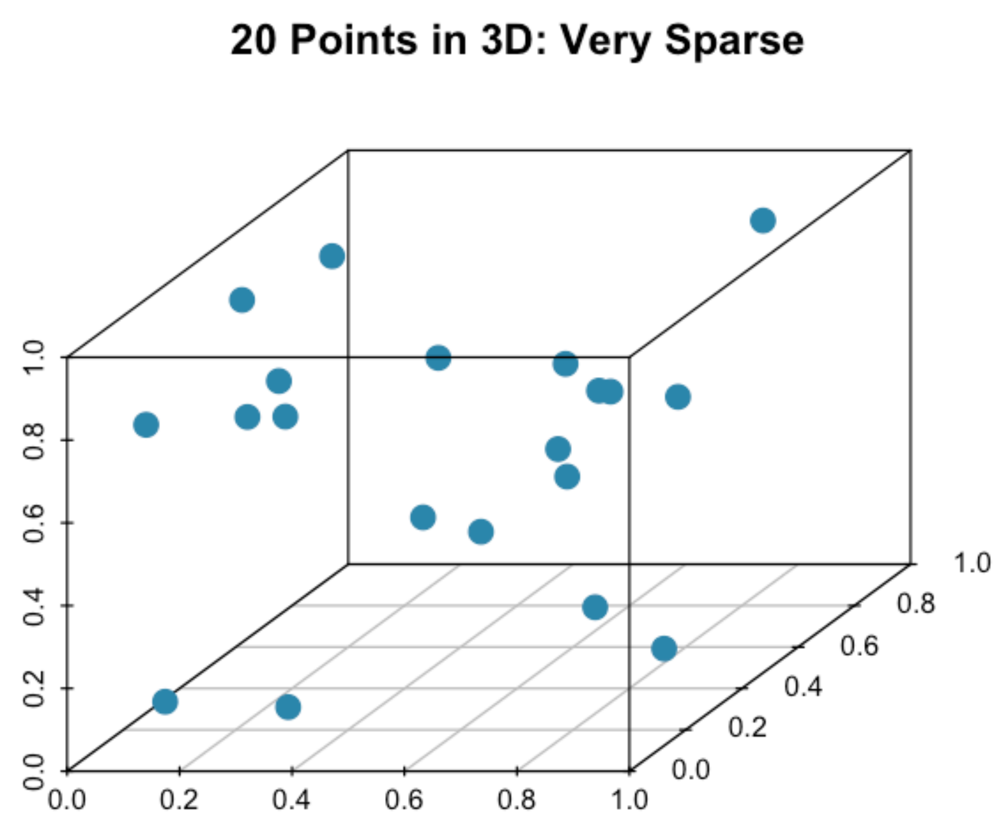
Most of the cube is empty. Now imagine 100D...

The "Curse of Dimensionality"

- High-dimensional data is very **sparse** (i.e. empty)
  - Say you have **50 datapoints**, spread across...



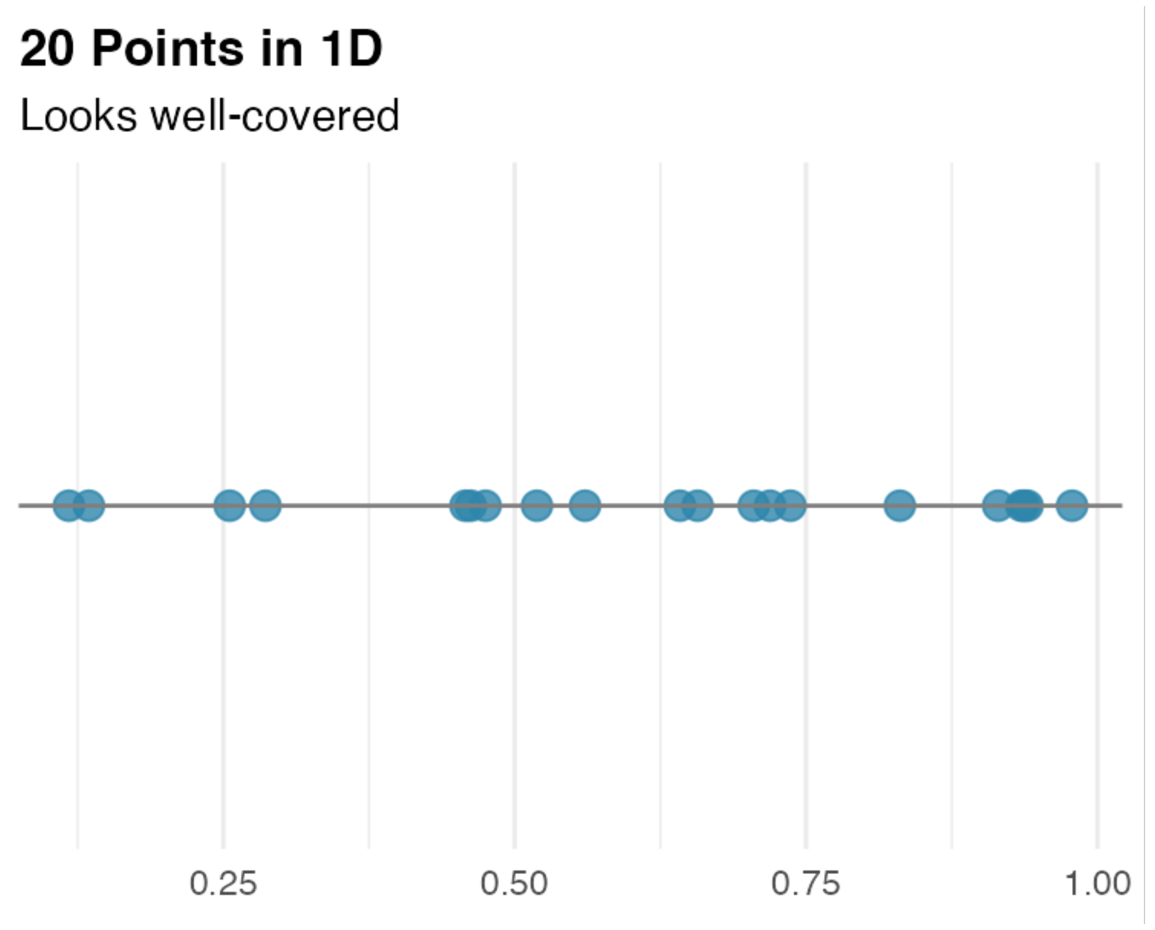


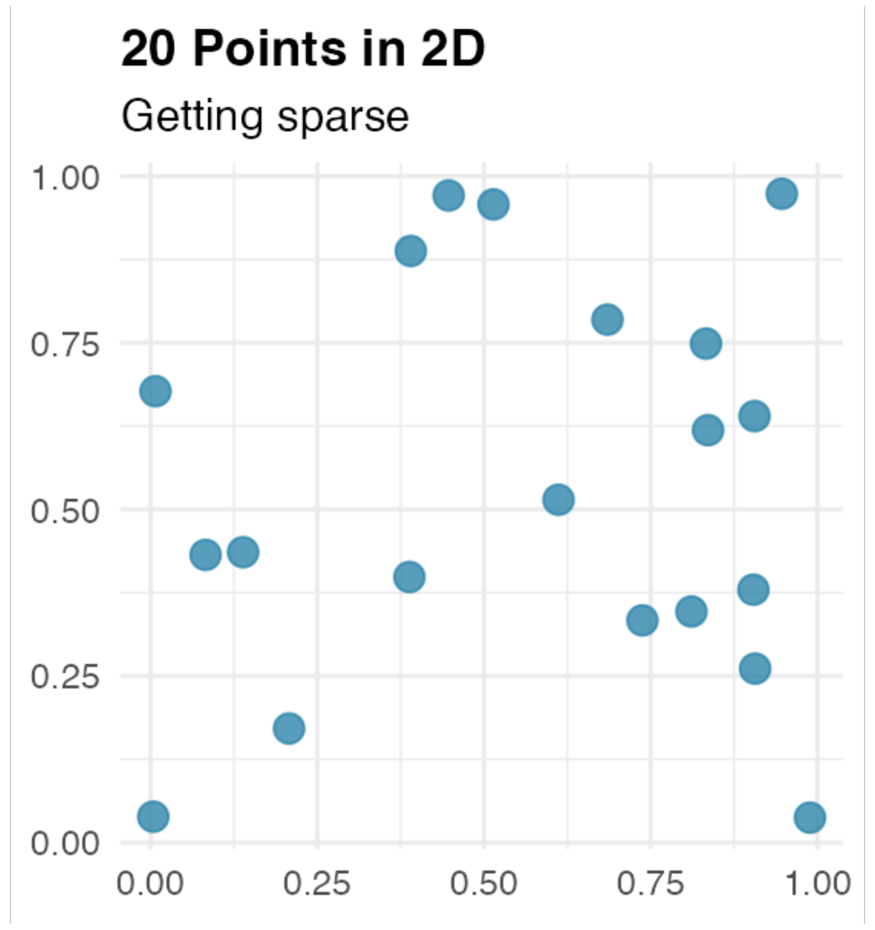


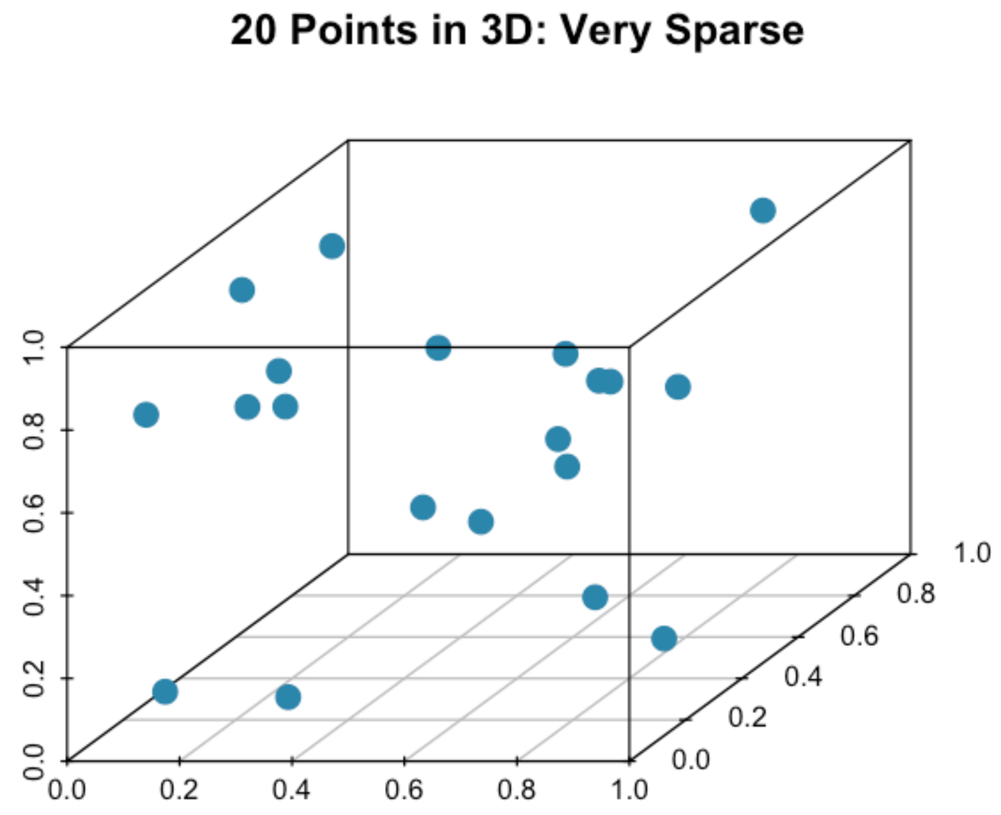
Most of the cube is empty. Now imagine 100D...

The "Curse of Dimensionality"

- High-dimensional data is very **sparse** (i.e. empty)
  - Say you have **50 datapoints**, spread across...
  - A **10x10 grid** → half full



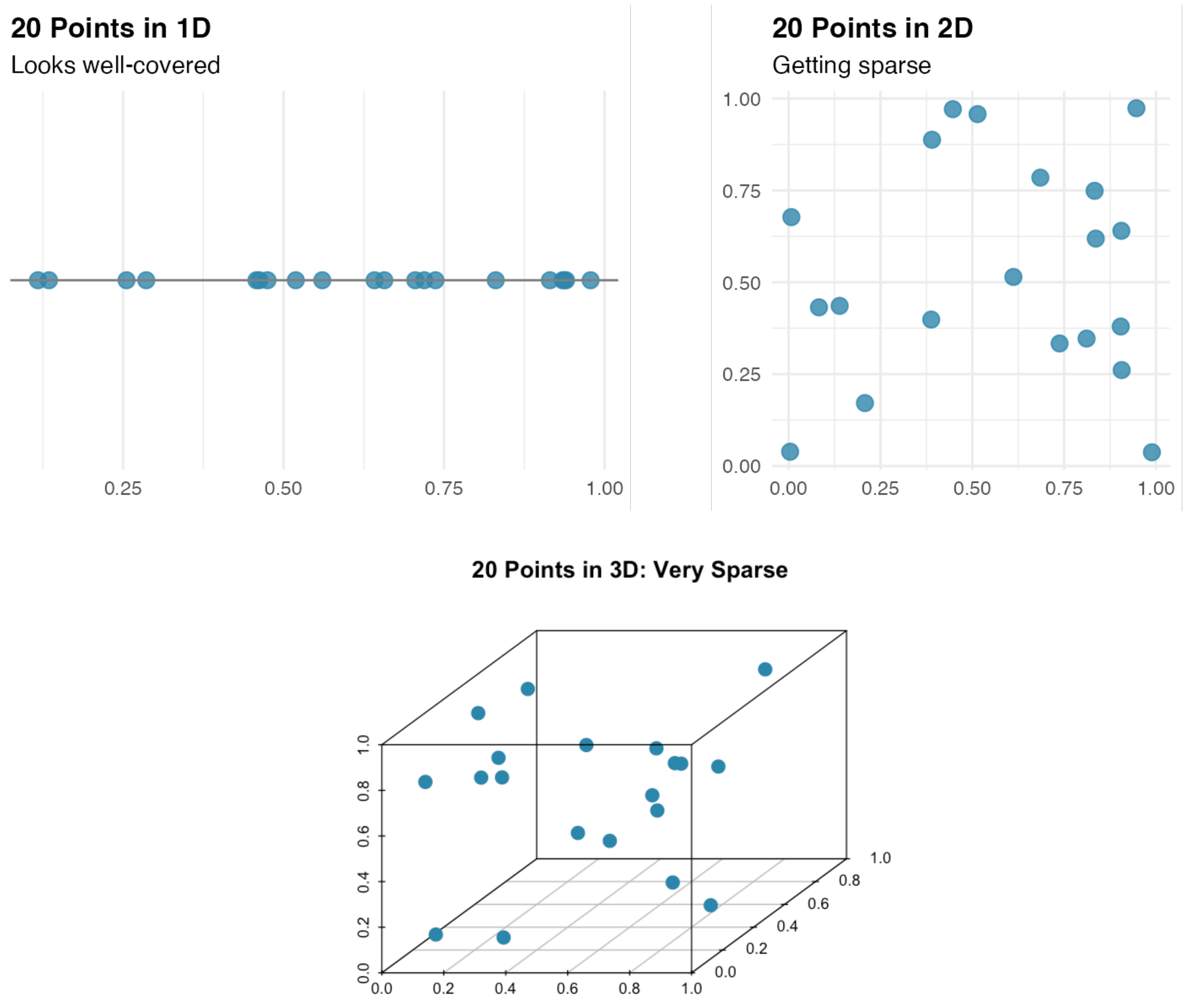




Most of the cube is empty. Now imagine 100D...

The "Curse of Dimensionality"

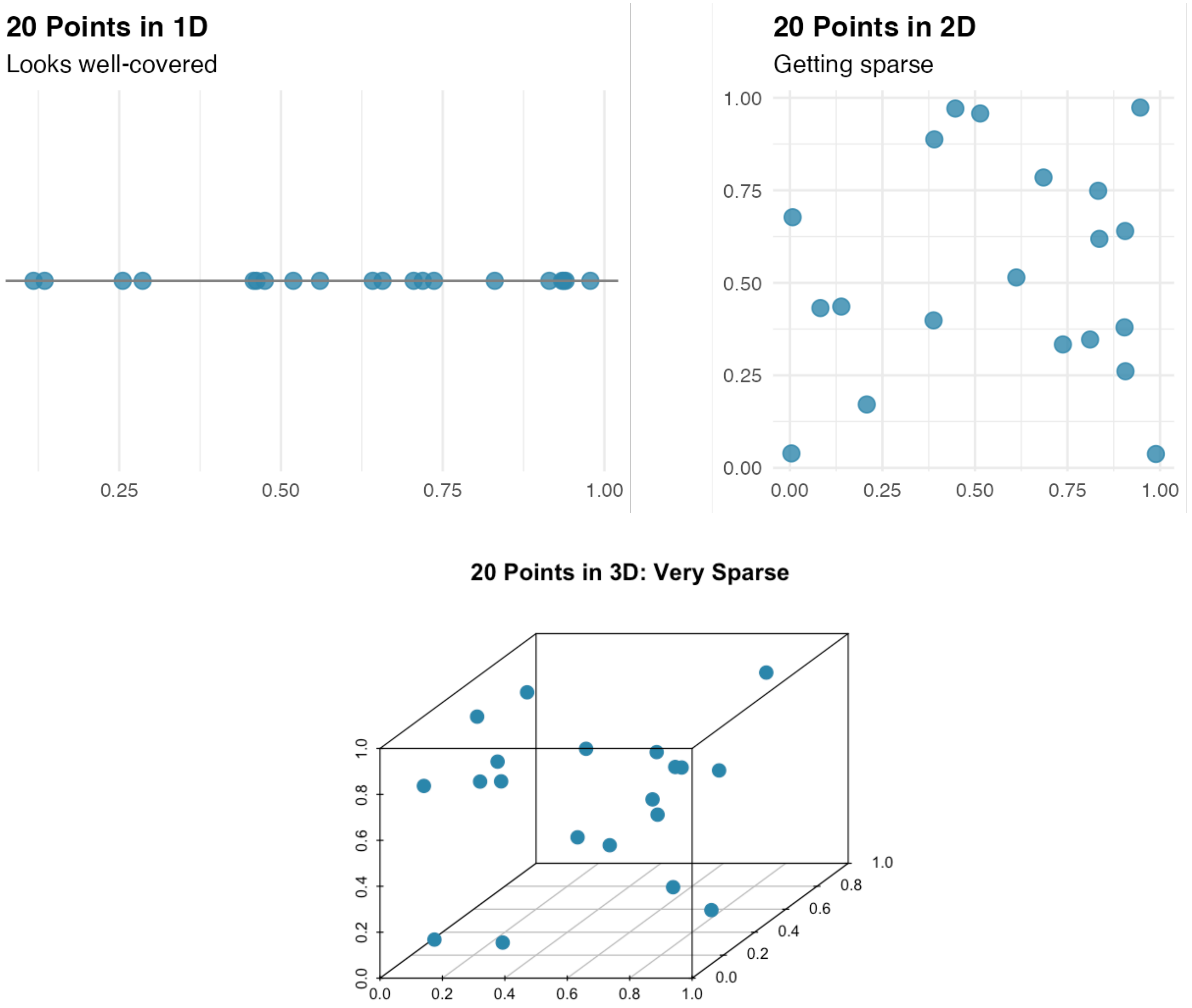
- High-dimensional data is very **sparse** (i.e. empty)
  - Say you have **50 datapoints**, spread across...
  - A **10x10 grid** → half full
  - A **10x10x10 cube** → 5% full



Most of the cube is empty. Now imagine 100D...

The "Curse of Dimensionality"

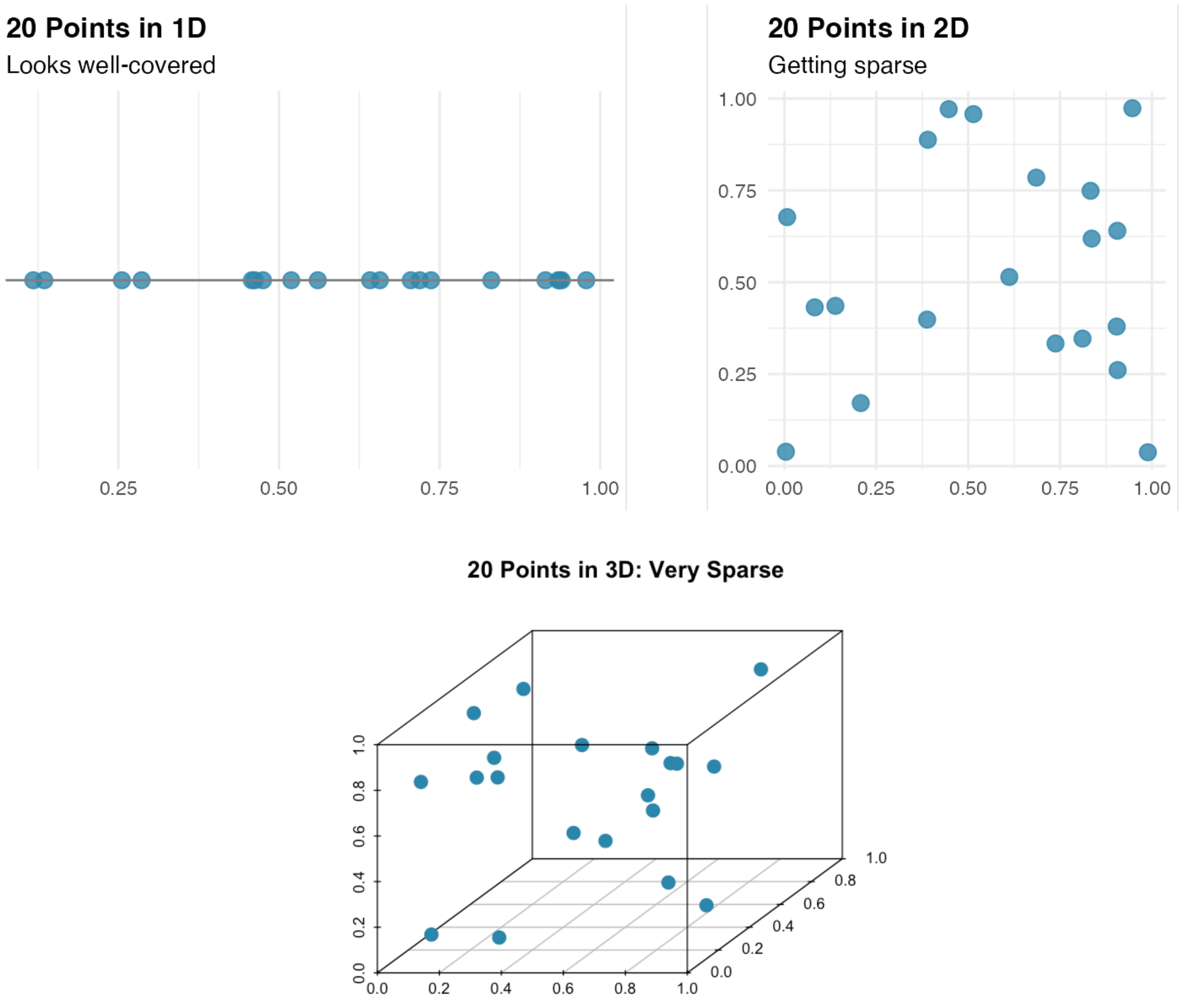
- High-dimensional data is very **sparse** (i.e. empty)
  - Say you have **50 datapoints**, spread across...
  - A **10x10 grid** → half full
  - A **10x10x10 cube** → 5% full
  - A **10x10x10x10 hypercube** → **0.05% full**



Most of the cube is empty. Now imagine 100D...

The "Curse of Dimensionality"

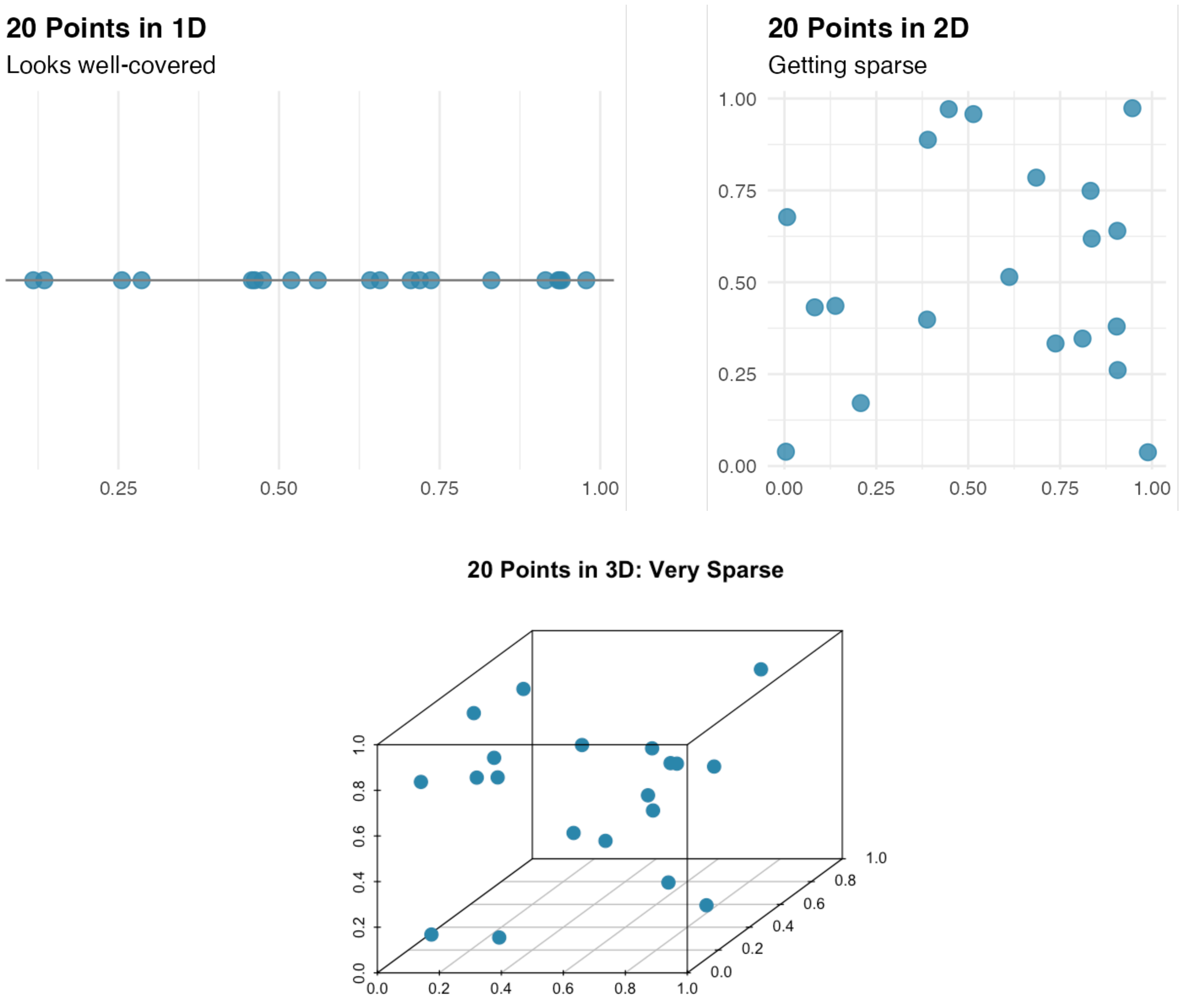
- High-dimensional data is very **sparse** (i.e. empty)
  - Say you have **50 datapoints**, spread across...
  - A **10x10 grid** → half full
  - A **10x10x10 cube** → 5% full
  - A **10x10x10x10 hypercube** → **0.05% full**
- Calculating **distances** between points becomes **meaningless**



Most of the cube is empty. Now imagine 100D...

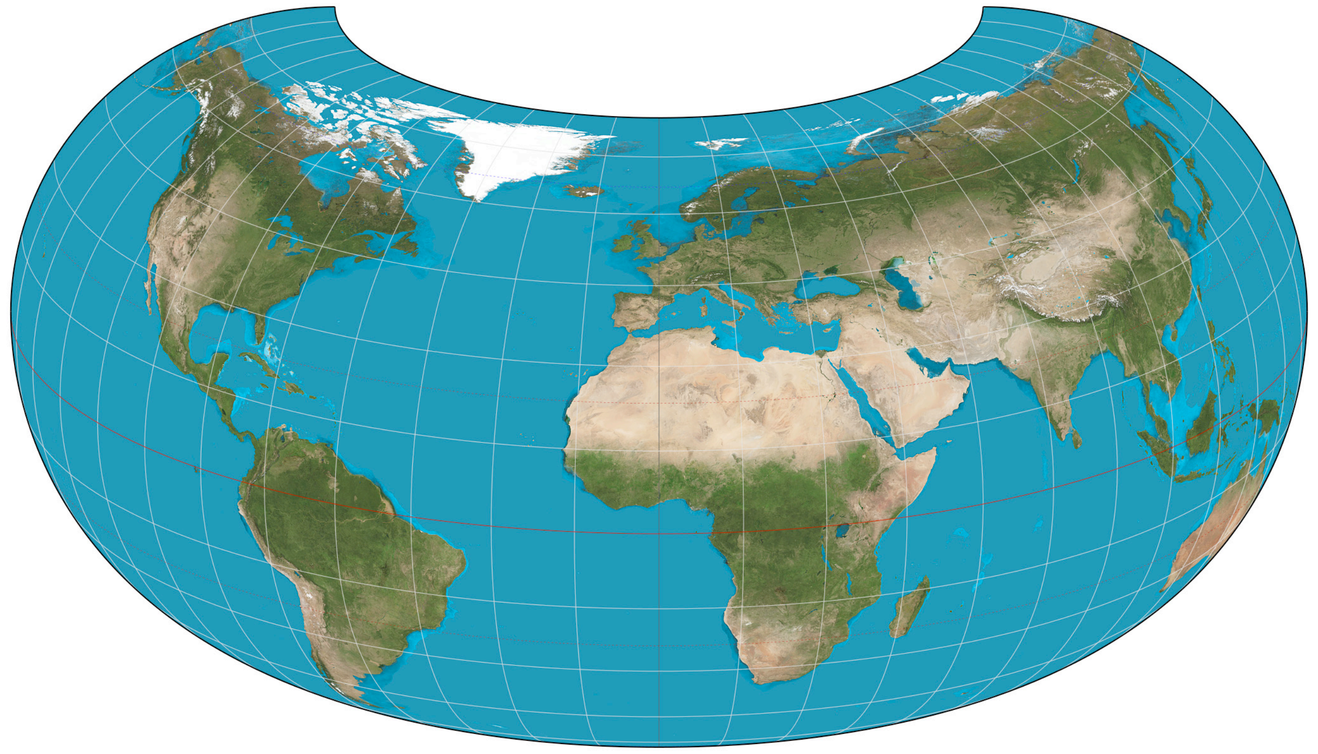
The "Curse of Dimensionality"

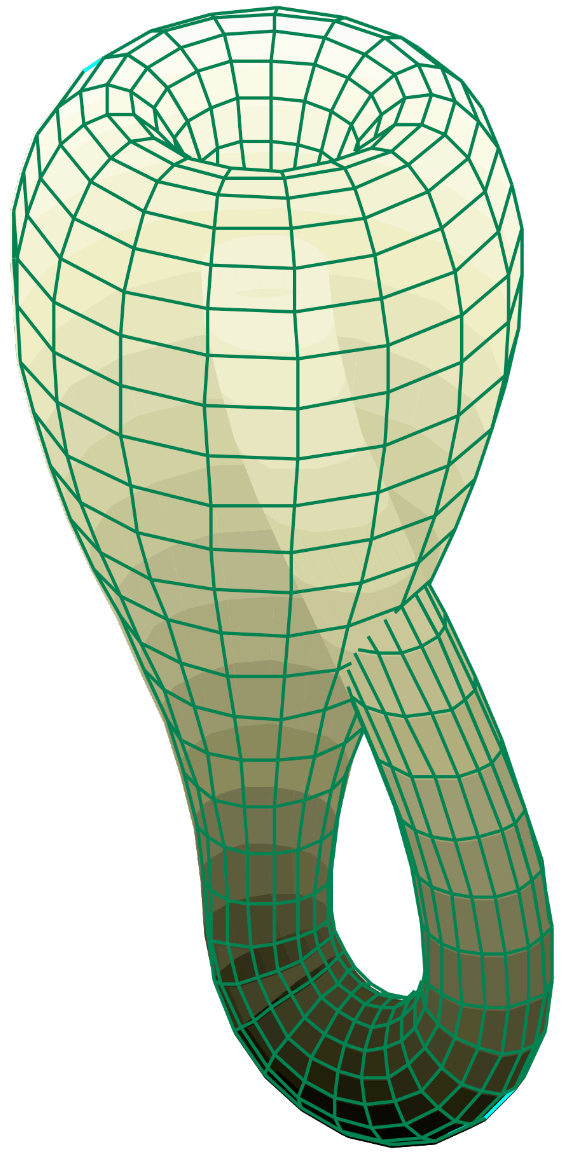
- High-dimensional data is very **sparse** (i.e. empty)
  - Say you have **50 datapoints**, spread across...
  - A **10x10 grid** → half full
  - A **10x10x10 cube** → 5% full
  - A **10x10x10x10 hypercube** → **0.05% full**
- Calculating **distances** between points becomes **meaningless**
- Models of this space become **complex** and **data-hungry**

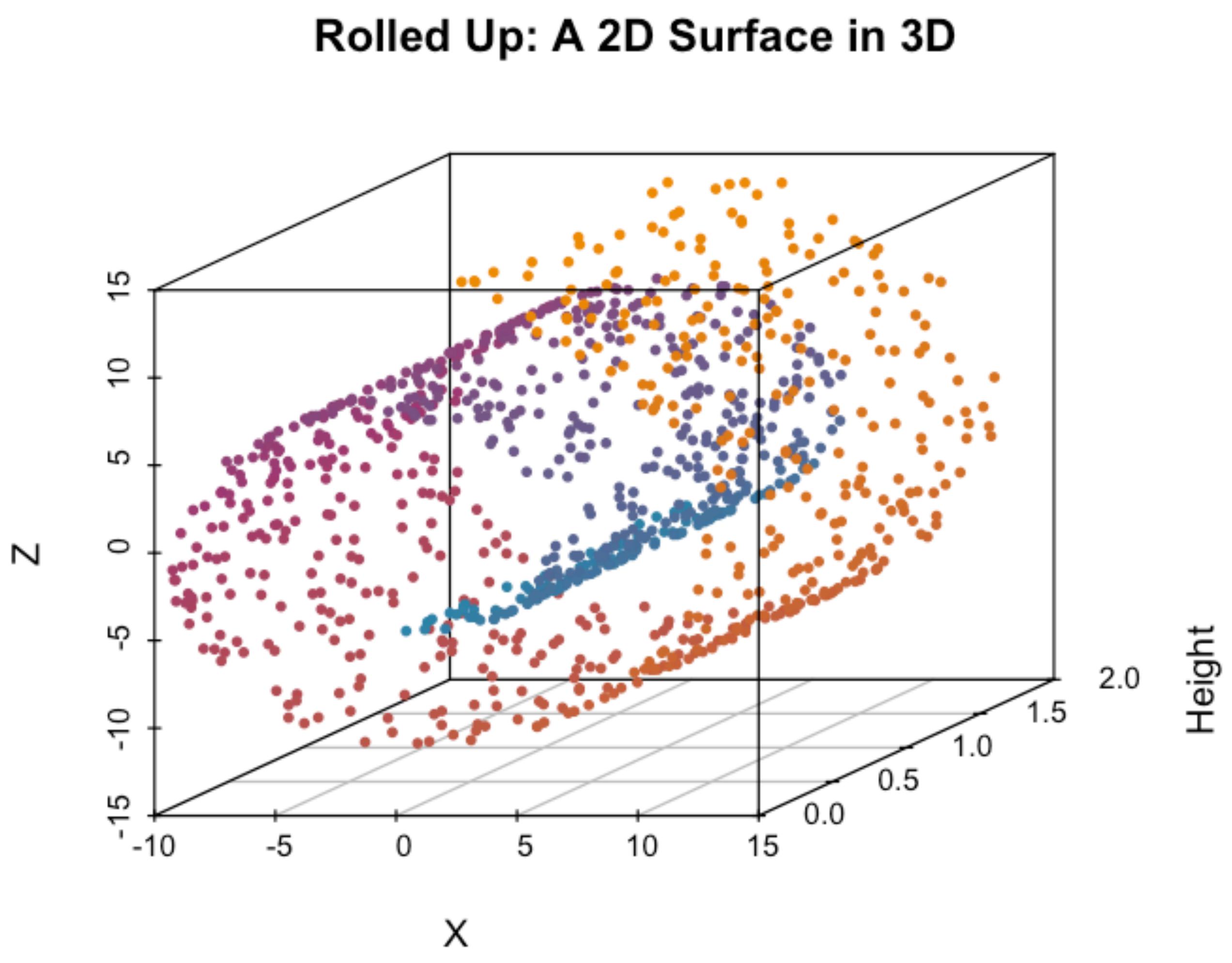


Most of the cube is empty. Now imagine 100D...

The "Manifold Hypothesis"

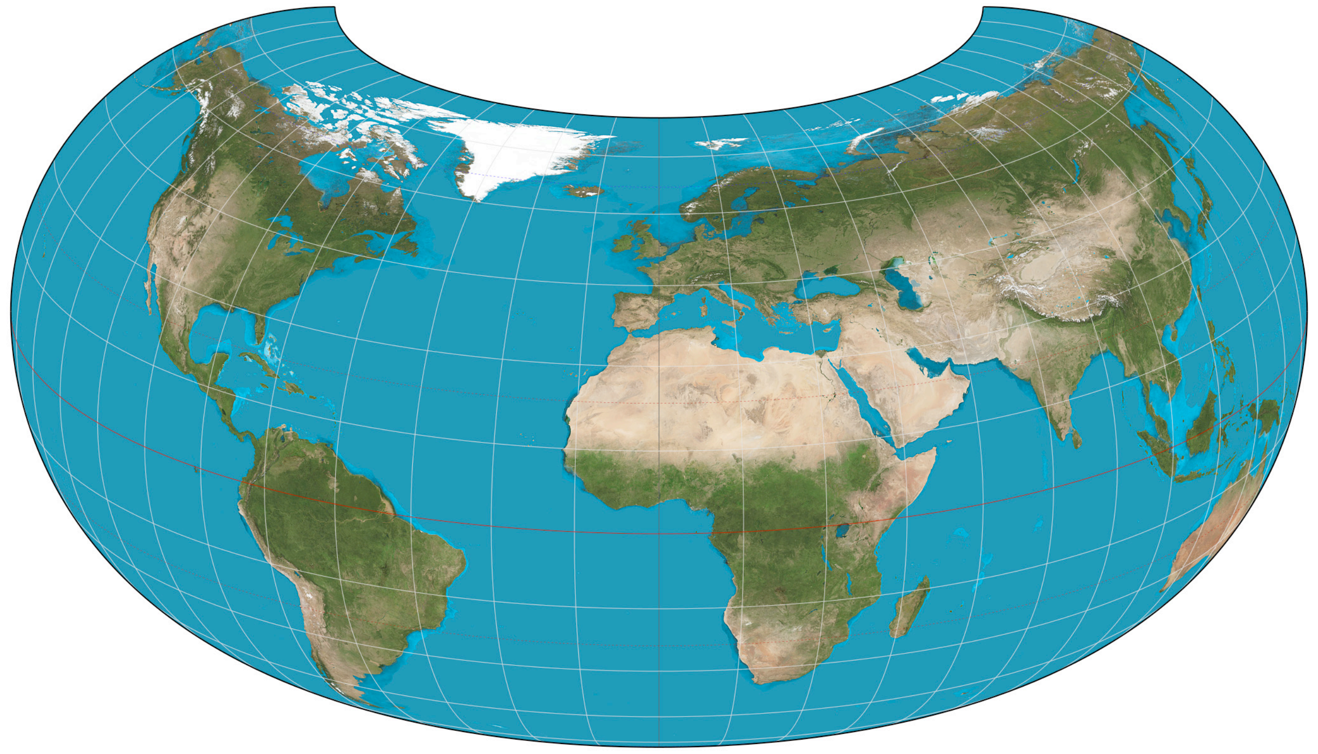


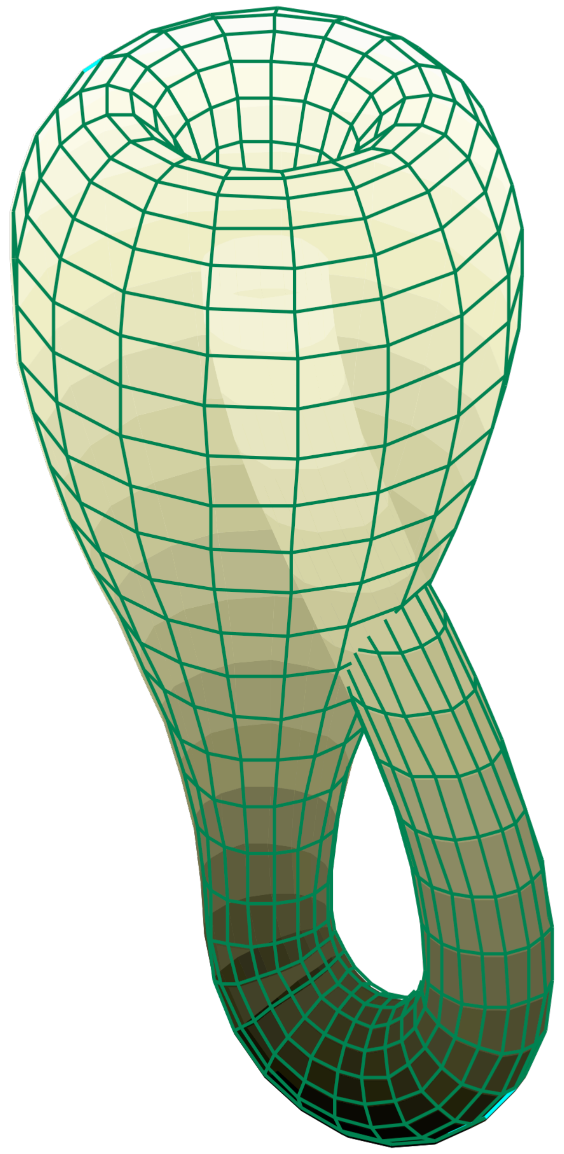


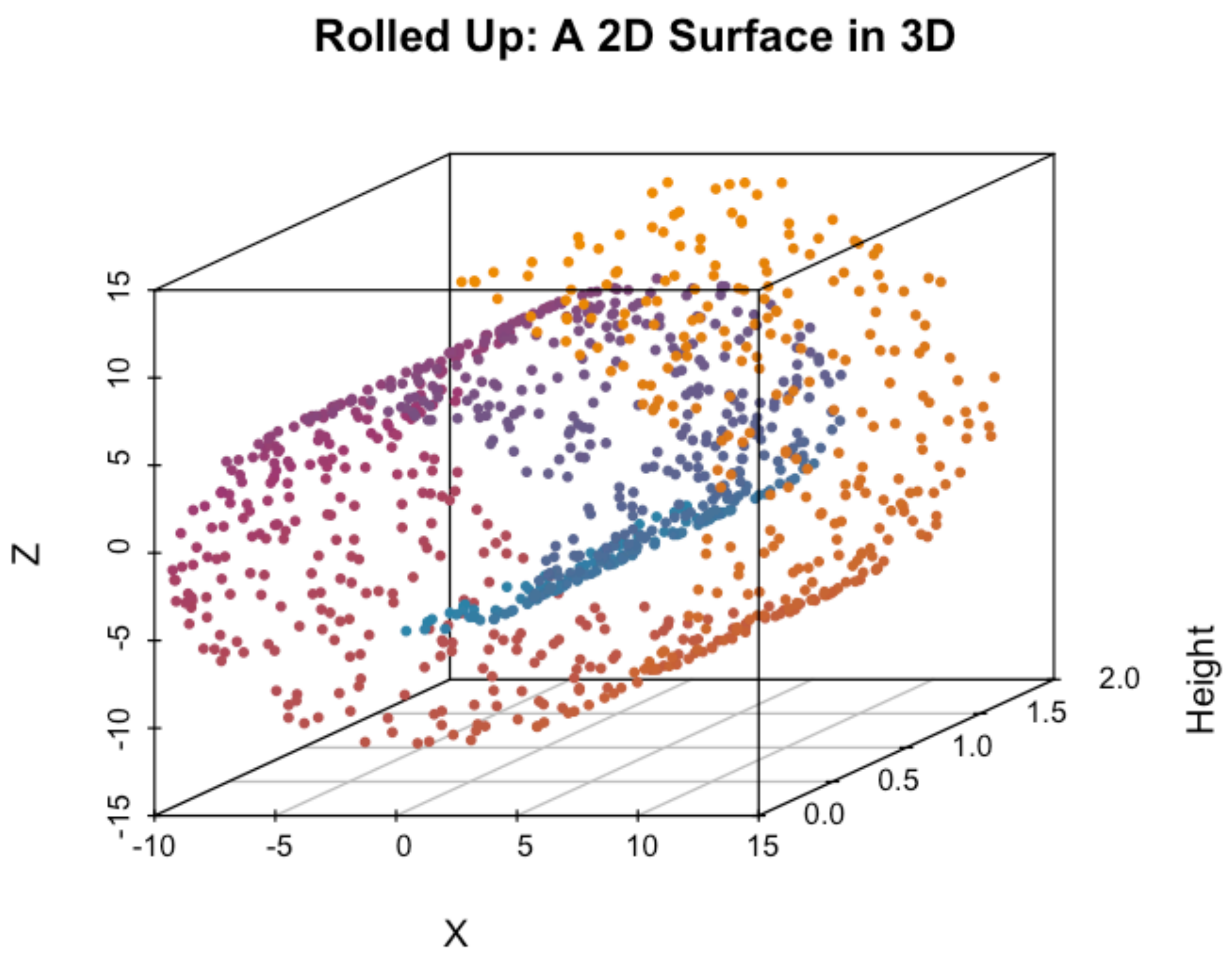


The "Manifold Hypothesis"

- Our hope: the interesting data patterns live on a **lower-dimensional manifold** within the high-dimensional space

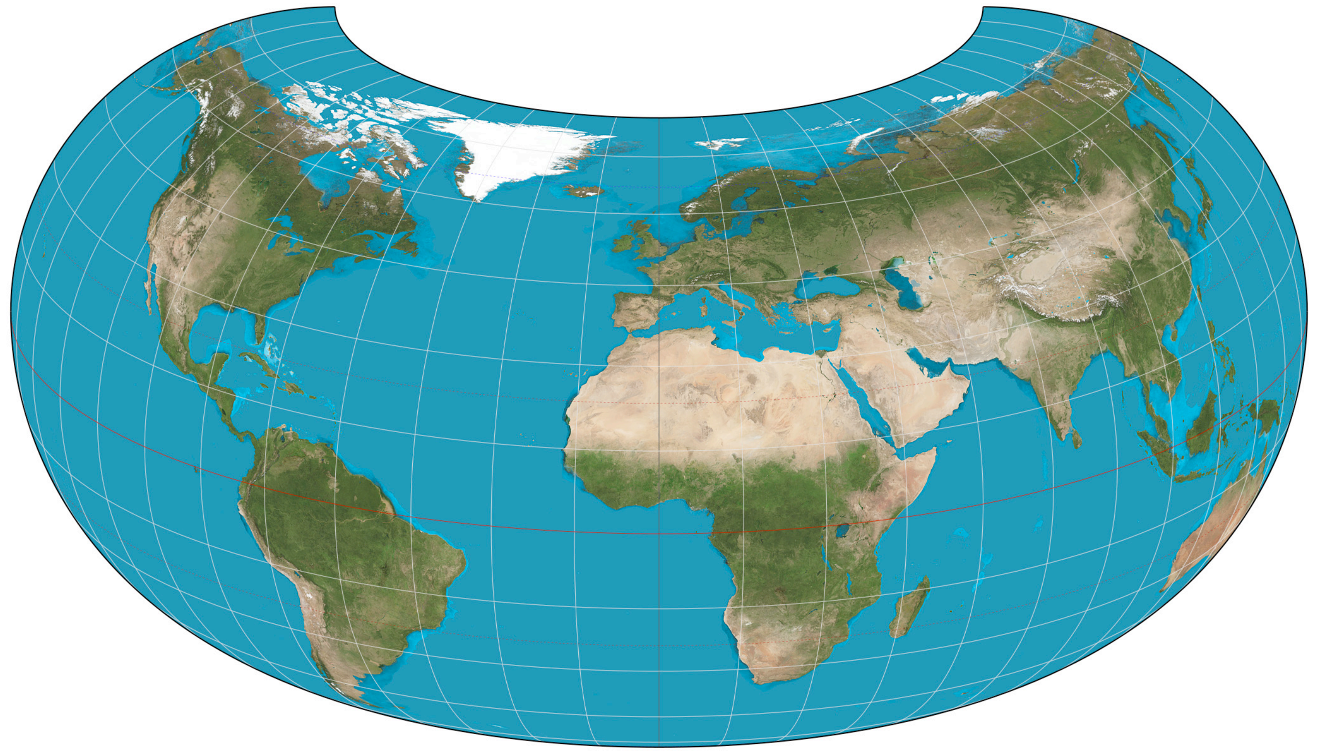


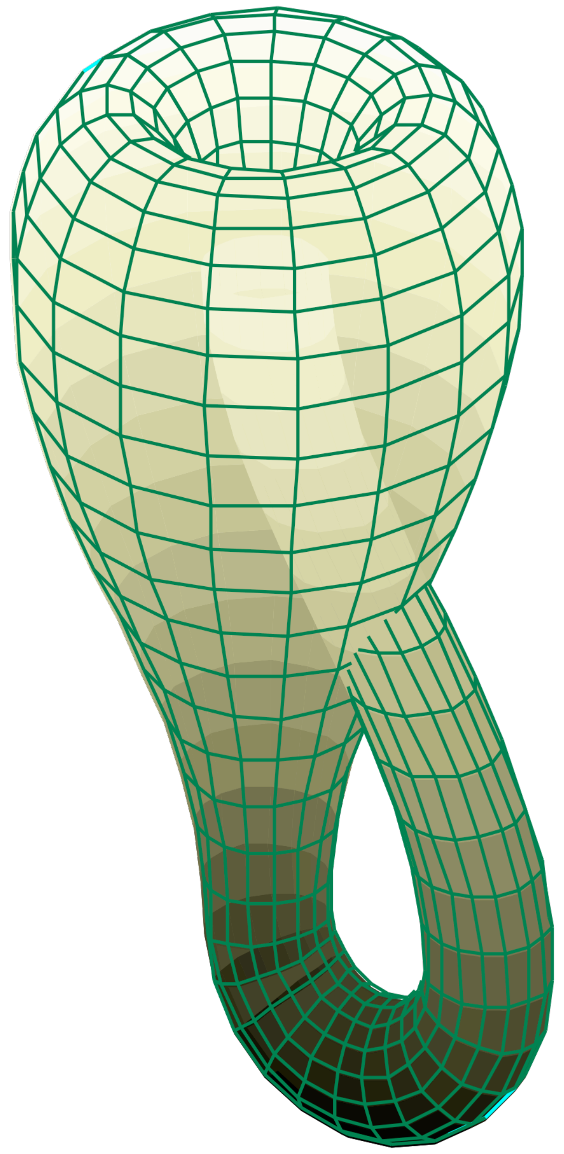


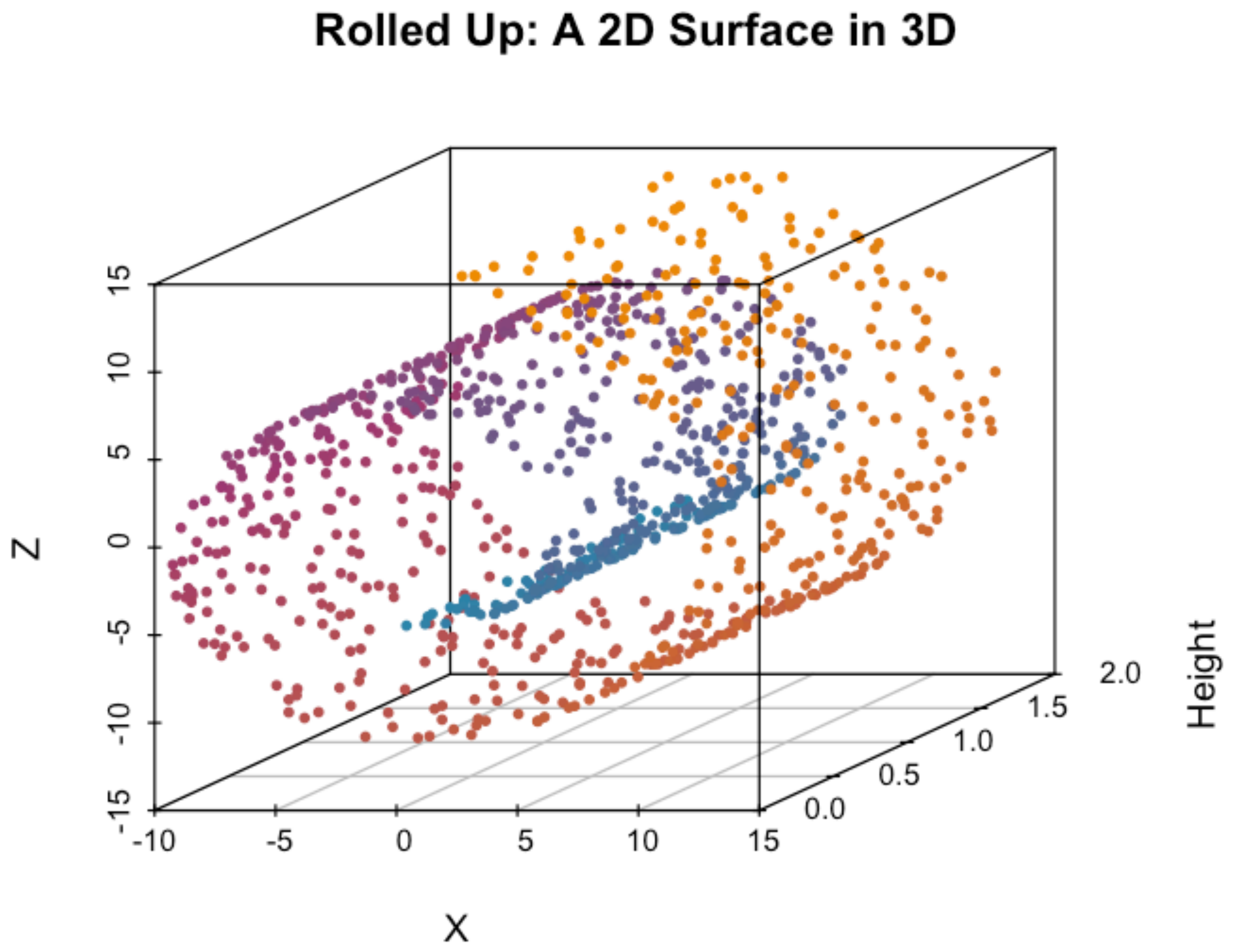


The "Manifold Hypothesis"

- Our hope: the interesting data patterns live on a **lower-dimensional manifold** within the high-dimensional space
  - ...what is a manifold? Think of it as a **possibly-curved surface or shape**

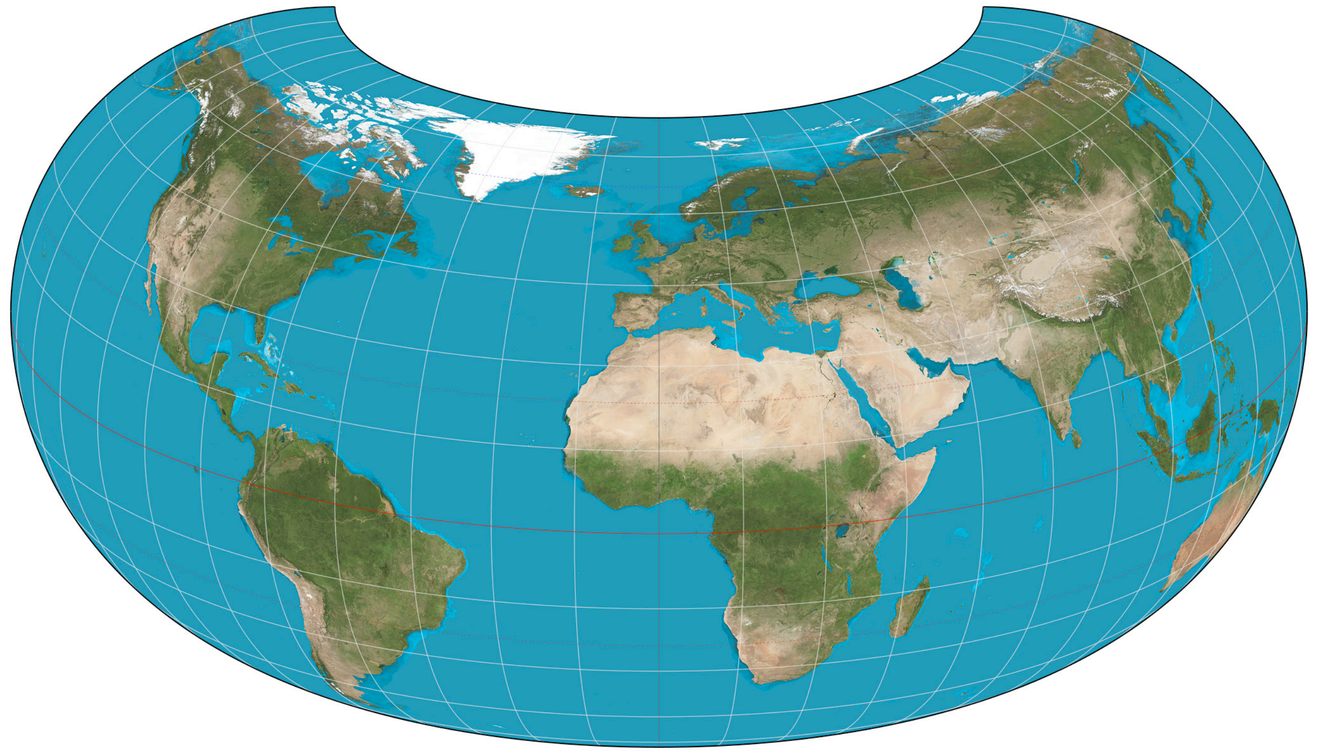


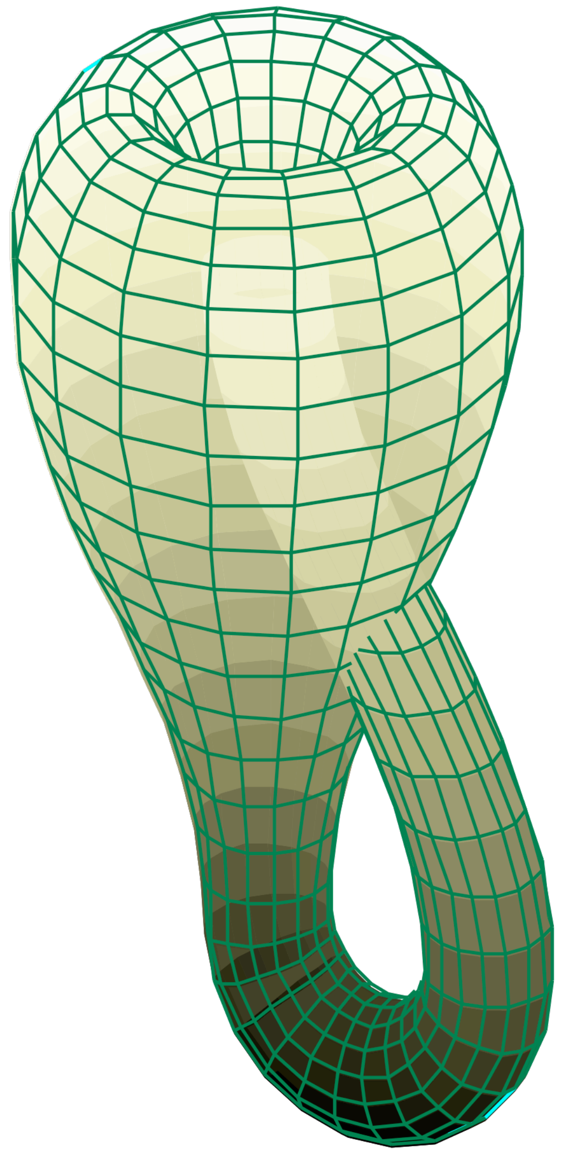


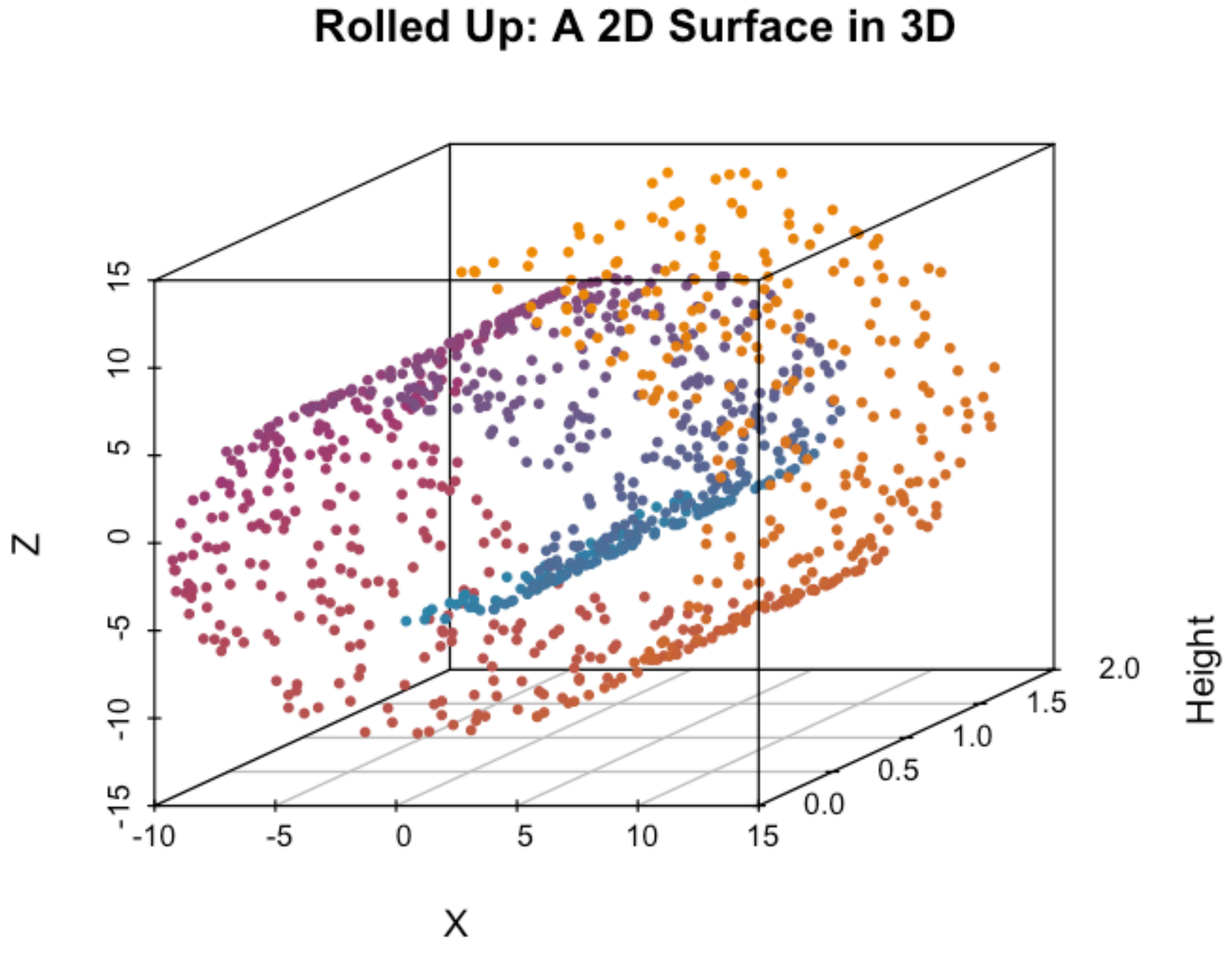


The "Manifold Hypothesis"

- Our hope: the interesting data patterns live on a **lower-dimensional manifold** within the high-dimensional space
  - ...what is a manifold? Think of it as a **possibly-curved surface or shape**
  - They also have **dimensionality** that let us **reason** about them like **normal (Euclidian) space**... at least if you **zoom in close enough**

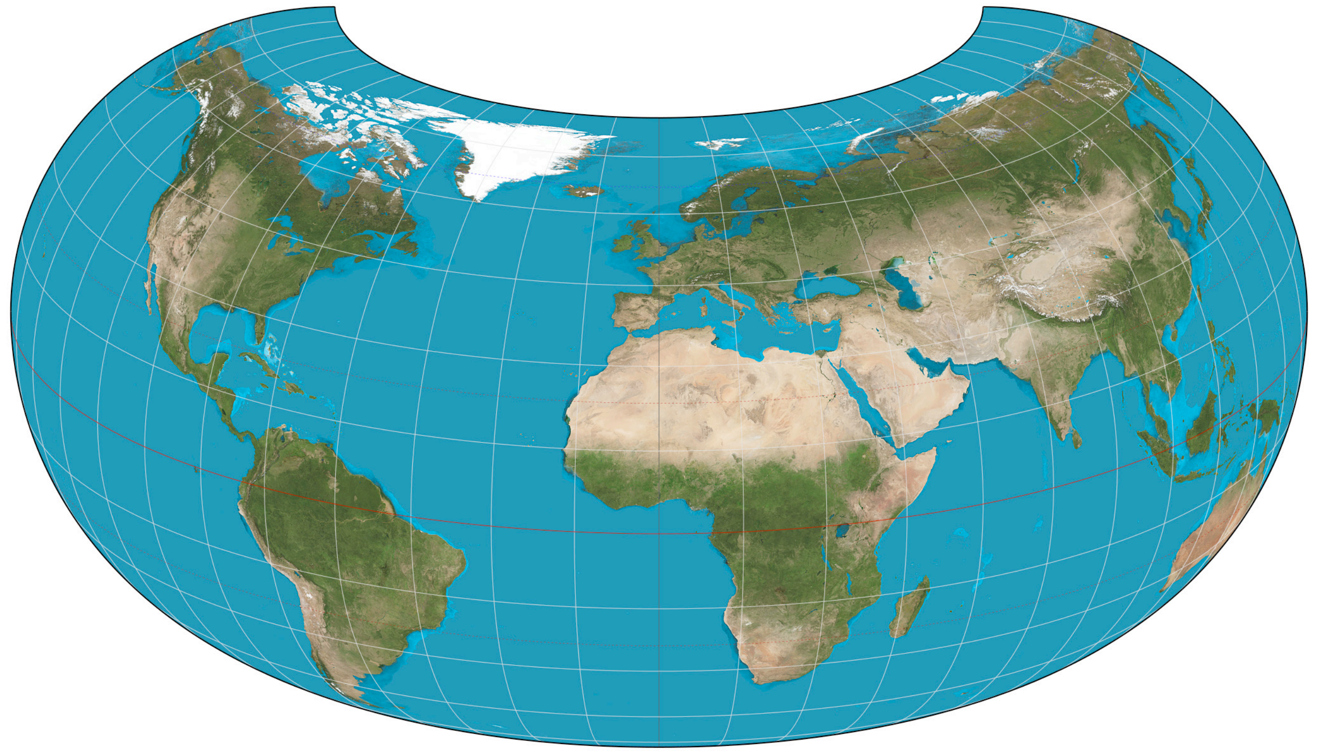


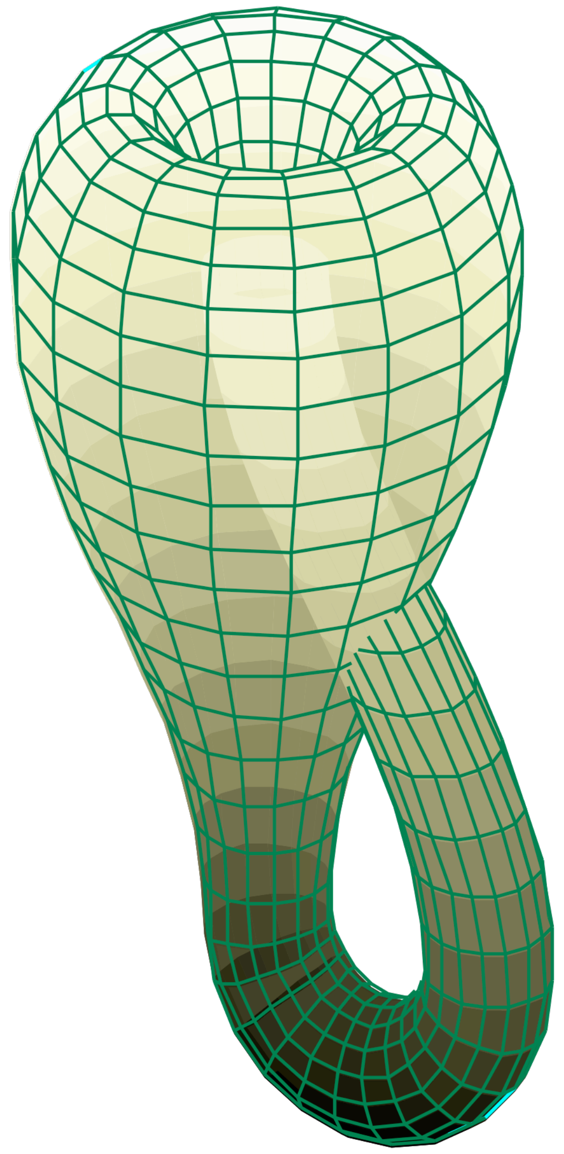


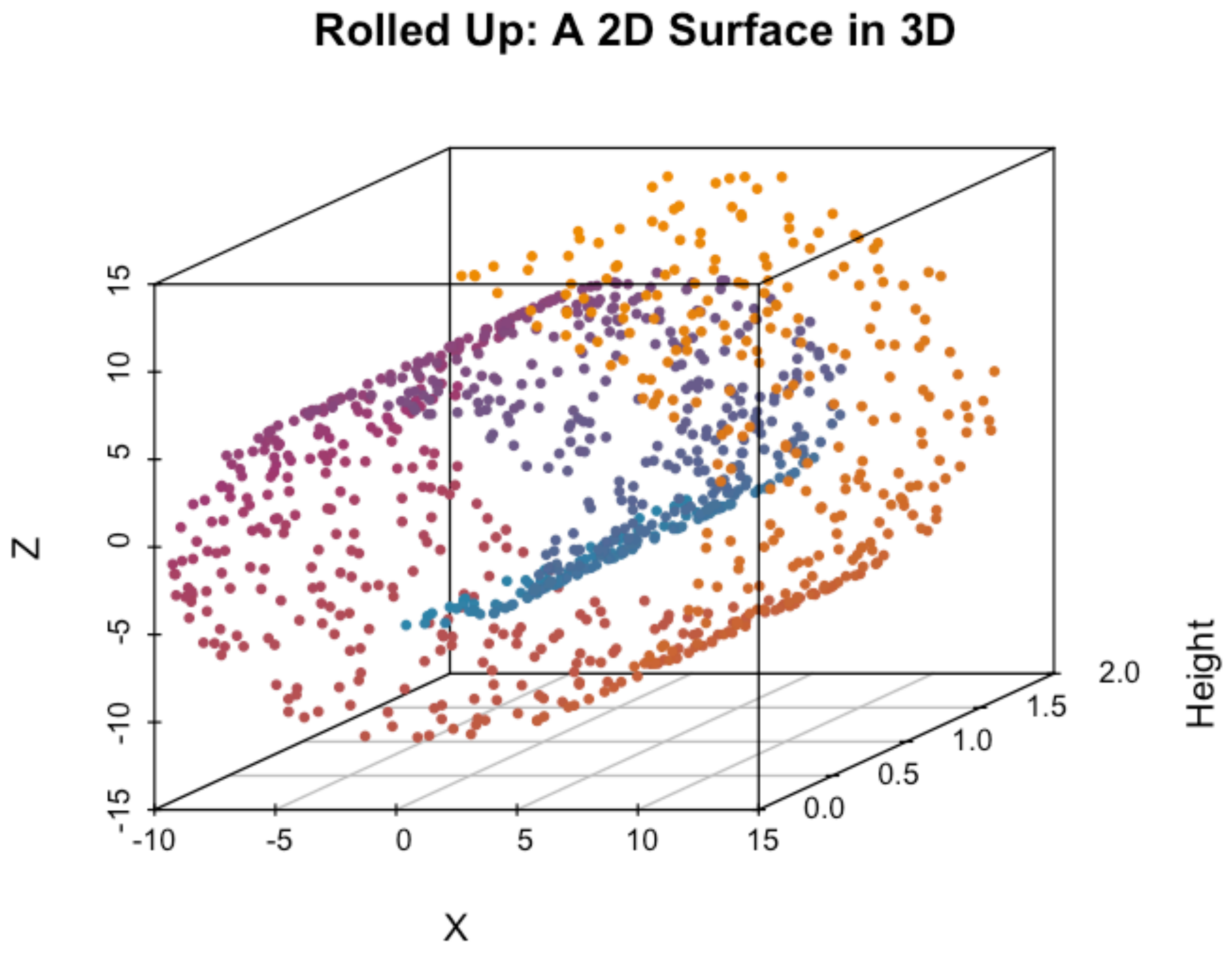


The "Manifold Hypothesis"

- Our hope: the interesting data patterns live on a **lower-dimensional manifold** within the high-dimensional space
  - ...what is a manifold? Think of it as a **possibly-curved surface or shape**
  - They also have **dimensionality** that let us **reason** about them like **normal (Euclidian) space**... at least if you **zoom in close enough**
  - Example: the **Earth's surface** can be **reasoned about in 2D** even though it curves

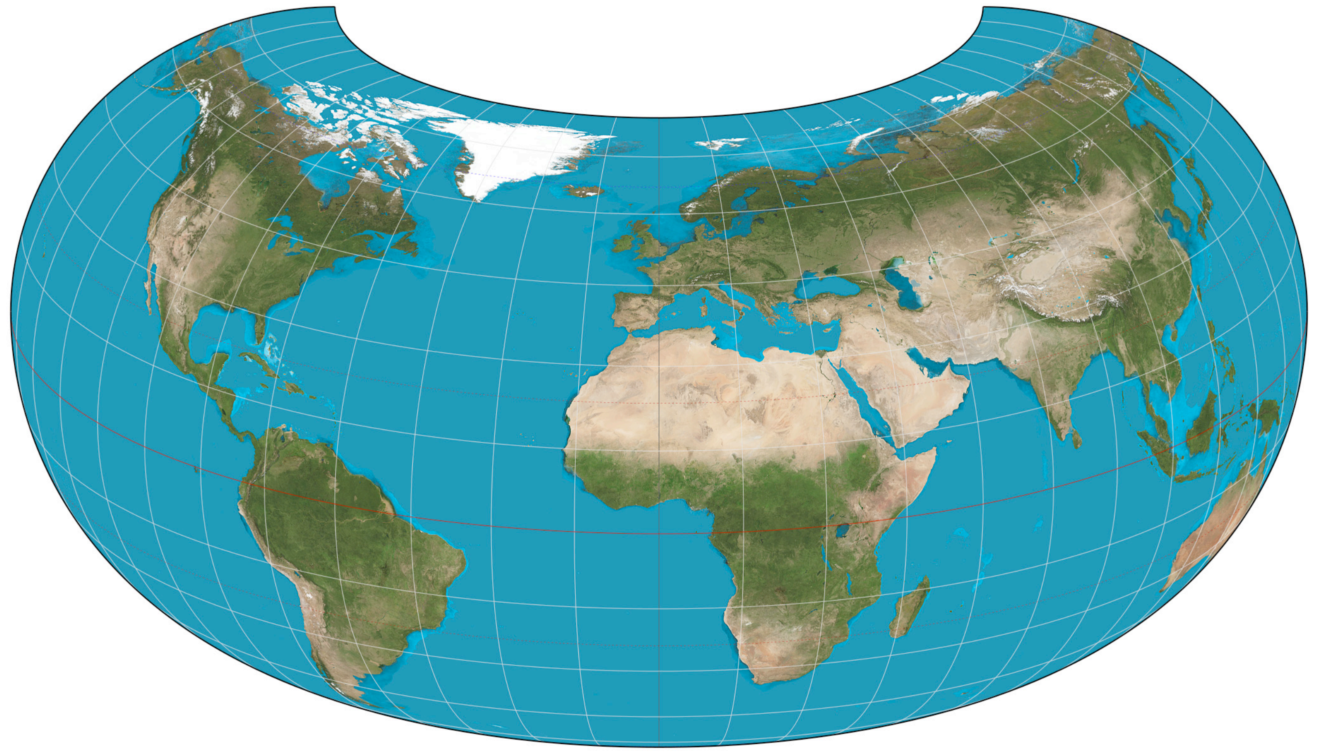


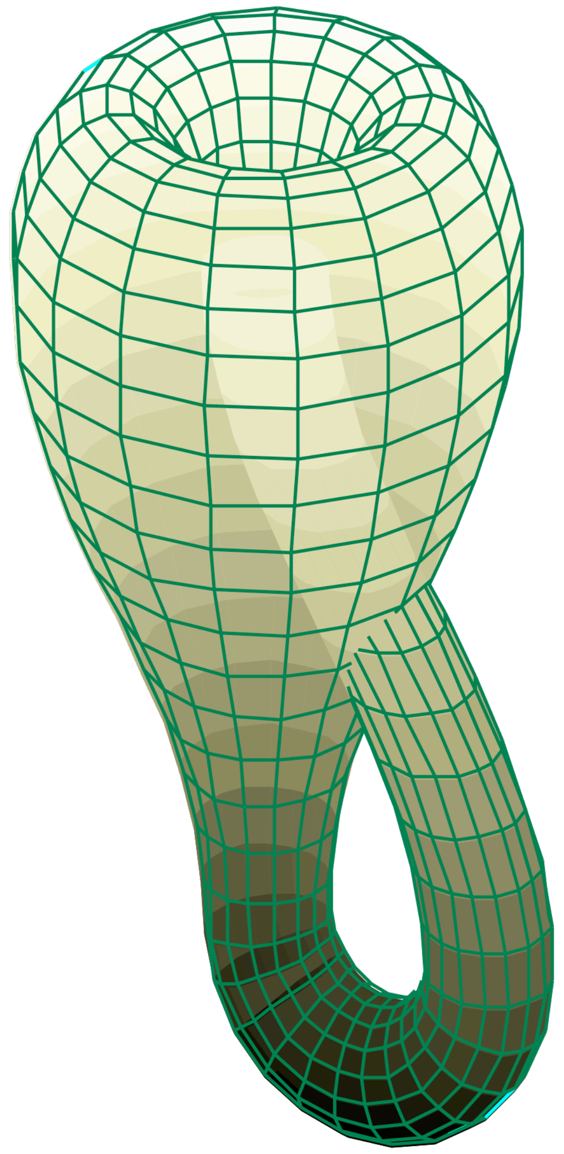


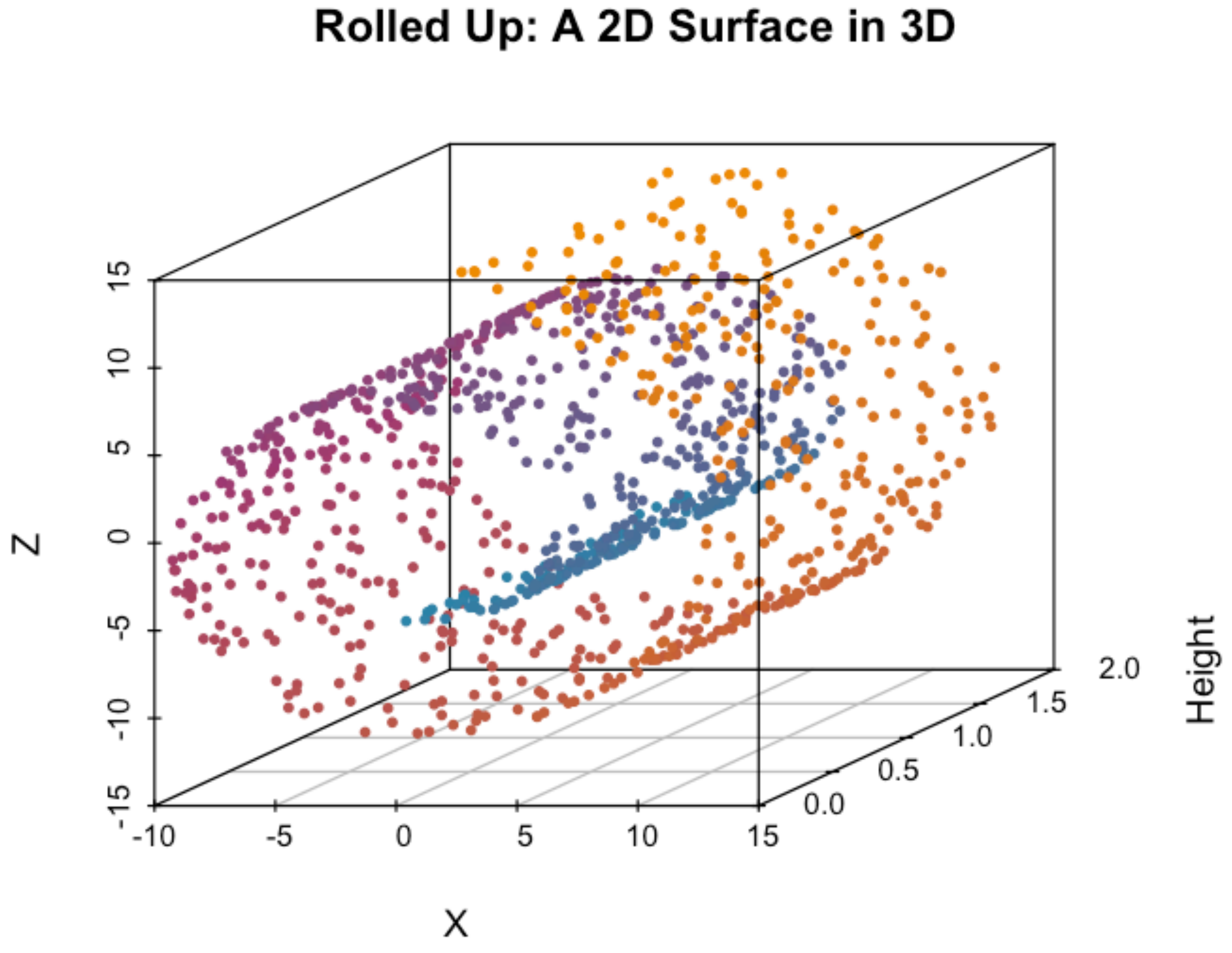


The "Manifold Hypothesis"

- Our hope: the interesting data patterns live on a **lower-dimensional manifold** within the high-dimensional space
  - ...what is a manifold? Think of it as a **possibly-curved surface or shape**
  - They also have **dimensionality** that let us **reason** about them like **normal (Euclidian) space**... at least if you **zoom in close enough**
  - Example: the **Earth's surface** can be **reasoned about in 2D** even though it curves
  - Another example: a **rolled-up picture**

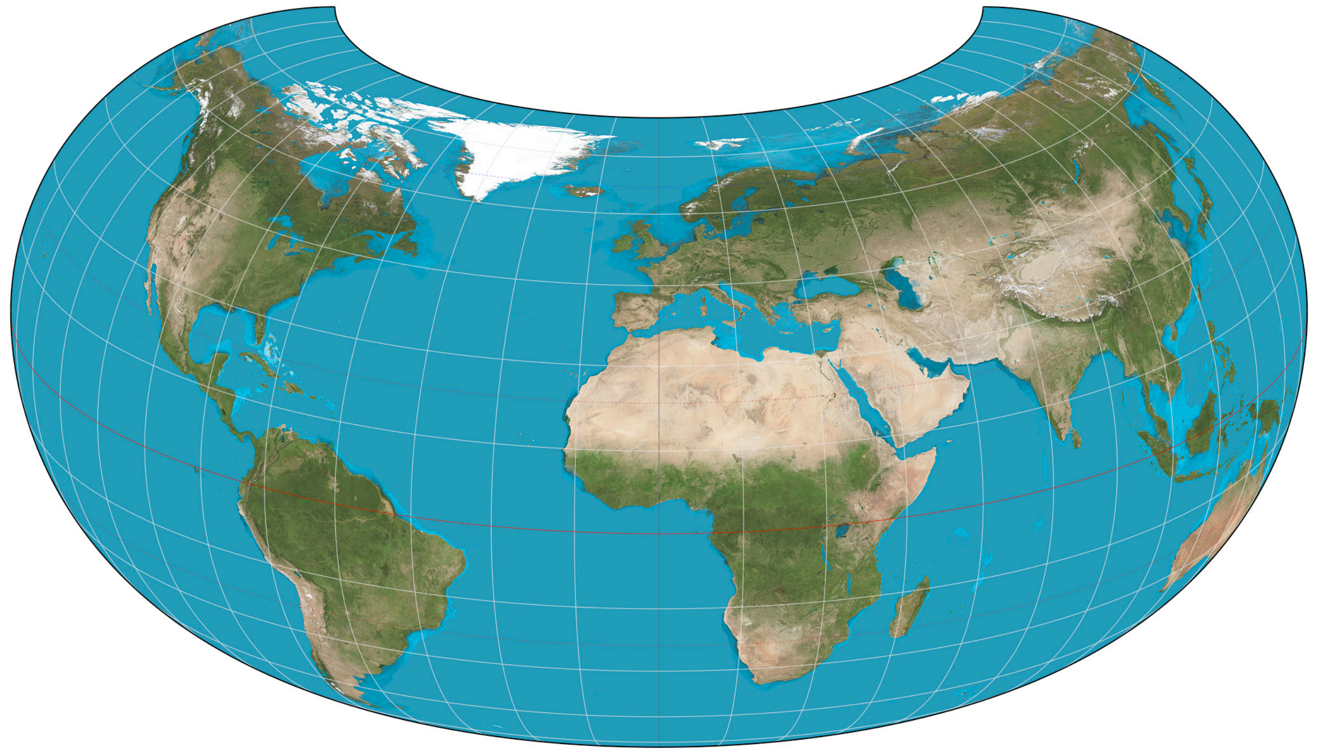


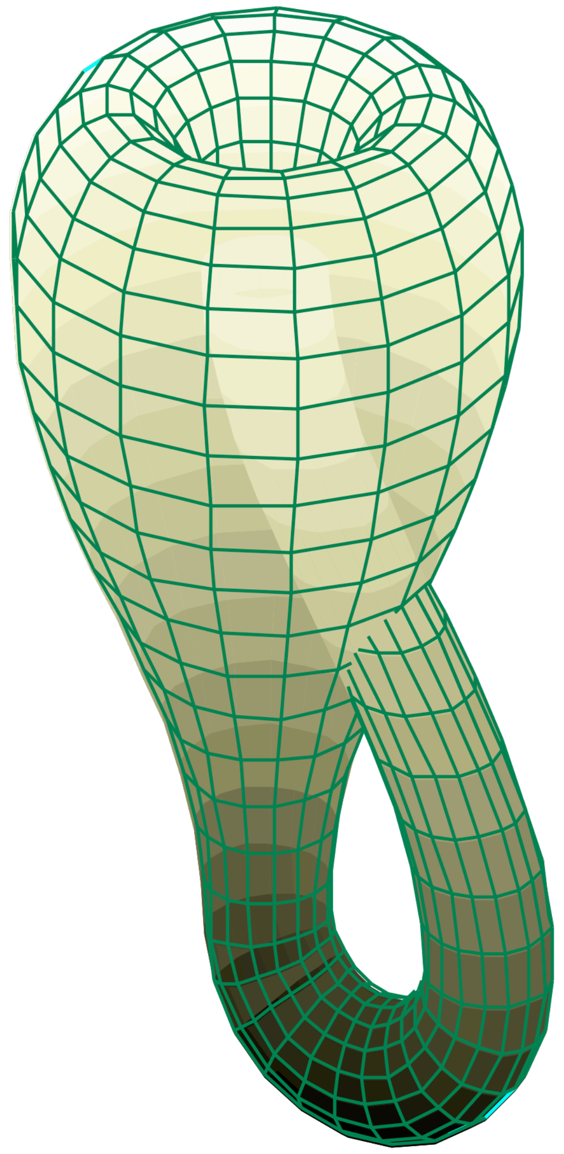


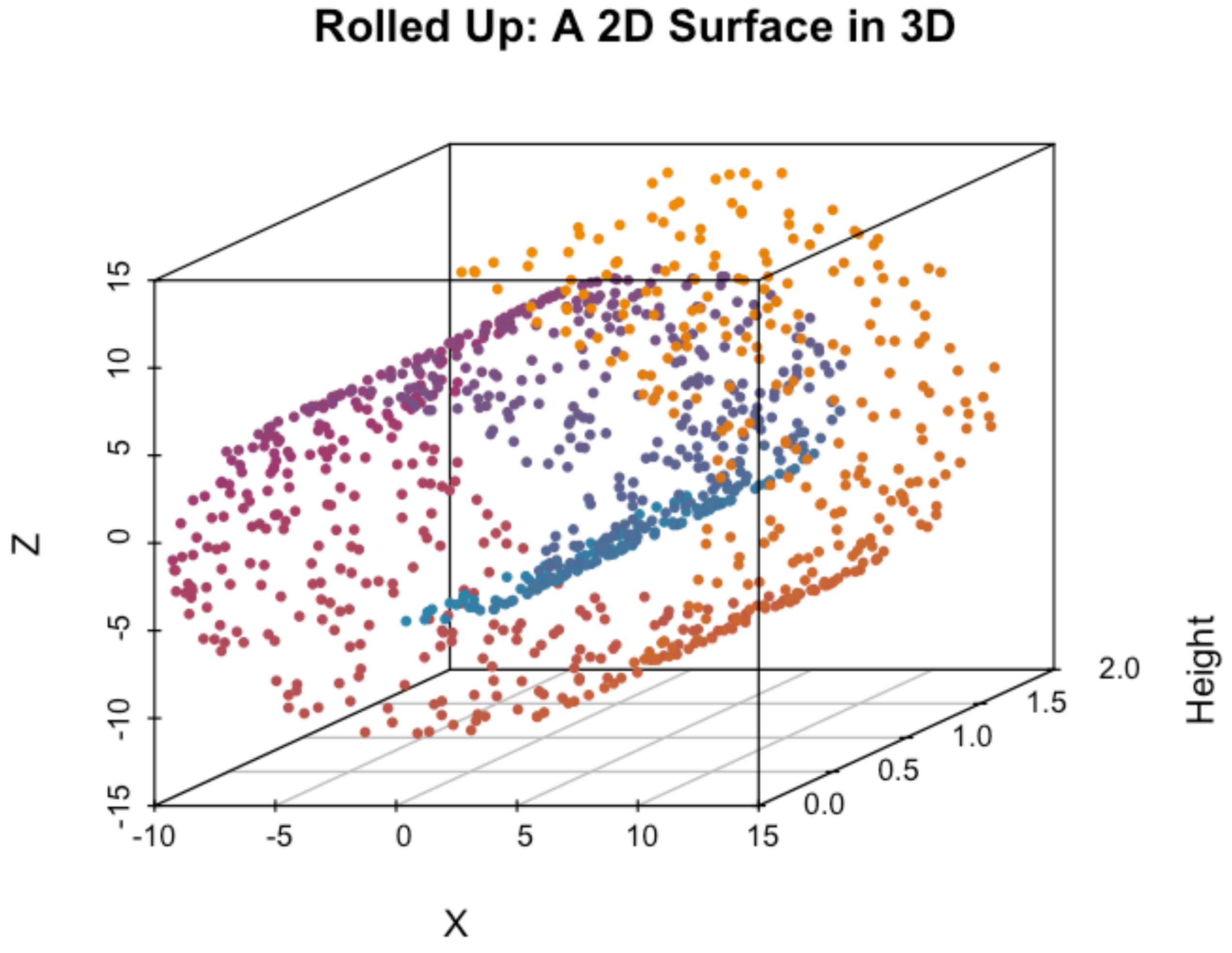


The "Manifold Hypothesis"

- Our hope: the interesting data patterns live on a **lower-dimensional manifold** within the high-dimensional space
  - ...what is a manifold? Think of it as a **possibly-curved surface or shape**
  - They also have **dimensionality** that let us **reason** about them like **normal (Euclidian) space**... at least if you **zoom in close enough**
  - Example: the **Earth's surface** can be **reasoned about in 2D** even though it curves
  - Another example: a **rolled-up picture**
- Goal: find the **dimensions that really matter**

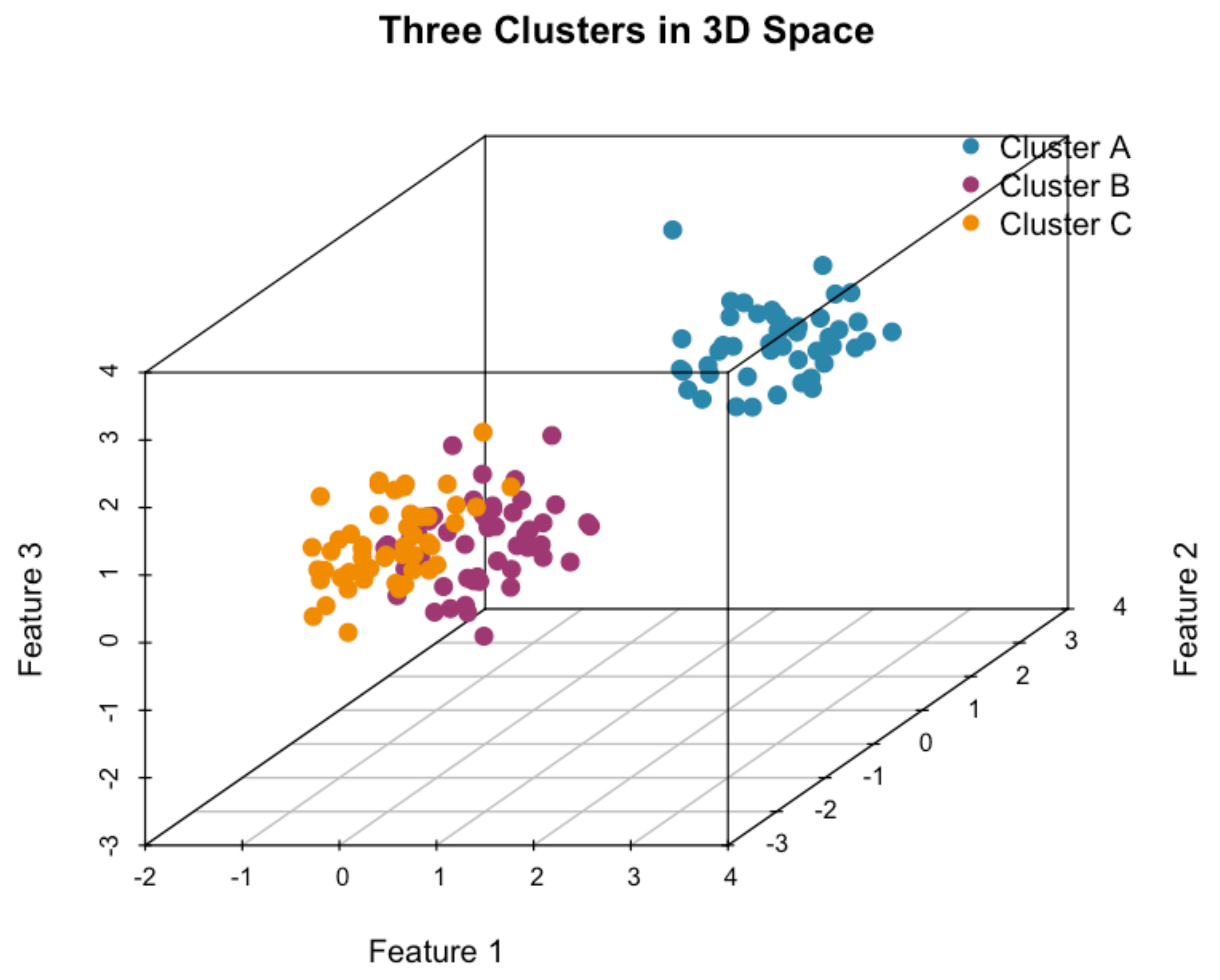


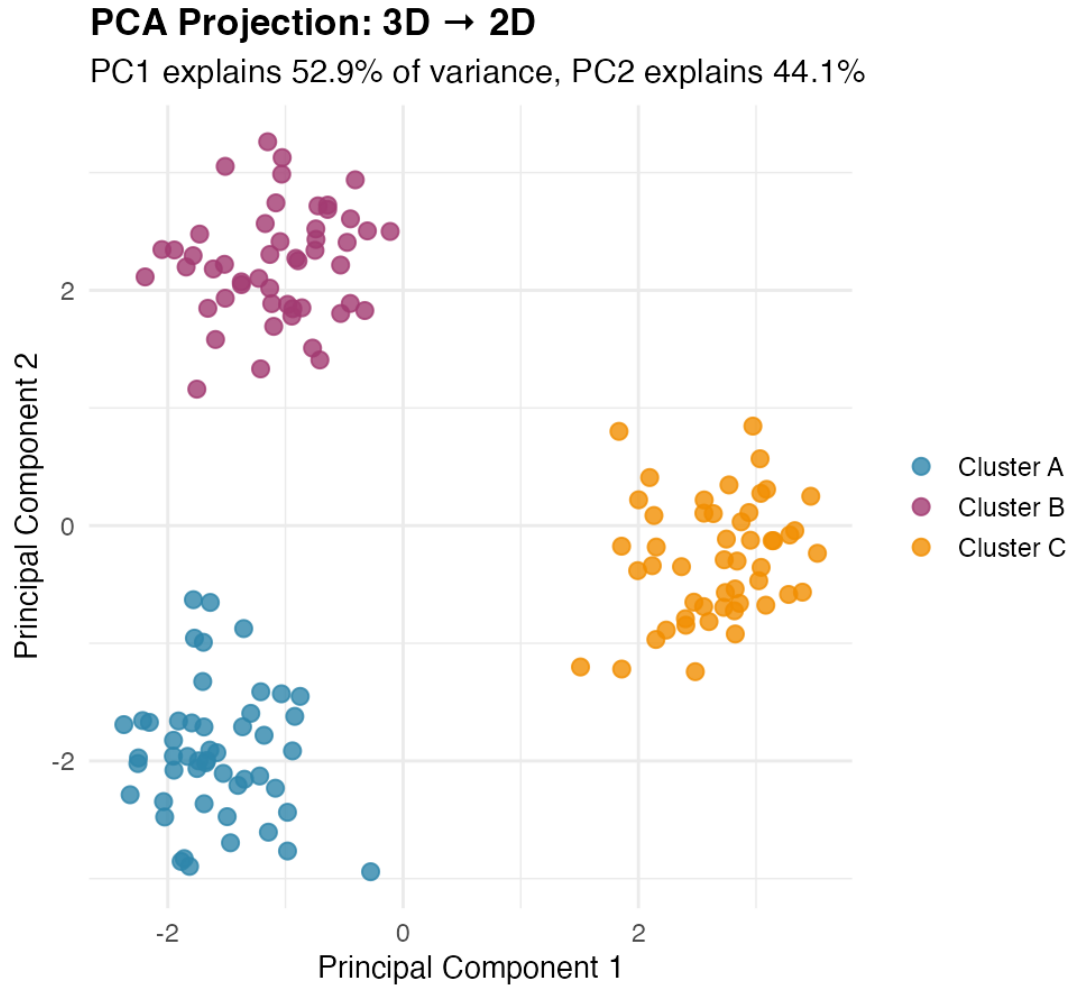




Principal Component Analysis (PCA)

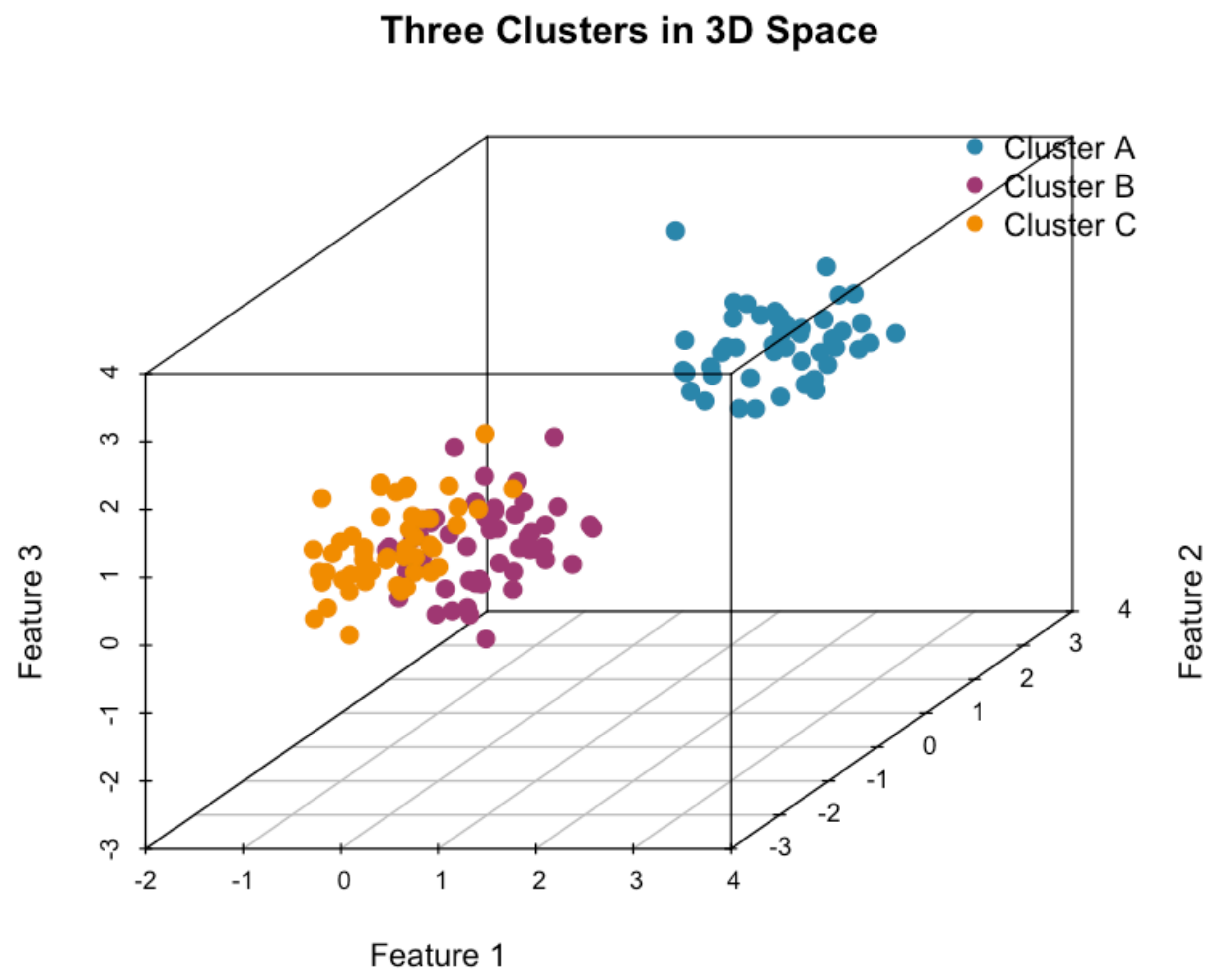
PCA Generally

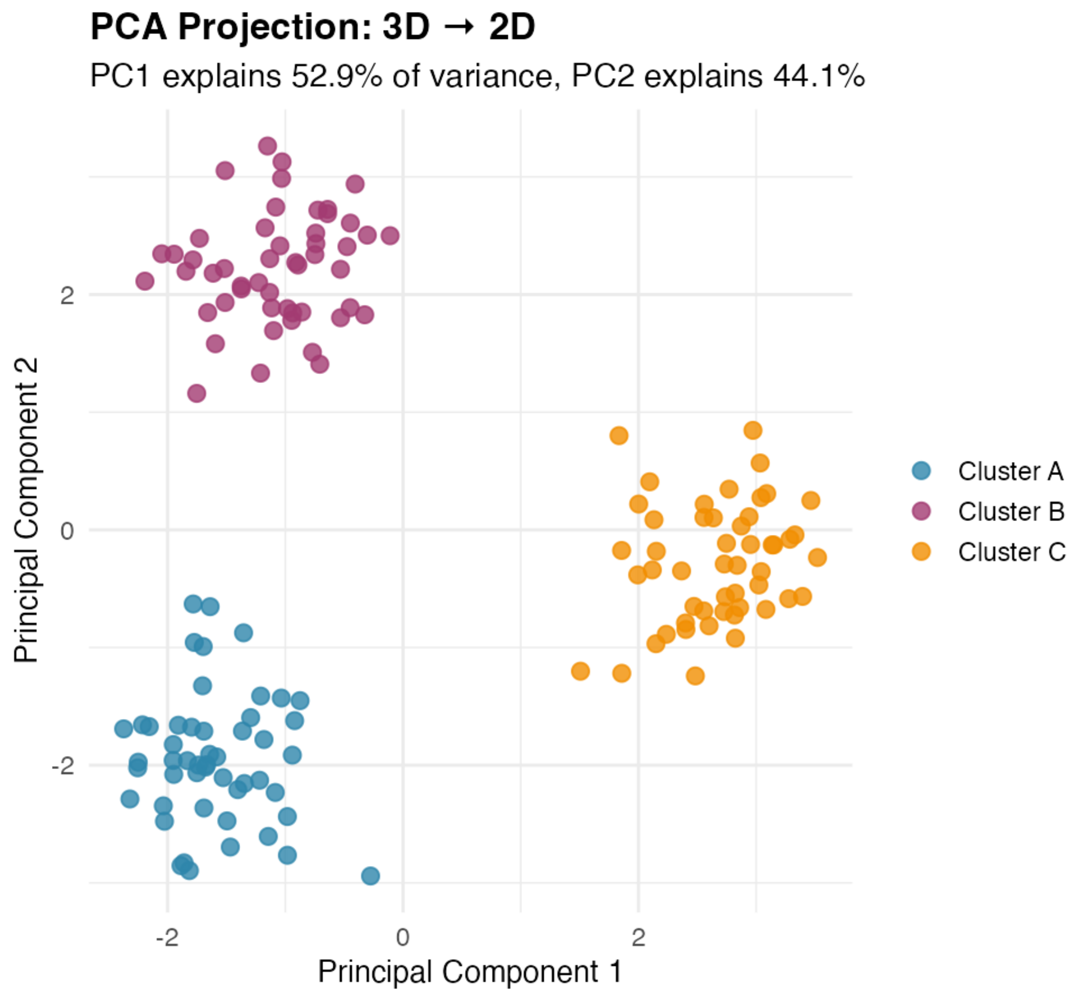




PCA Generally

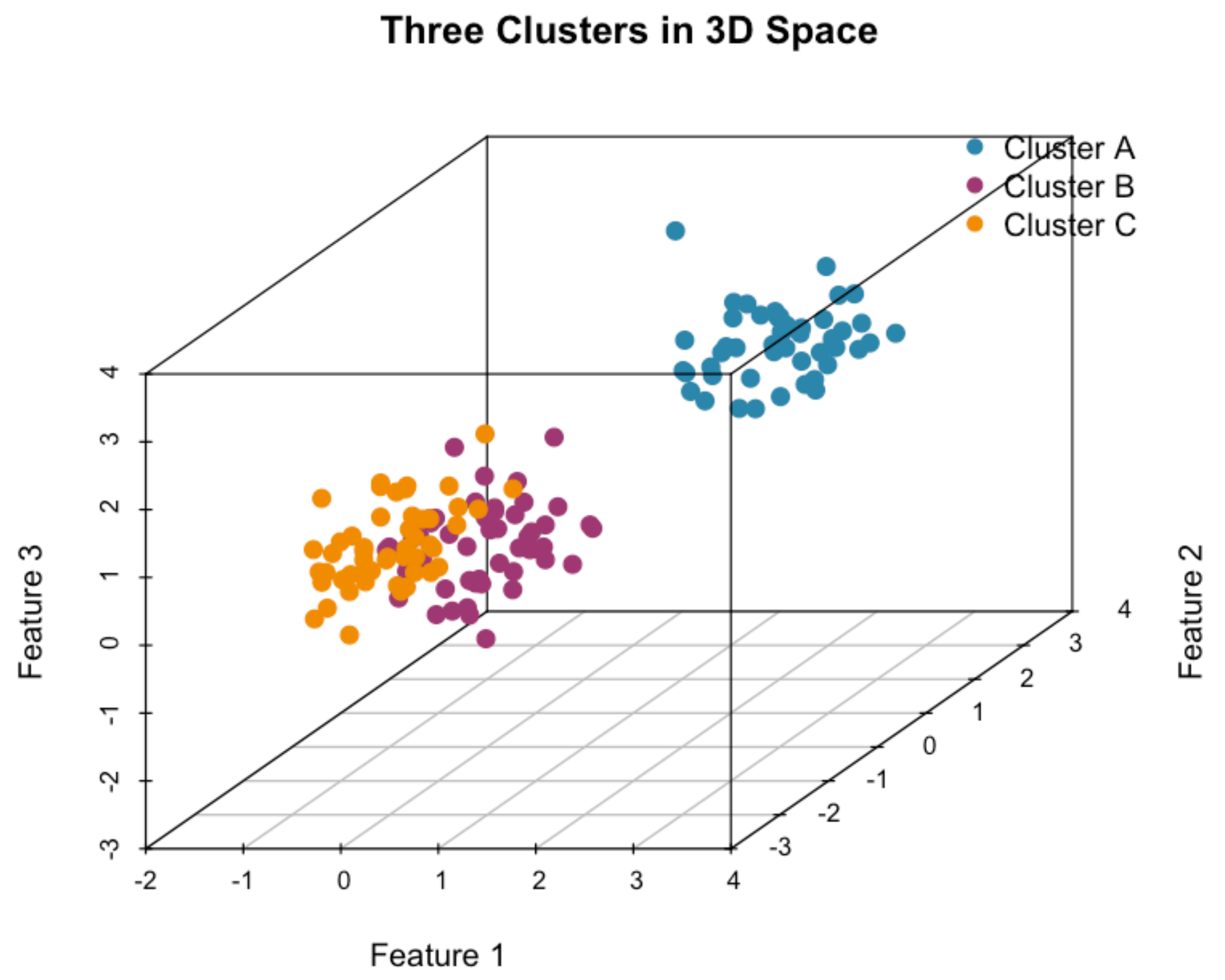
- Assume we have n datapoints of dimension d
  - I.e. a matrix X of size $d \times n$

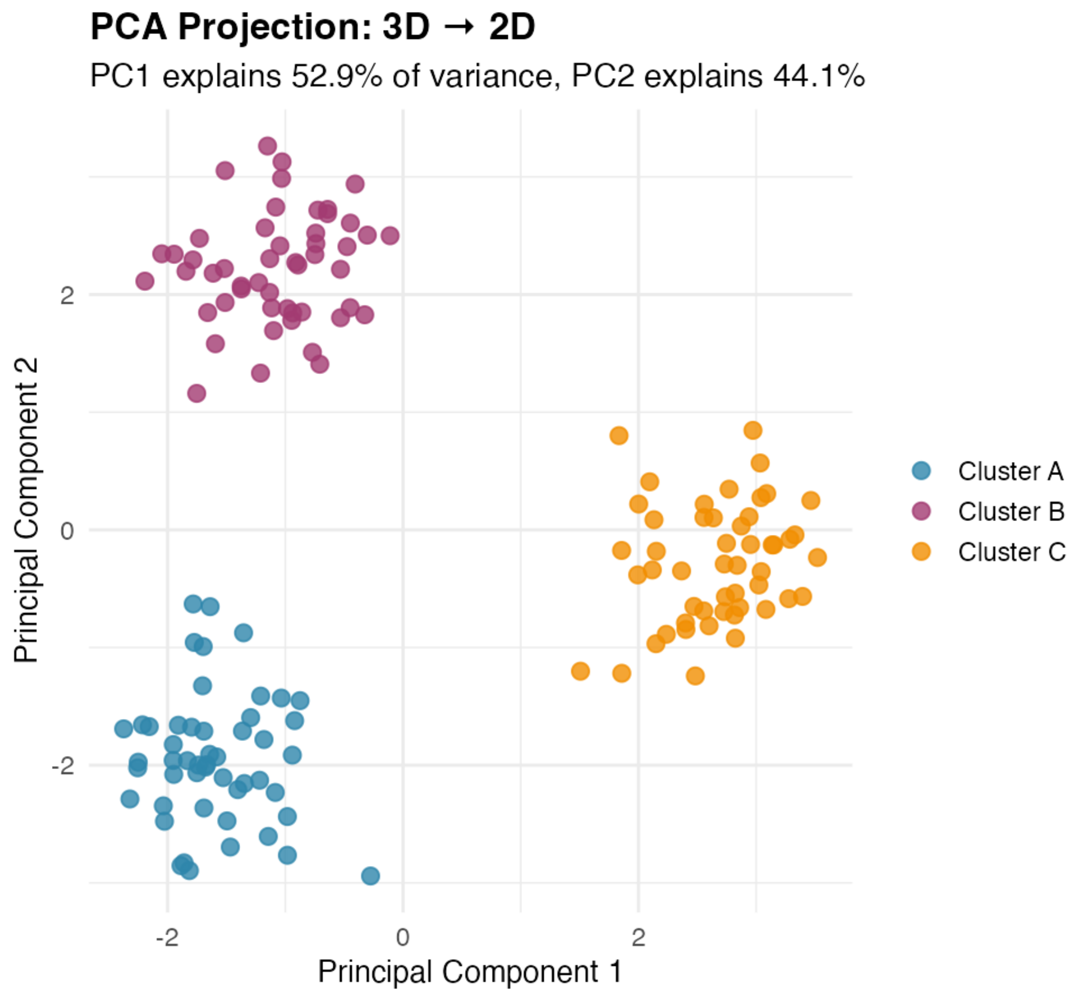




PCA Generally

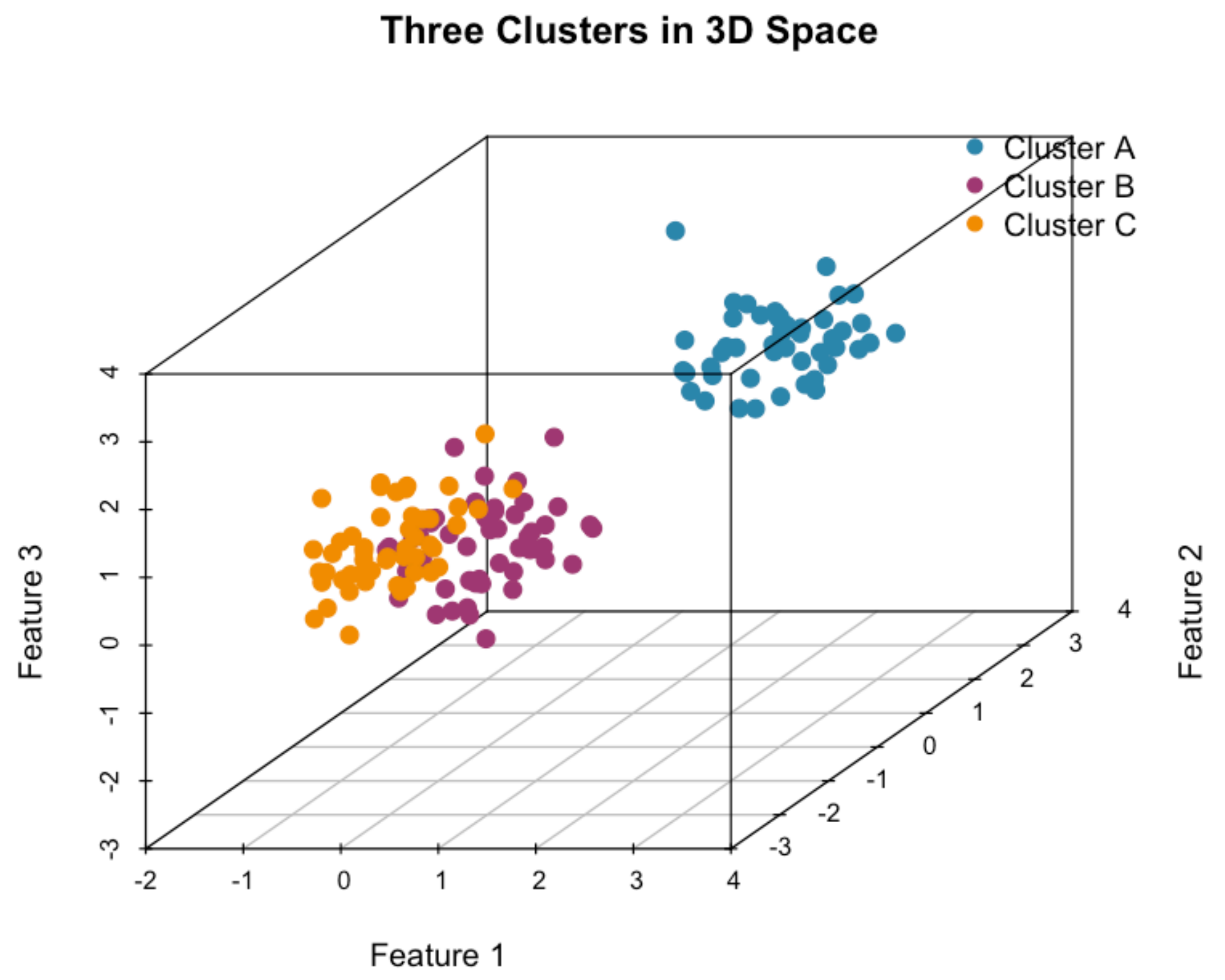
- Assume we have n datapoints of dimension d
  - I.e. a matrix X of size $d \times n$
- Goal: find the **most informative** $k < d$ **dimensions** (the "Principal Components")
  - These are the directions that **explain the most variance** in the data
  - Put another way: the **axes** along which the **data varies the most**

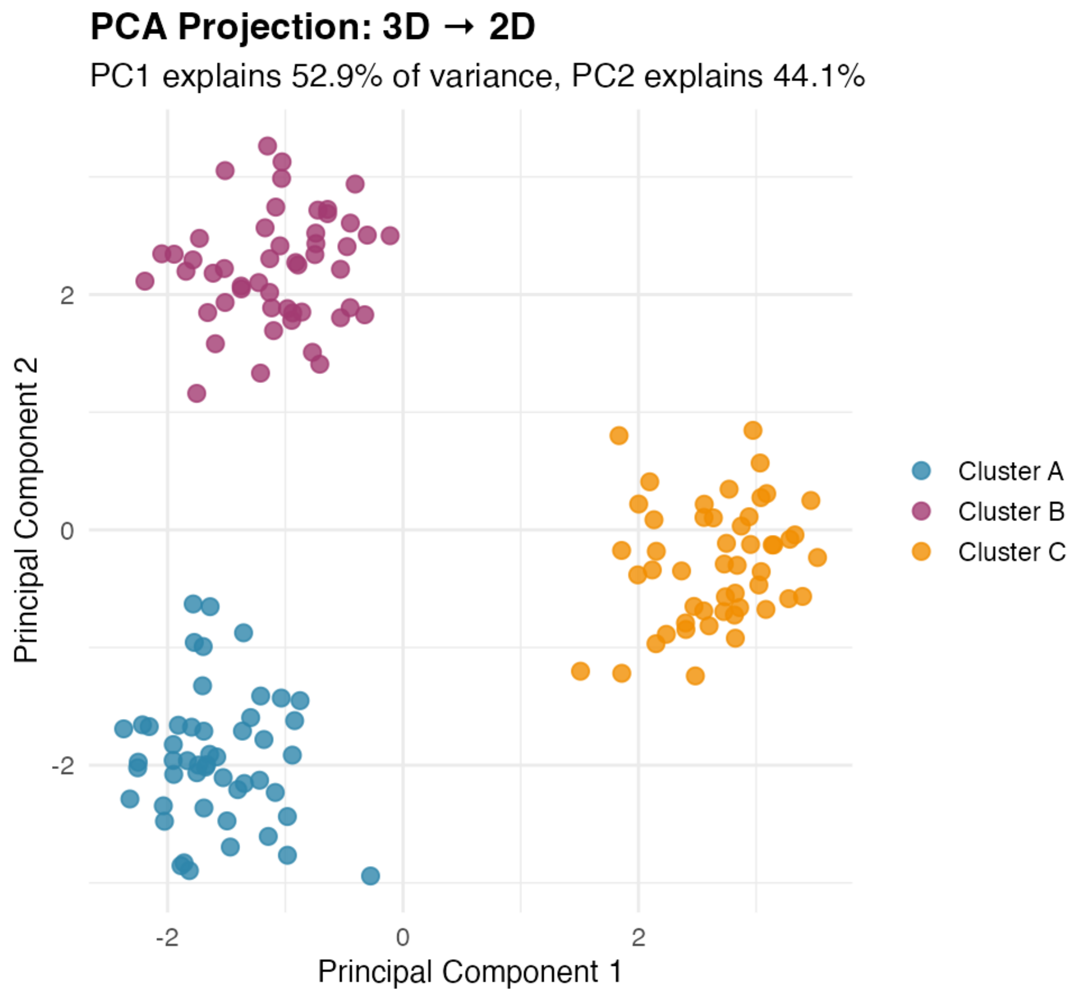




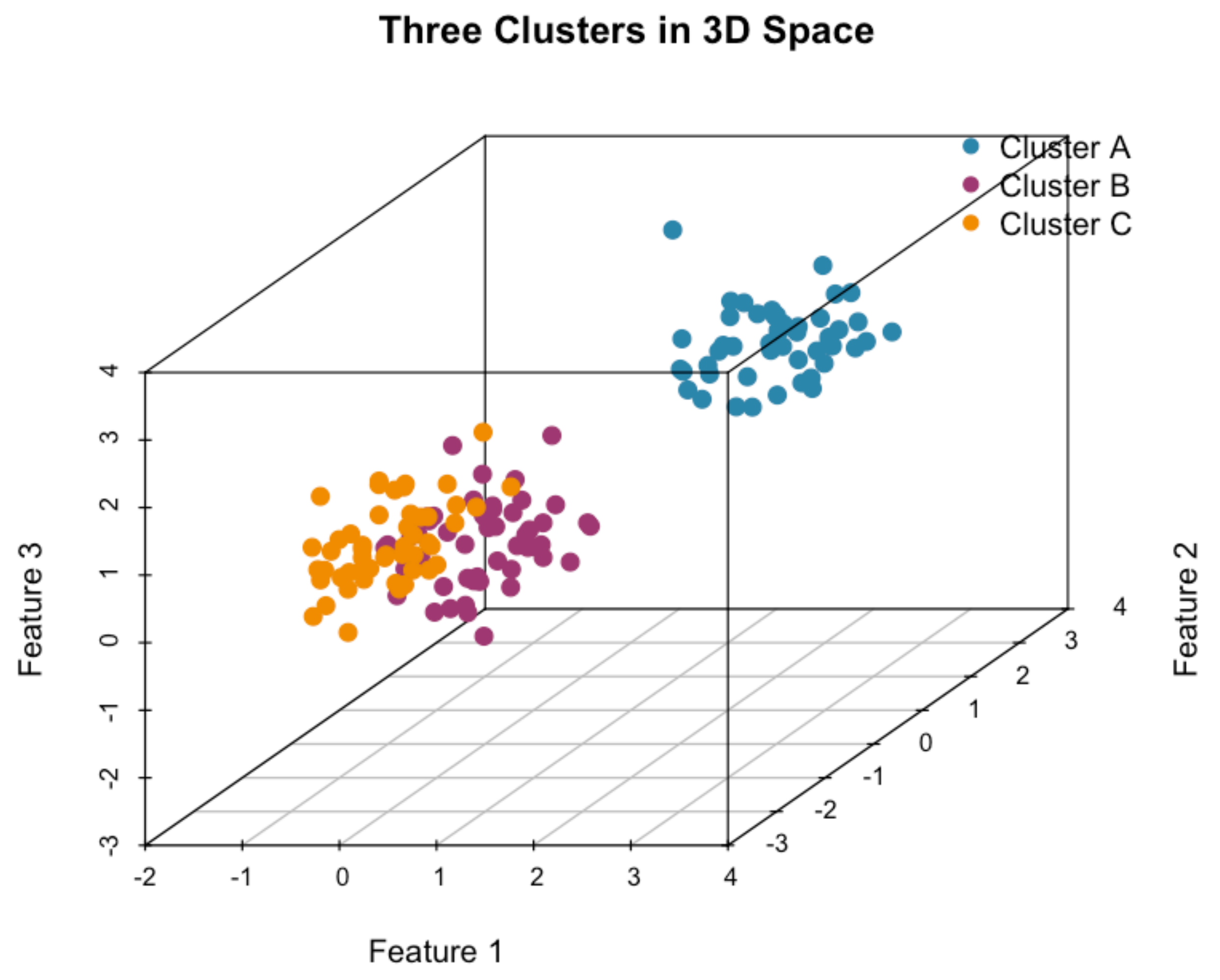
PCA Generally

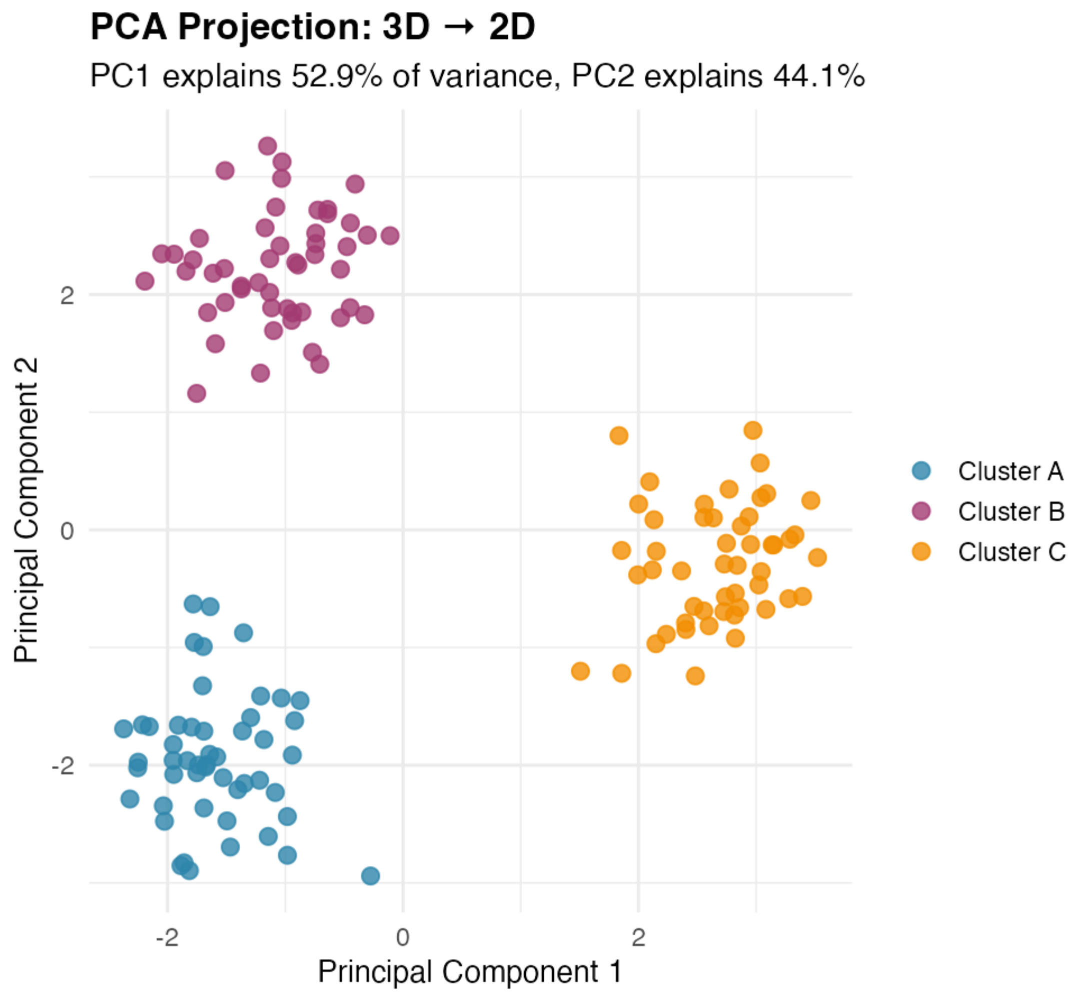
- Assume we have n datapoints of dimension d
  - I.e. a matrix X of size $d \times n$
- Goal: find the **most informative** $k < d$ **dimensions** (the "Principal Components")
  - These are the directions that **explain the most variance** in the data
  - Put another way: the **axes** along which the **data varies the most**
- Then **project** onto the **(hyper-)plane formed by those axes**
  - Projection ≈ **casting a shadow onto**





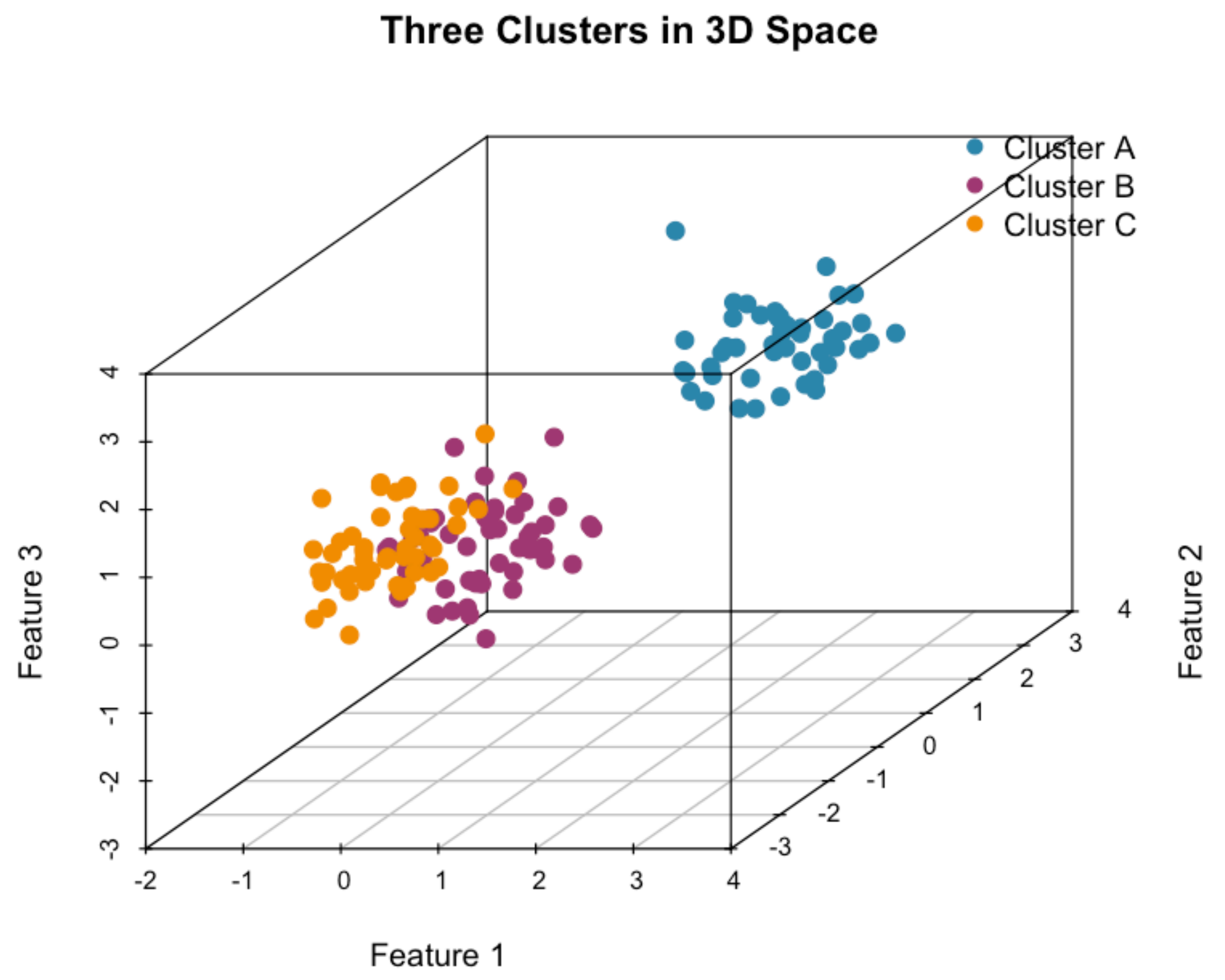
PCA and Manifolds

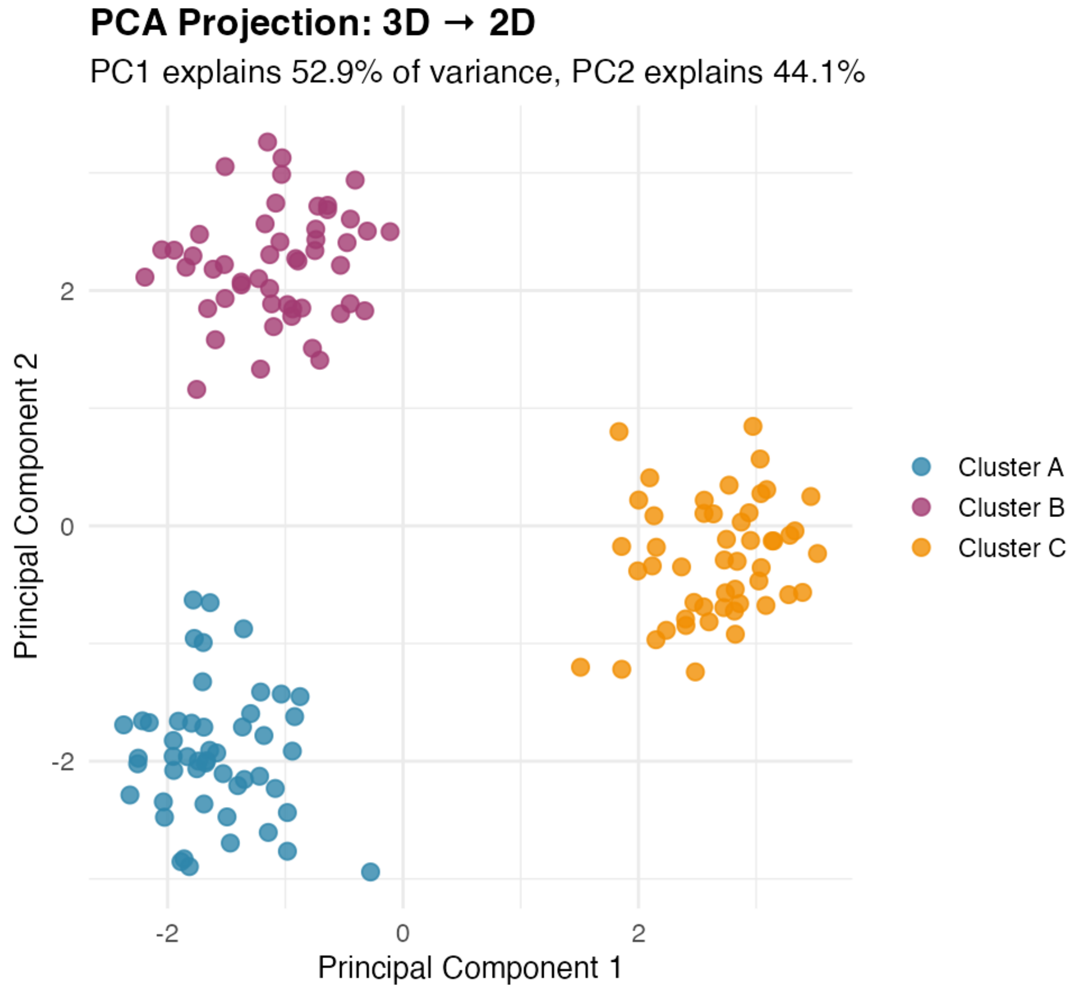




PCA and Manifolds

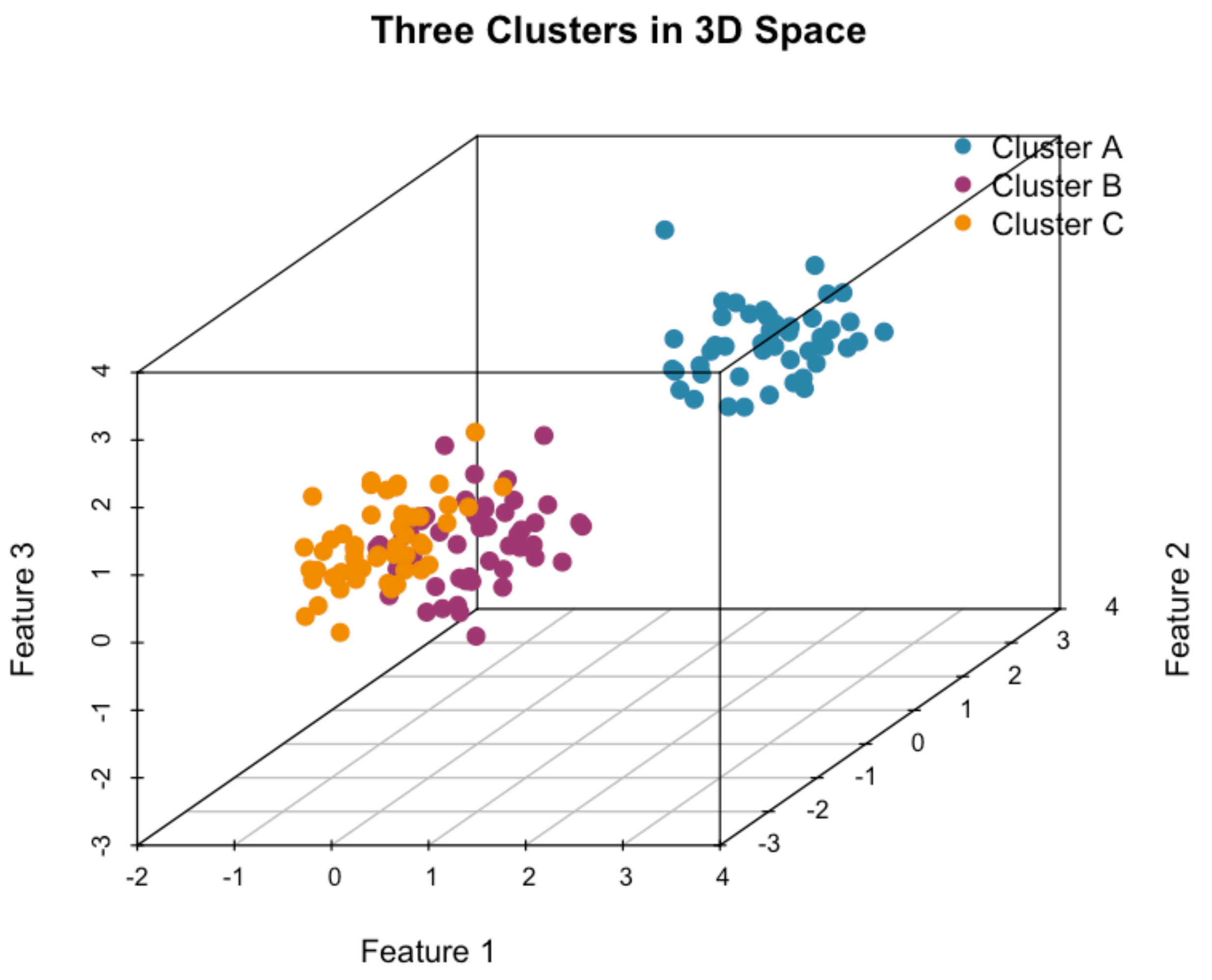
- Assumes the data is **near a linear manifold** (aka a [hyper-]**plane**)
  - Treats other dimensions as **noise**
  - Plane treated as the **"true nature"** of the data

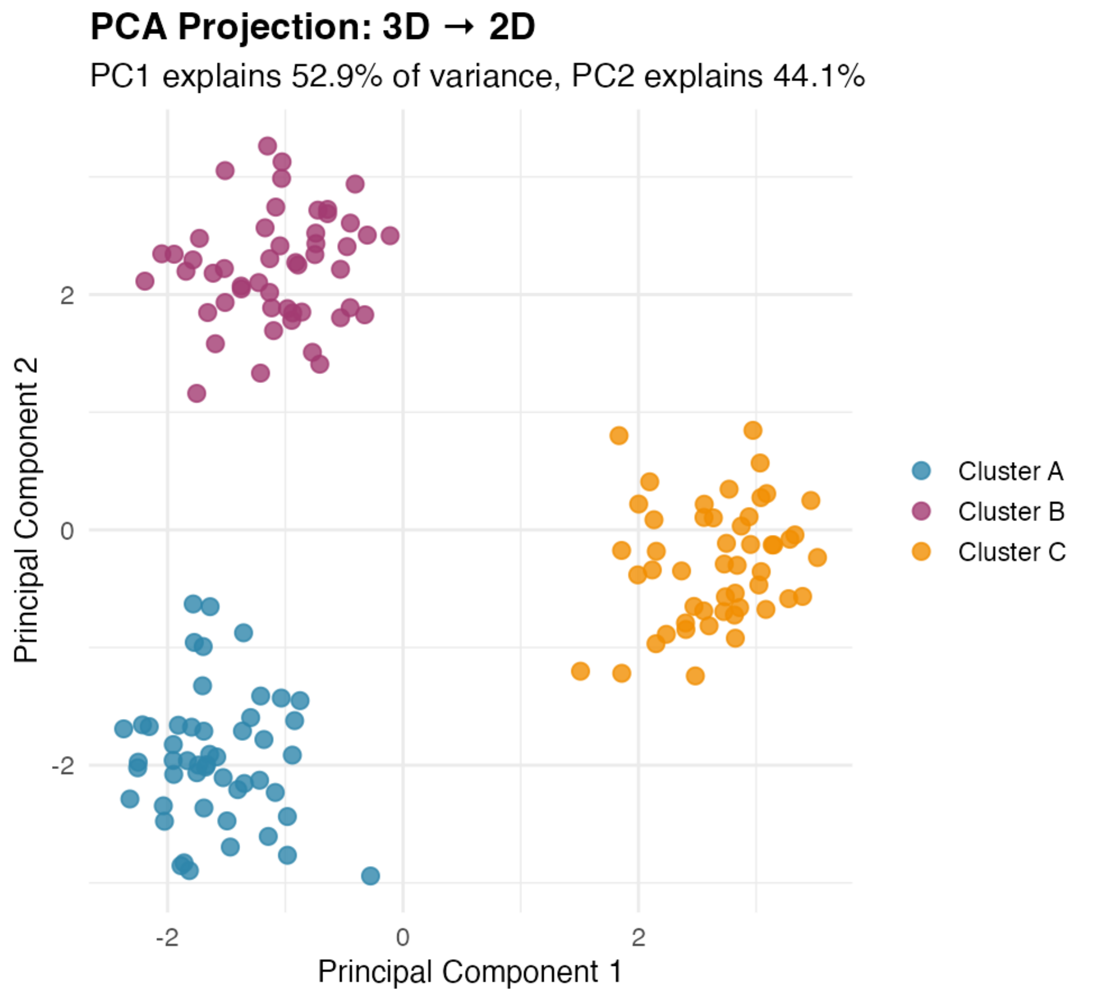




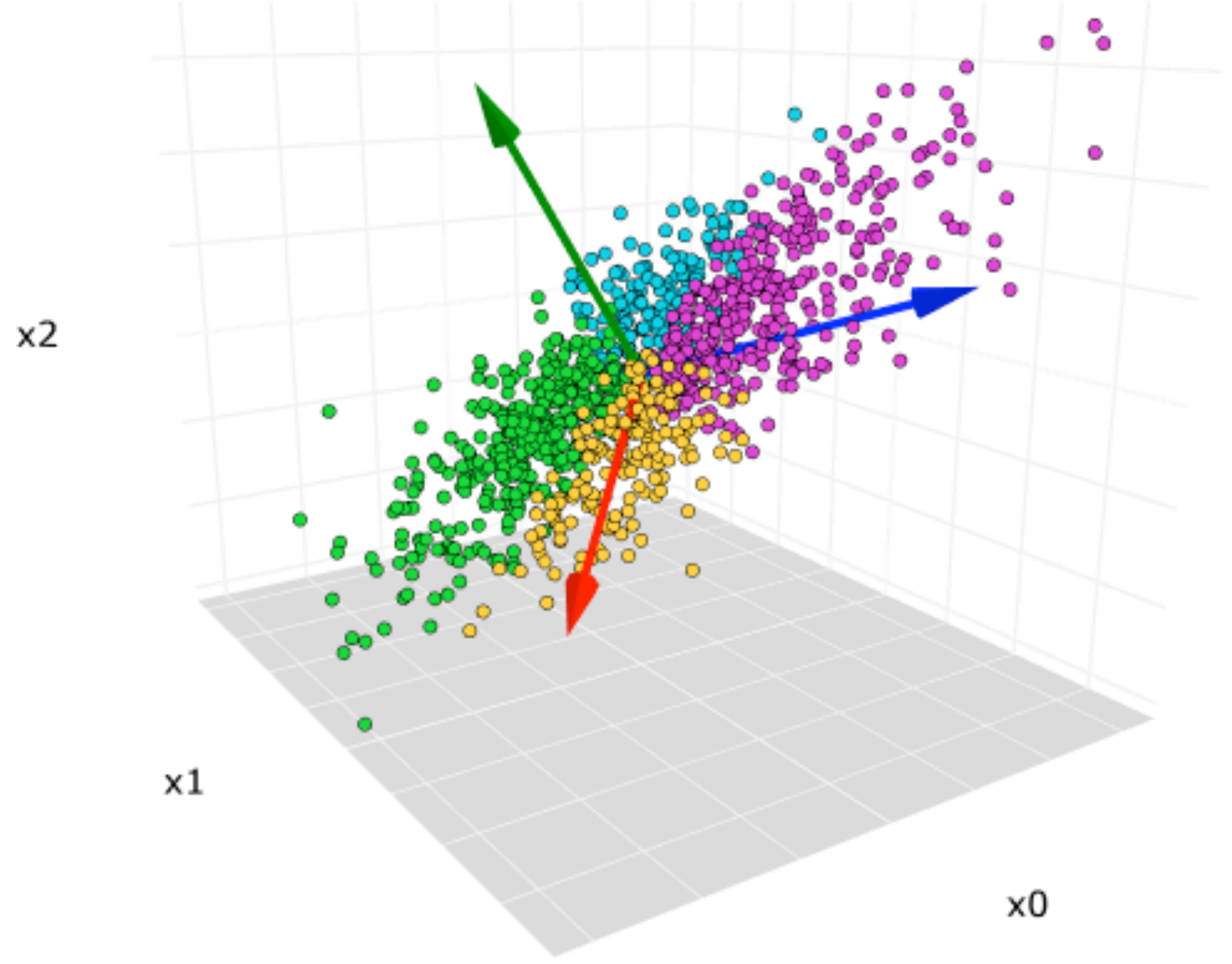
PCA and Manifolds

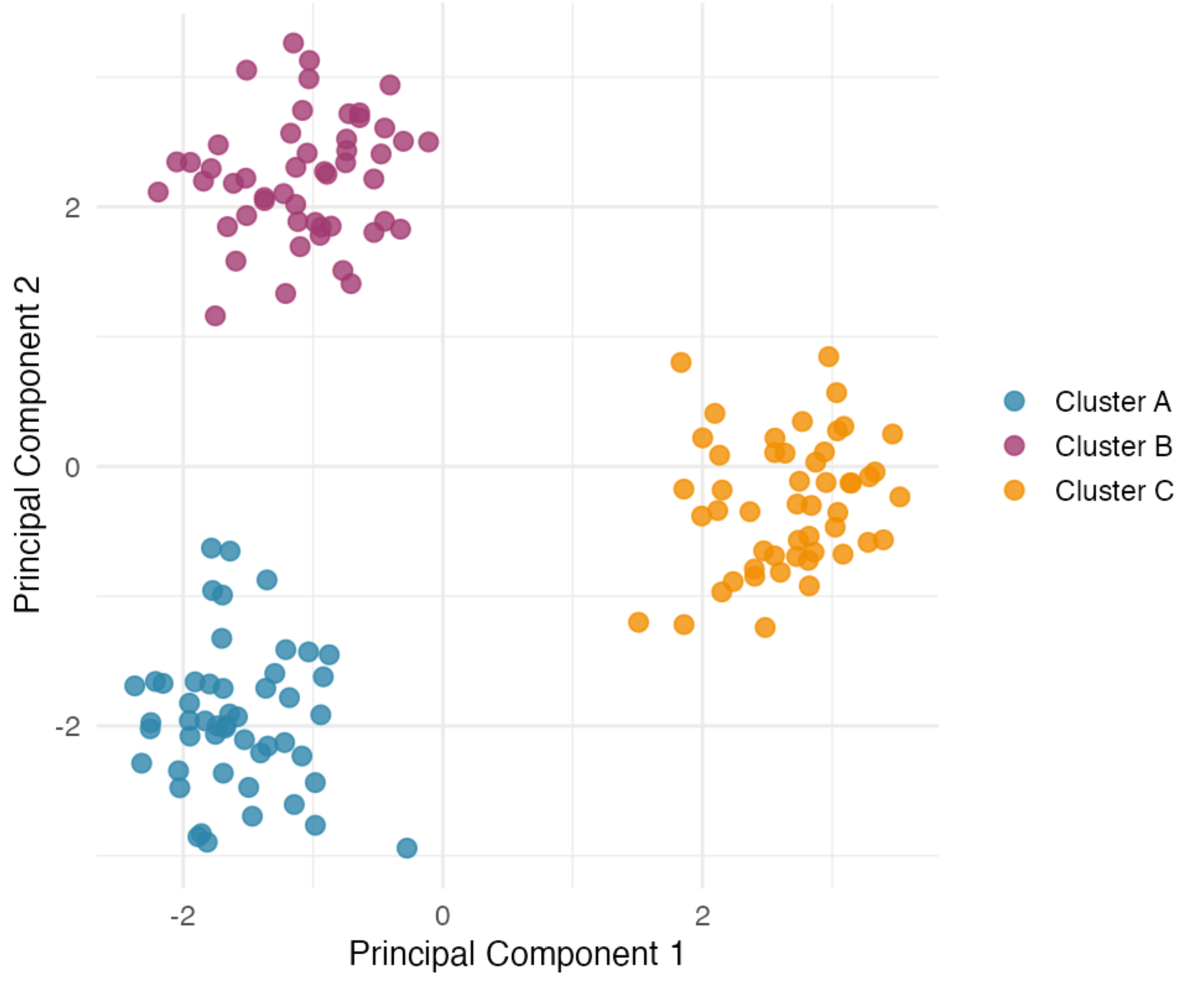
- Assumes the data is **near a linear manifold** (aka a [hyper-]**plane**)
  - Treats other dimensions as **noise**
  - Plane treated as the **"true nature"** of the data
- This is an **inductive bias!**
  - We'll see how this can **fail** (non-linear manifolds)





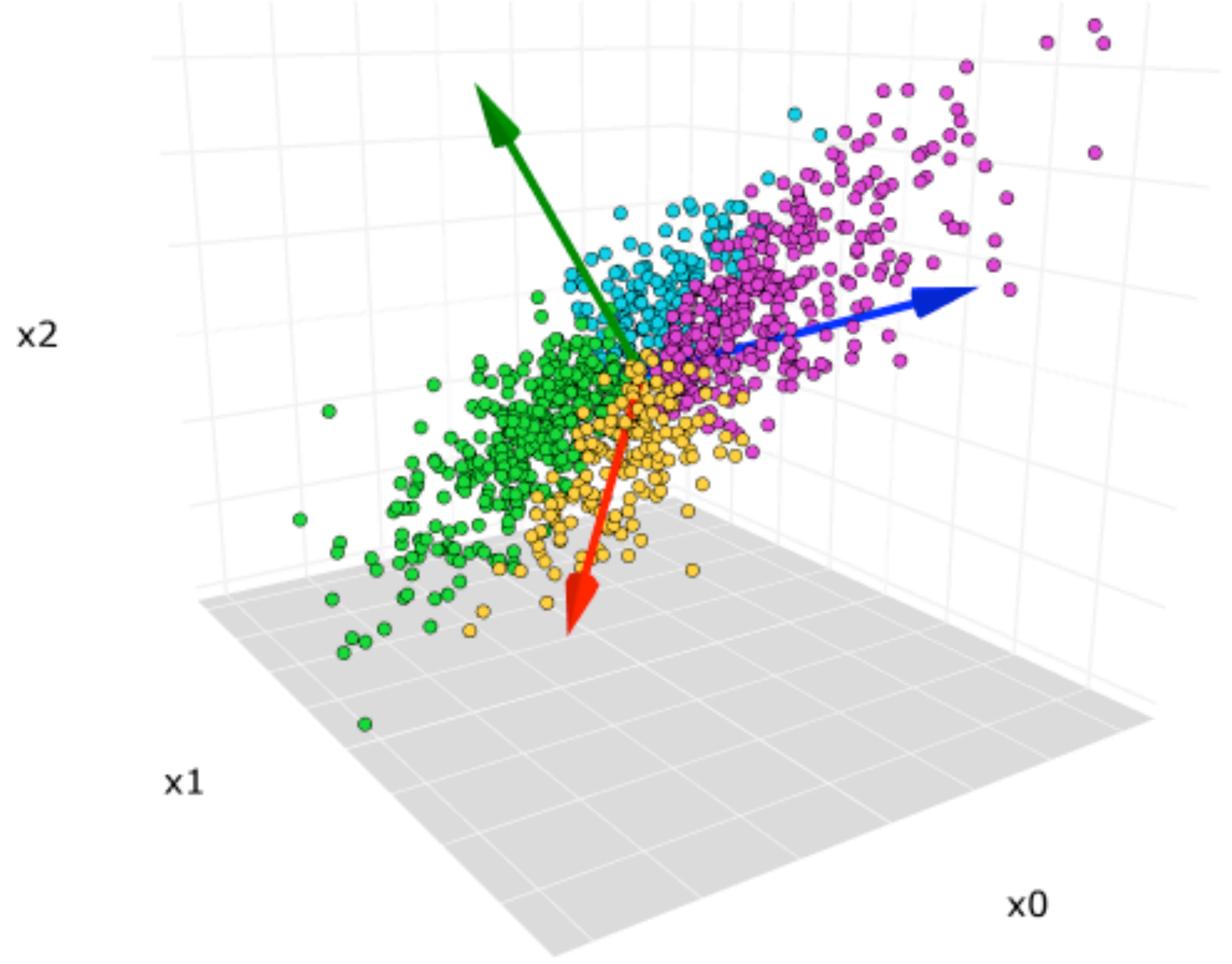
PCA Algorithm

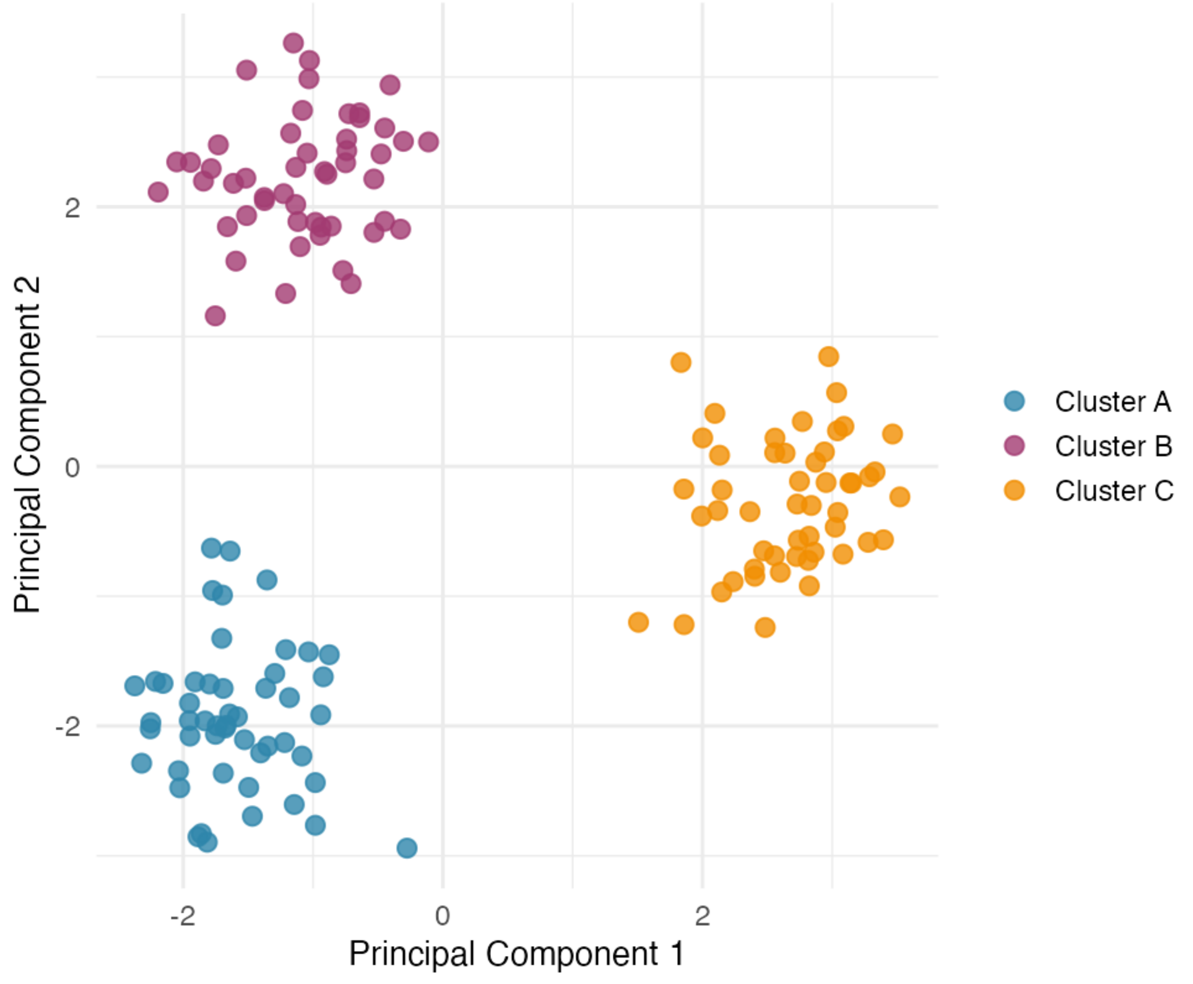




PCA Algorithm

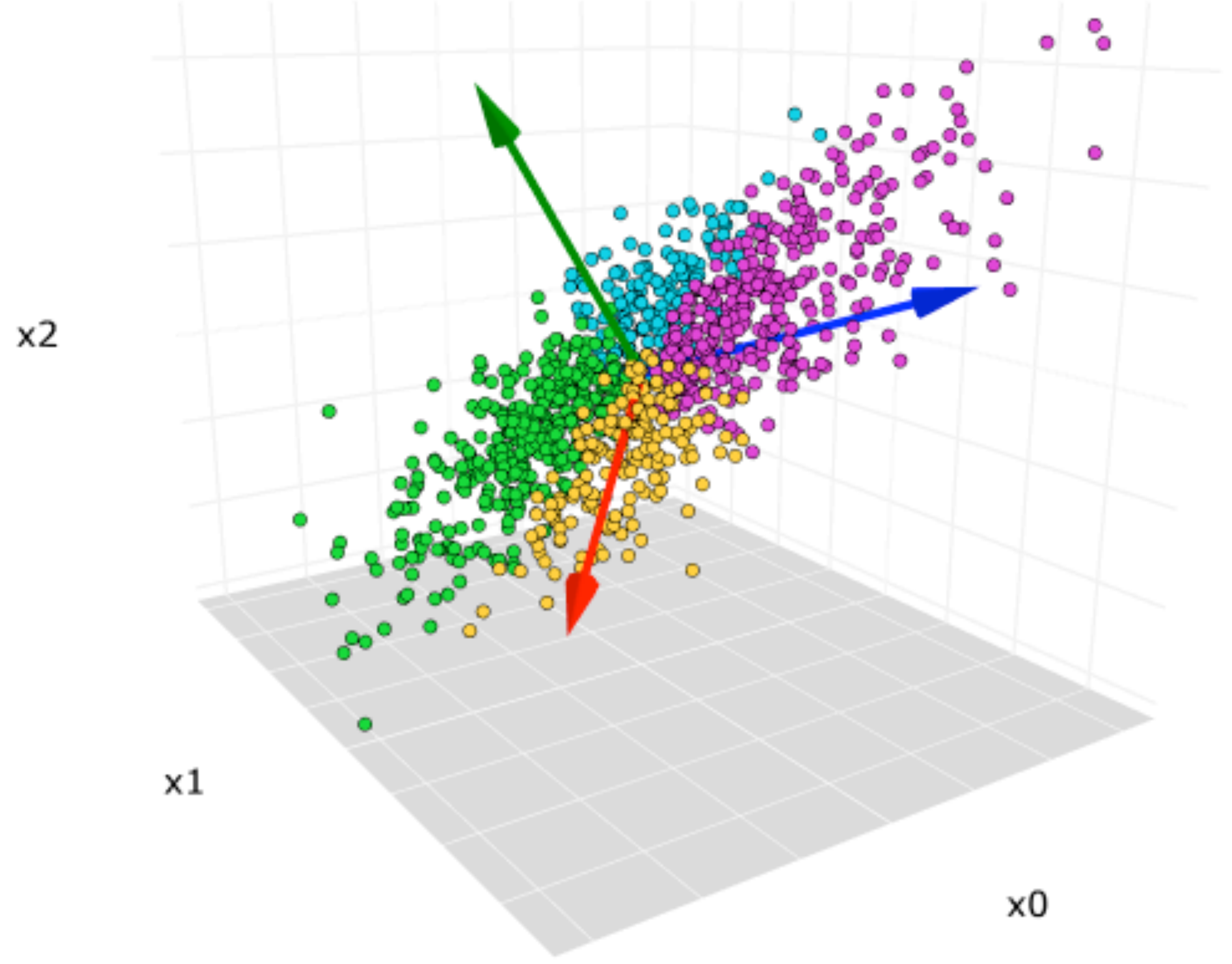
- Find the k **directions of greatest variance**
  - Gotten from the **covariance matrix** of the data (next slide)
  - These will be **orthogonal**

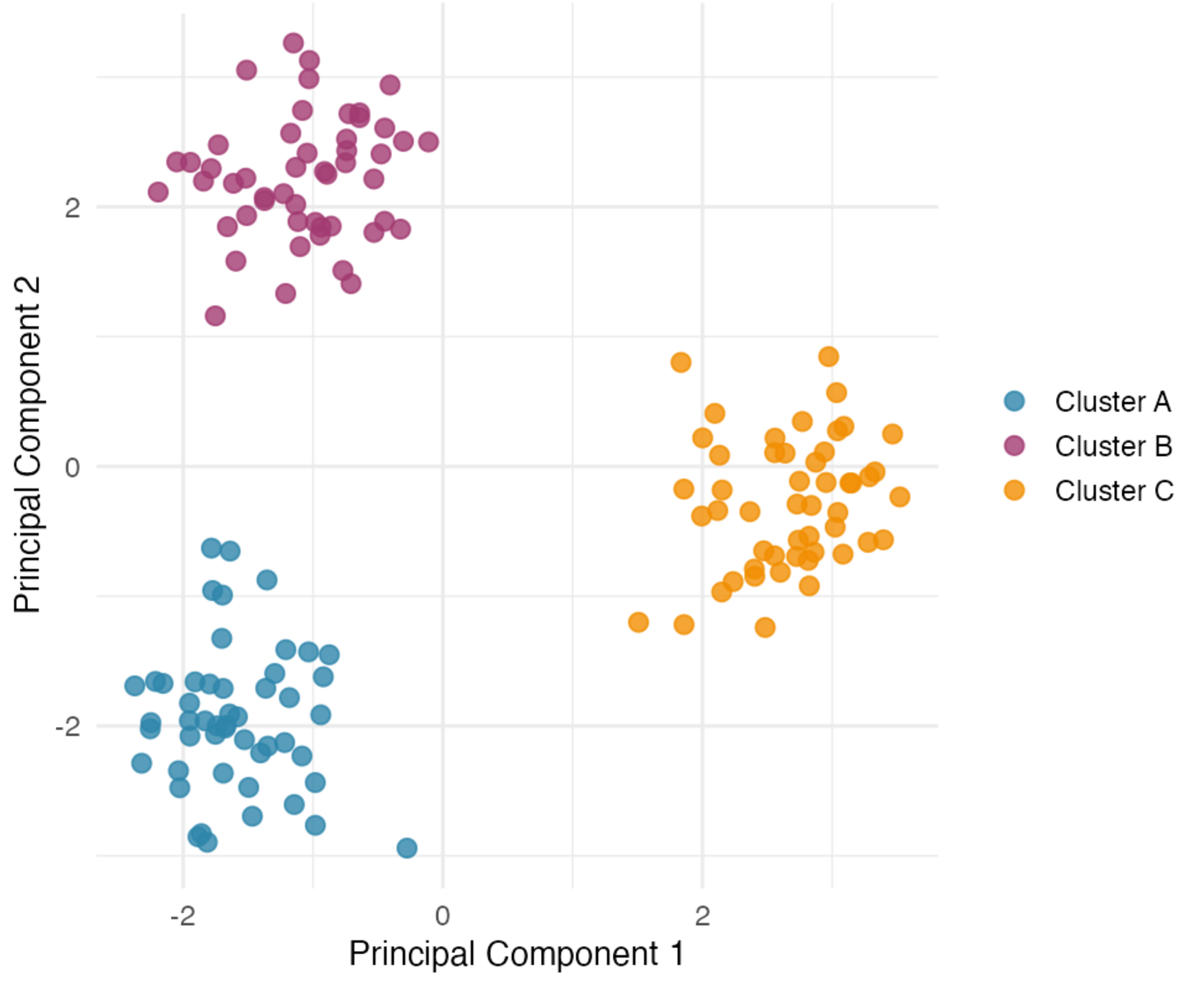




PCA Algorithm

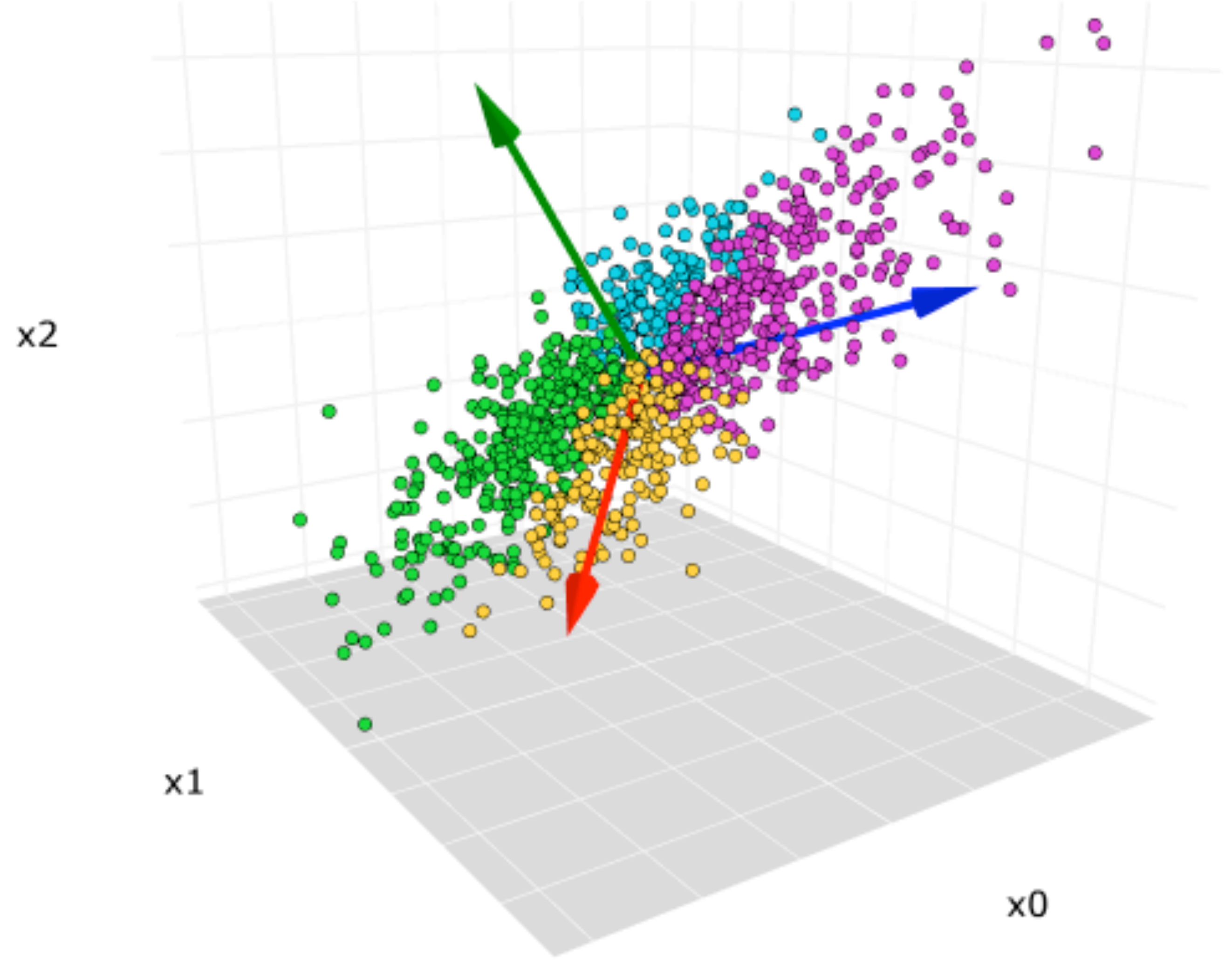
- Find the k **directions of greatest variance**
  - Gotten from the **covariance matrix** of the data (next slide)
  - These will be **orthogonal**
- The direction vectors **define a plane**

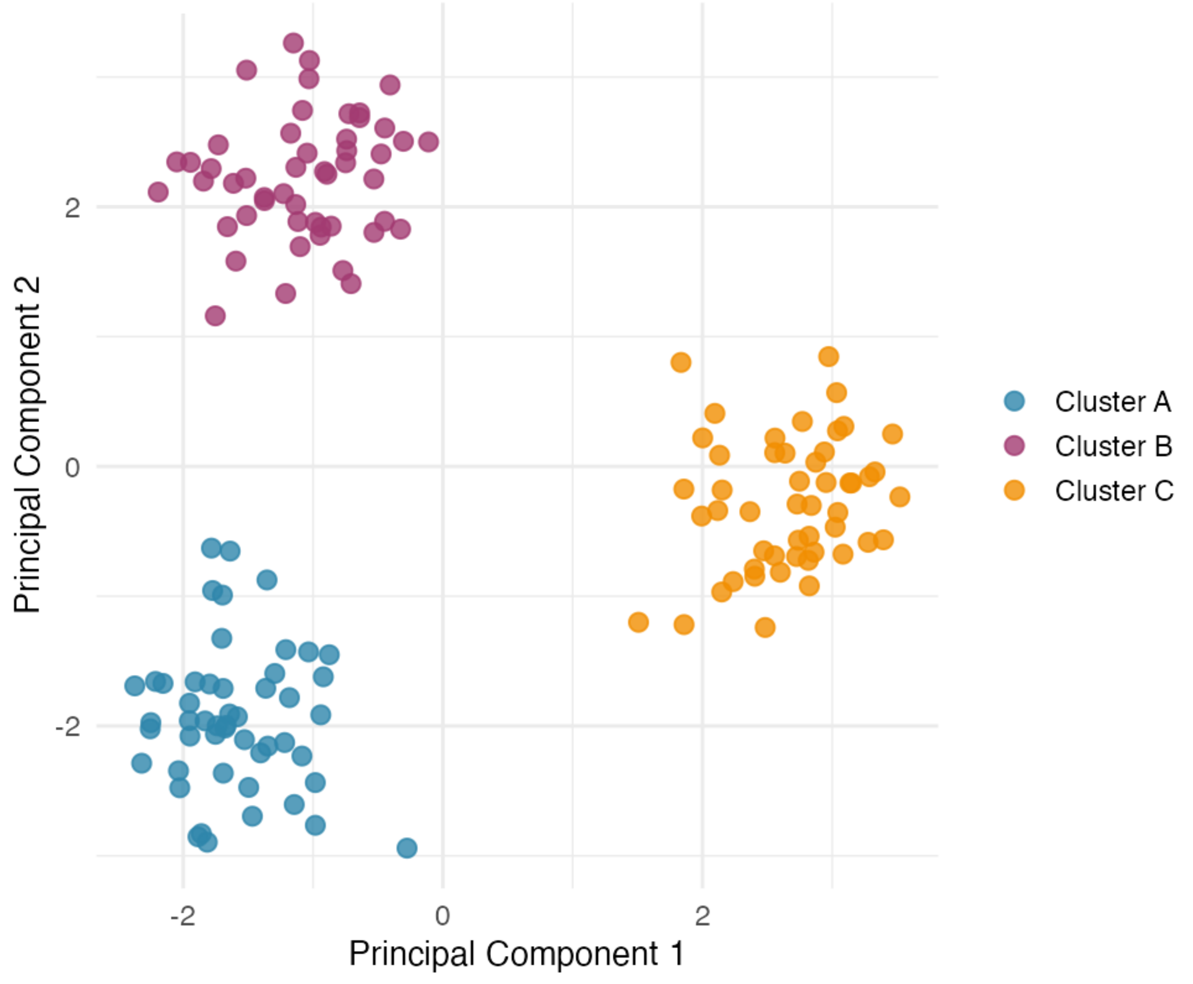




PCA Algorithm

- Find the k **directions of greatest variance**
  - Gotten from the **covariance matrix** of the data (next slide)
  - These will be **orthogonal**
- The direction vectors **define a plane**
- **Project** the data **onto that plane**
  - This is fairly simple Linear Algebra





Covariance Matrix

$$\sum_k \mathbf{x}_k \mathbf{x}_k^T = \sum_k \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} [x_1 \quad x_2 \quad \cdots \quad x_d] = \sum_k \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_d \\ x_2x_1 & x_2x_2 & \cdots & x_2x_d \\ \vdots & \vdots & \ddots & \vdots \\ x_dx_1 & x_dx_2 & \cdots & x_dx_d \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} --- & \mathbf{x}_1^T & --- \\ --- & \mathbf{x}_2^T & --- \\ & \vdots & \\ --- & \mathbf{x}_n^T & --- \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

Covariance Matrix

$$\sum_k \mathbf{x}_k \mathbf{x}_k^T = \sum_k \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_d \end{bmatrix} = \sum_k \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_d \\ x_2x_1 & x_2x_2 & \cdots & x_2x_d \\ \vdots & \vdots & \ddots & \vdots \\ x_dx_1 & x_dx_2 & \cdots & x_dx_d \end{bmatrix}$$

one high-dimensional datapoint

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \text{---} & \mathbf{x}_1^T & \text{---} \\ \text{---} & \mathbf{x}_2^T & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{x}_n^T & \text{---} \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

Covariance Matrix

$$\sum_k \mathbf{x}_k \mathbf{x}_k^T = \sum_k \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} [x_1 \quad x_2 \quad \cdots \quad x_d] = \sum_k \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_d \\ x_2x_1 & x_2x_2 & \cdots & x_2x_d \\ \vdots & \vdots & \ddots & \vdots \\ x_dx_1 & x_dx_2 & \cdots & x_dx_d \end{bmatrix}$$

one high-dimensional datapoint

These three definitions are **equivalent!**

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \text{---} & \mathbf{x}_1^T & \text{---} \\ \text{---} & \mathbf{x}_2^T & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{x}_n^T & \text{---} \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

Covariance Matrix

$$\sum_k \mathbf{x}_k \mathbf{x}_k^T = \sum_k \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} [x_1 \quad x_2 \quad \cdots \quad x_d] = \sum_k \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_d \\ x_2x_1 & x_2x_2 & \cdots & x_2x_d \\ \vdots & \vdots & \ddots & \vdots \\ x_dx_1 & x_dx_2 & \cdots & x_dx_d \end{bmatrix}$$

one high-dimensional datapoint

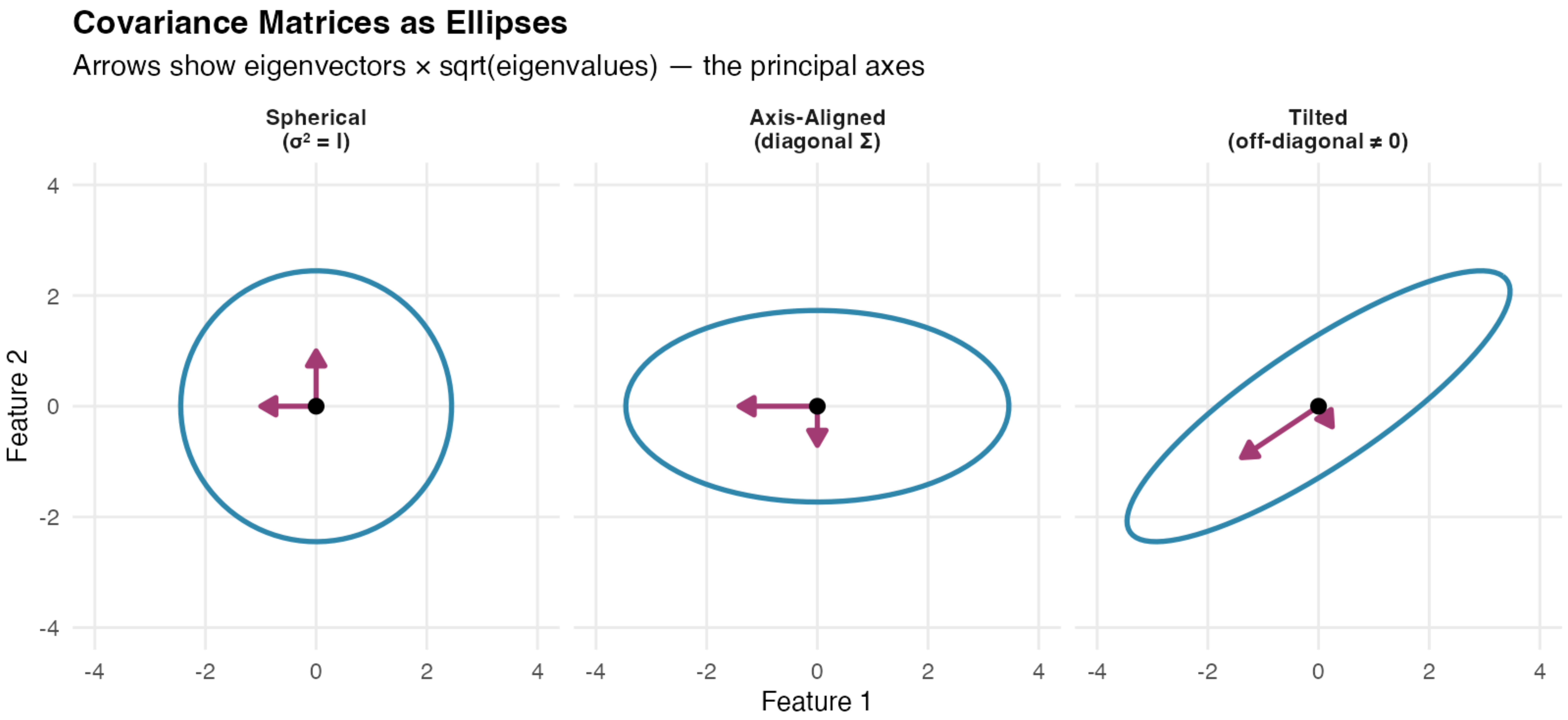
These three definitions are **equivalent!**

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \text{---} & \mathbf{x}_1^T & \text{---} \\ \text{---} & \mathbf{x}_2^T & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{x}_n^T & \text{---} \end{bmatrix}$$

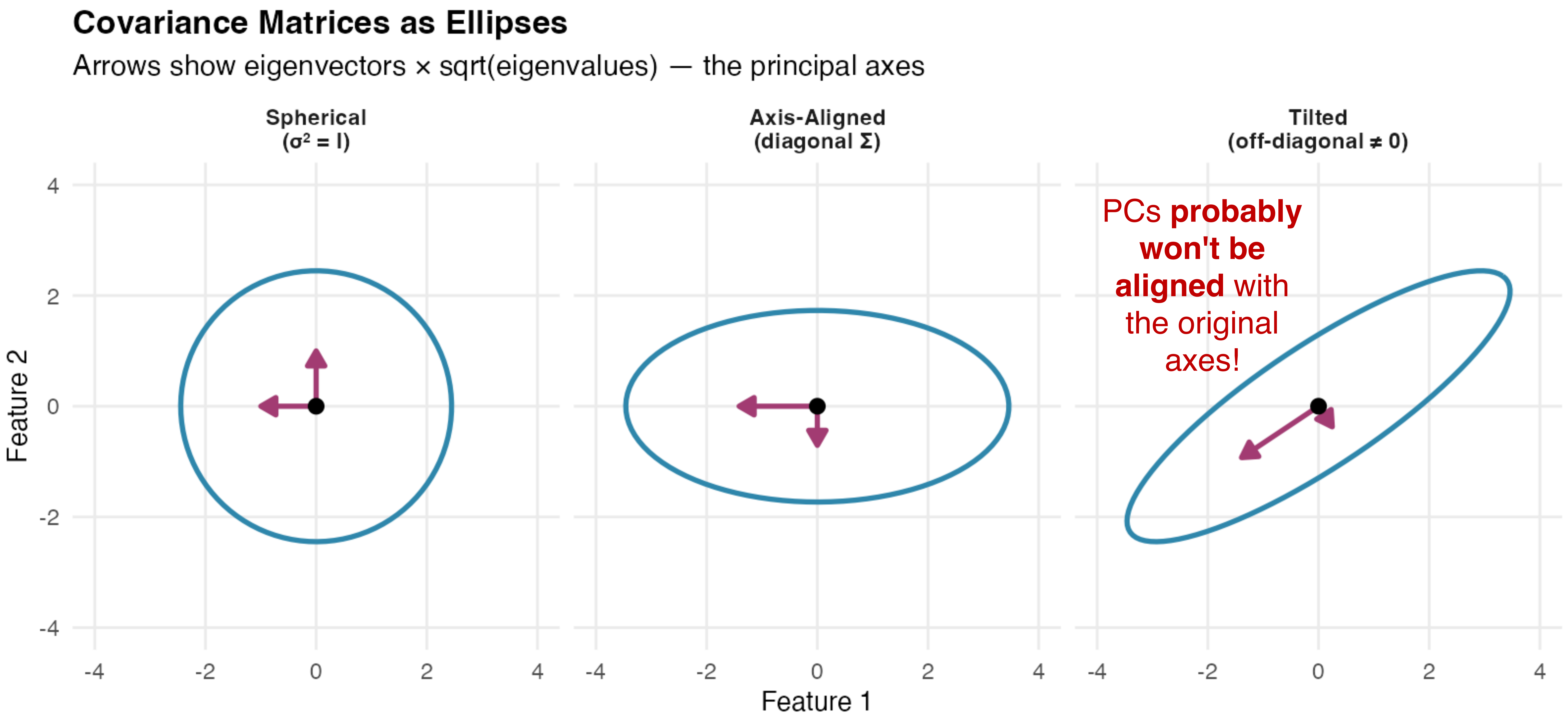
How much **co-variation** does **each pair of variables** have?

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

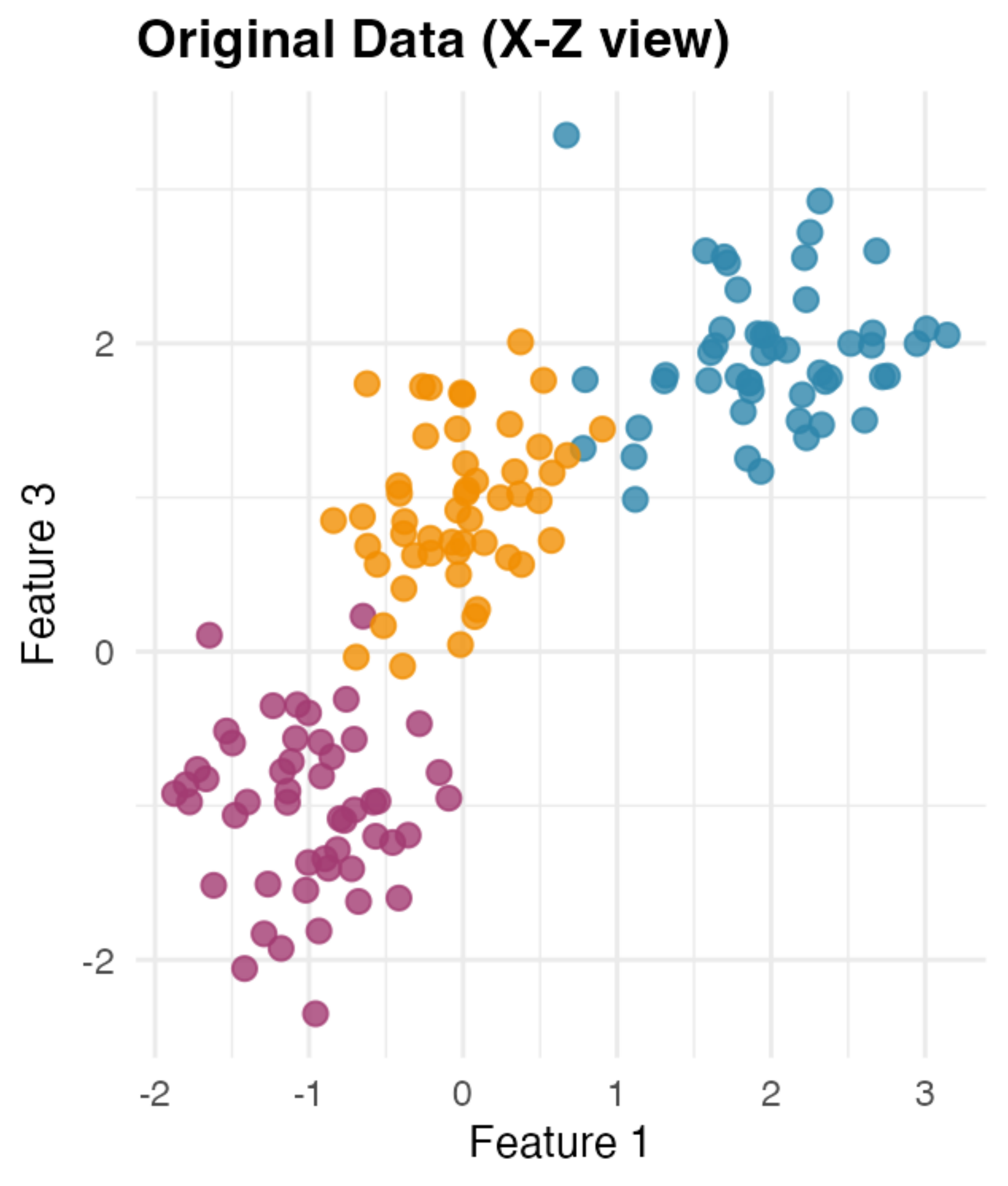
Covariance matrix defines an ellipse

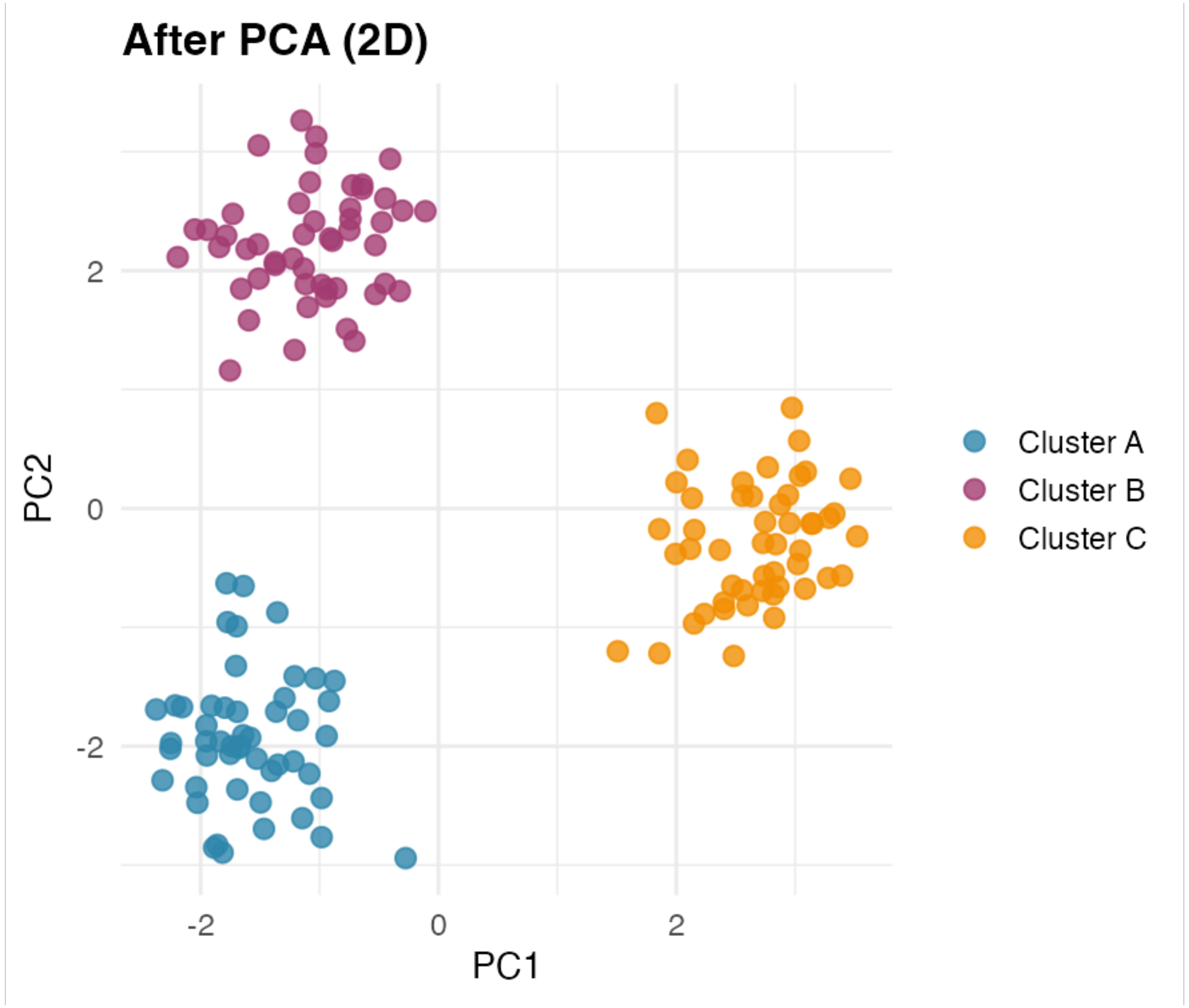


Covariance matrix defines an ellipse

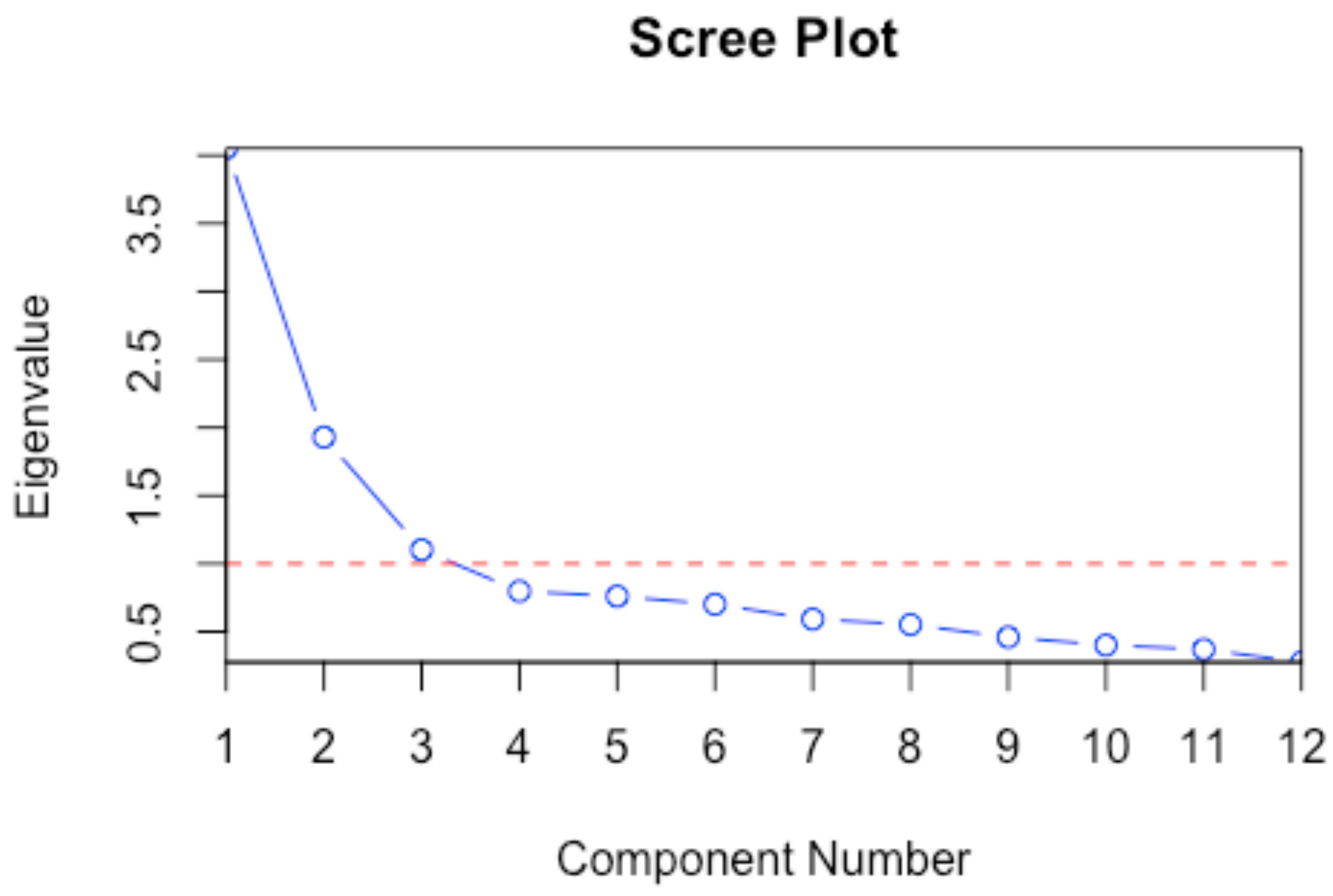


PCA aligns to PC axes



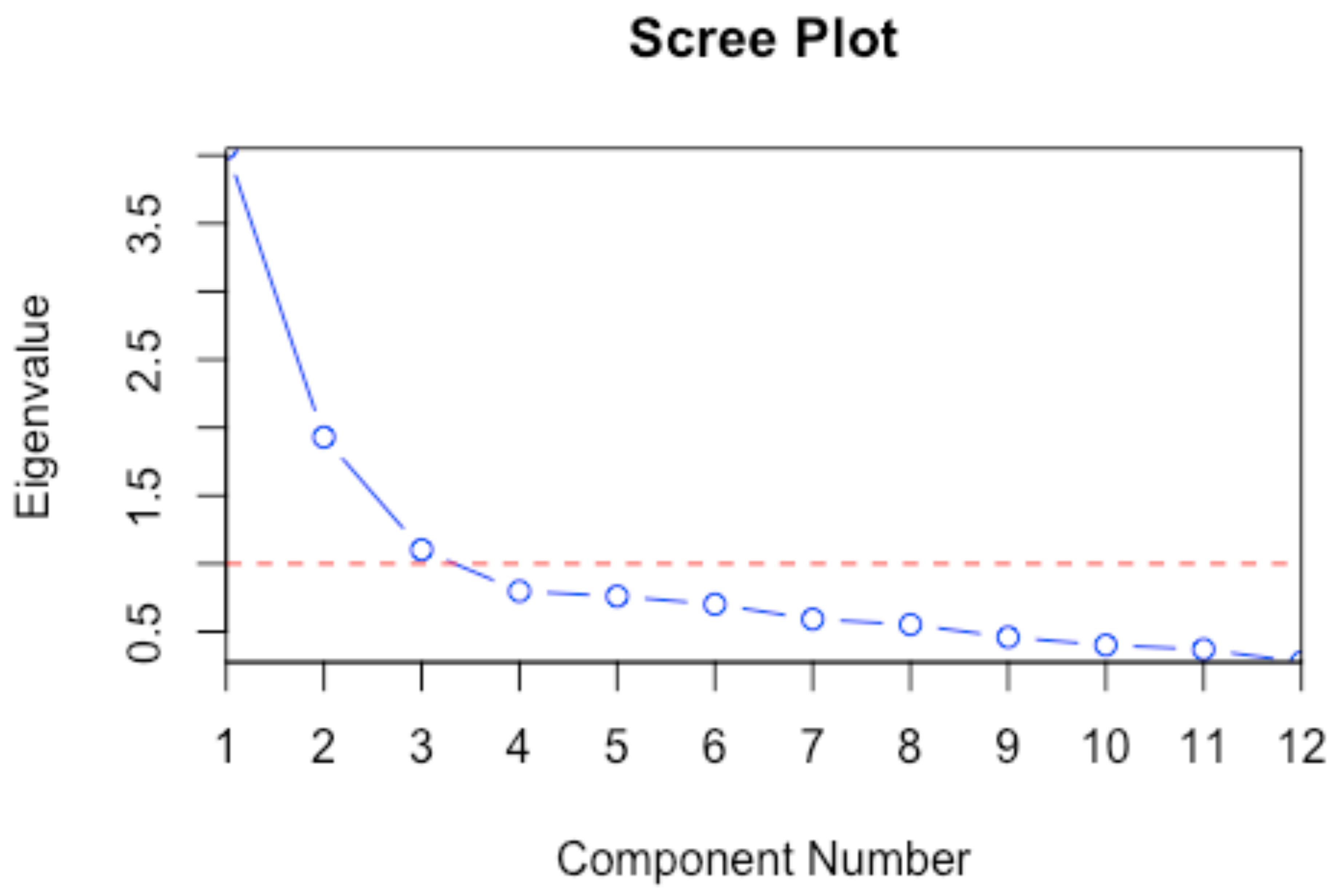


Choosing Component Numbers



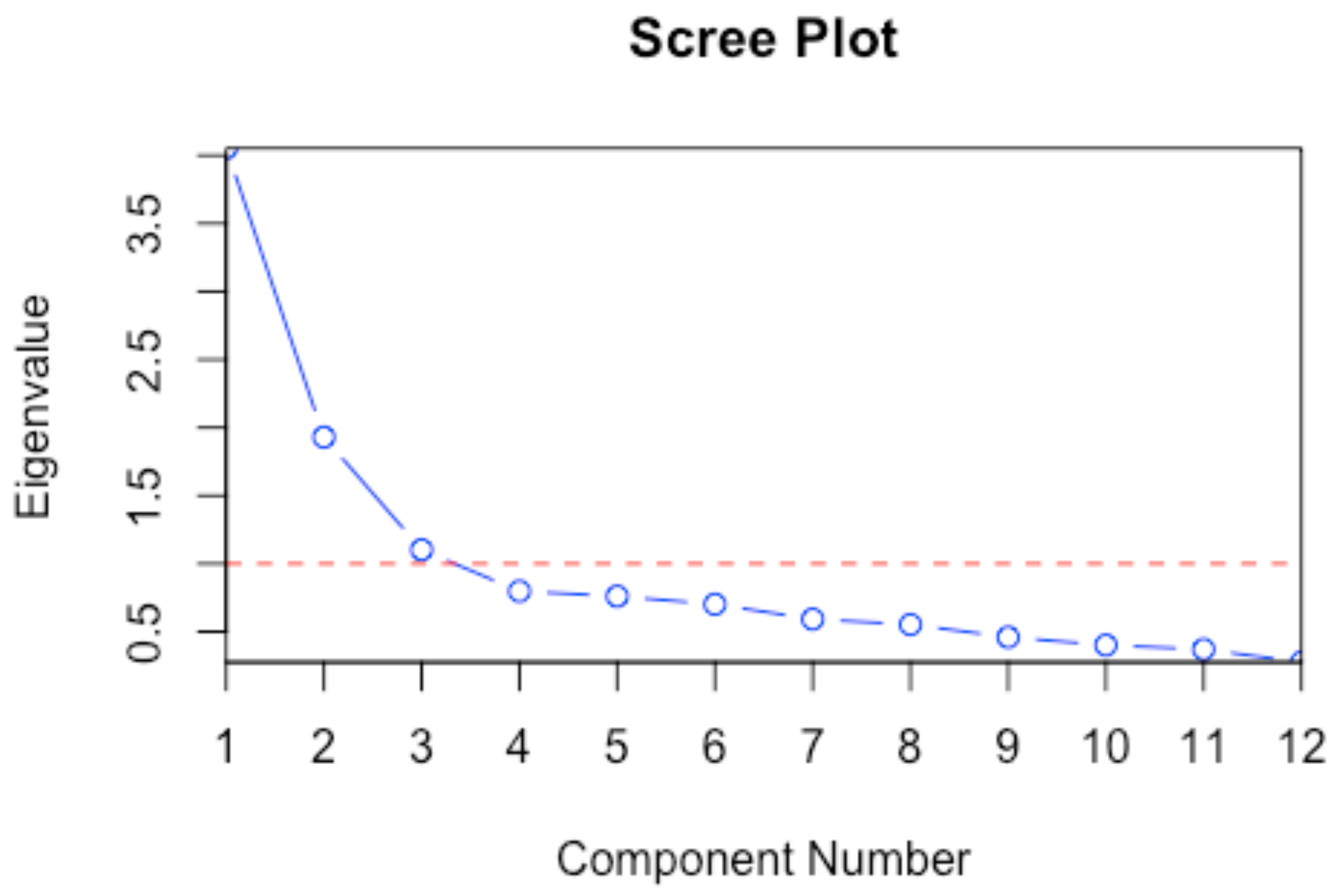
Choosing Component Numbers

- Number of components k to keep usually chosen with a **Scree Plot**
  - Plot of **how much variance** each component explains
  - These are also called the **Eigenvalues**

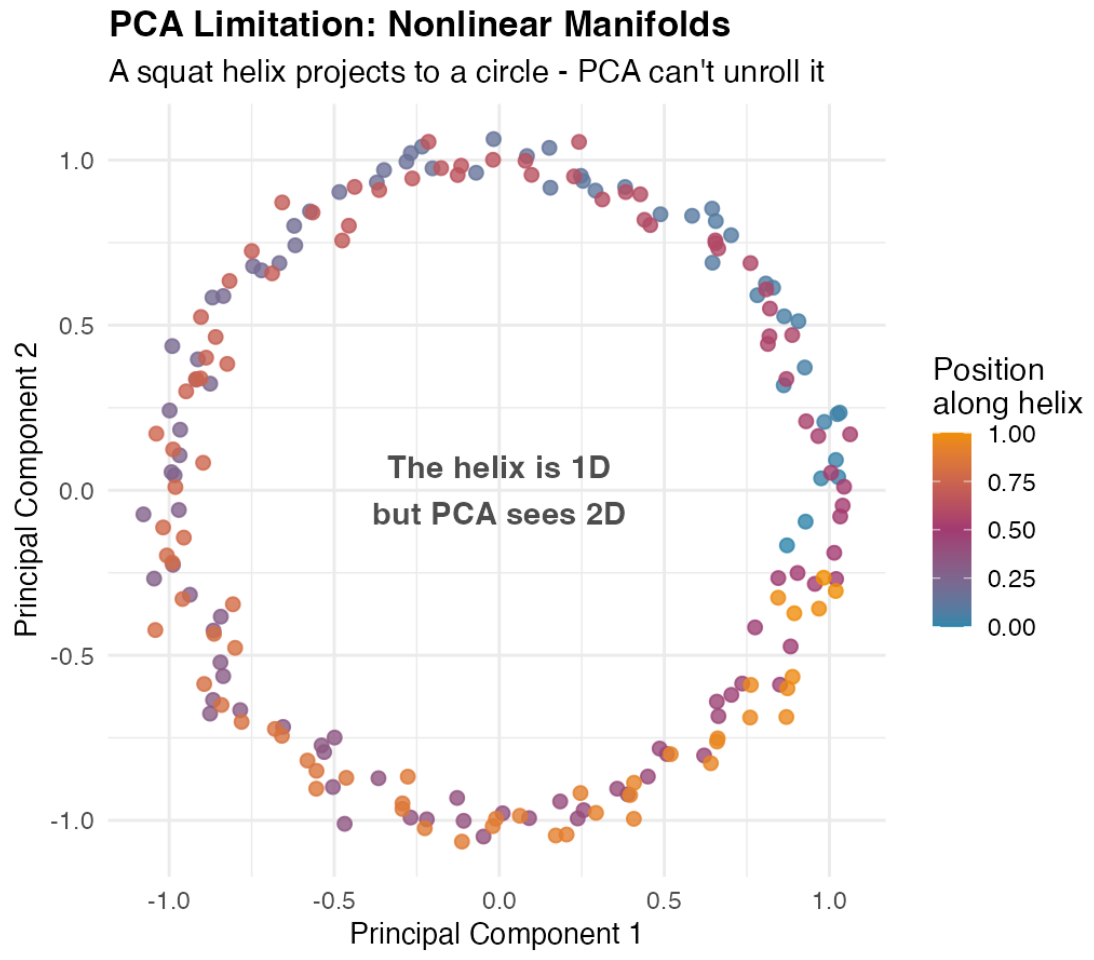


Choosing Component Numbers

- Number of components k to keep usually chosen with a **Scree Plot**
  - Plot of **how much variance** each component explains
  - These are also called the **Eigenvalues**
- Often the **point of inflection** is chosen
  - (i.e. diminishing returns)

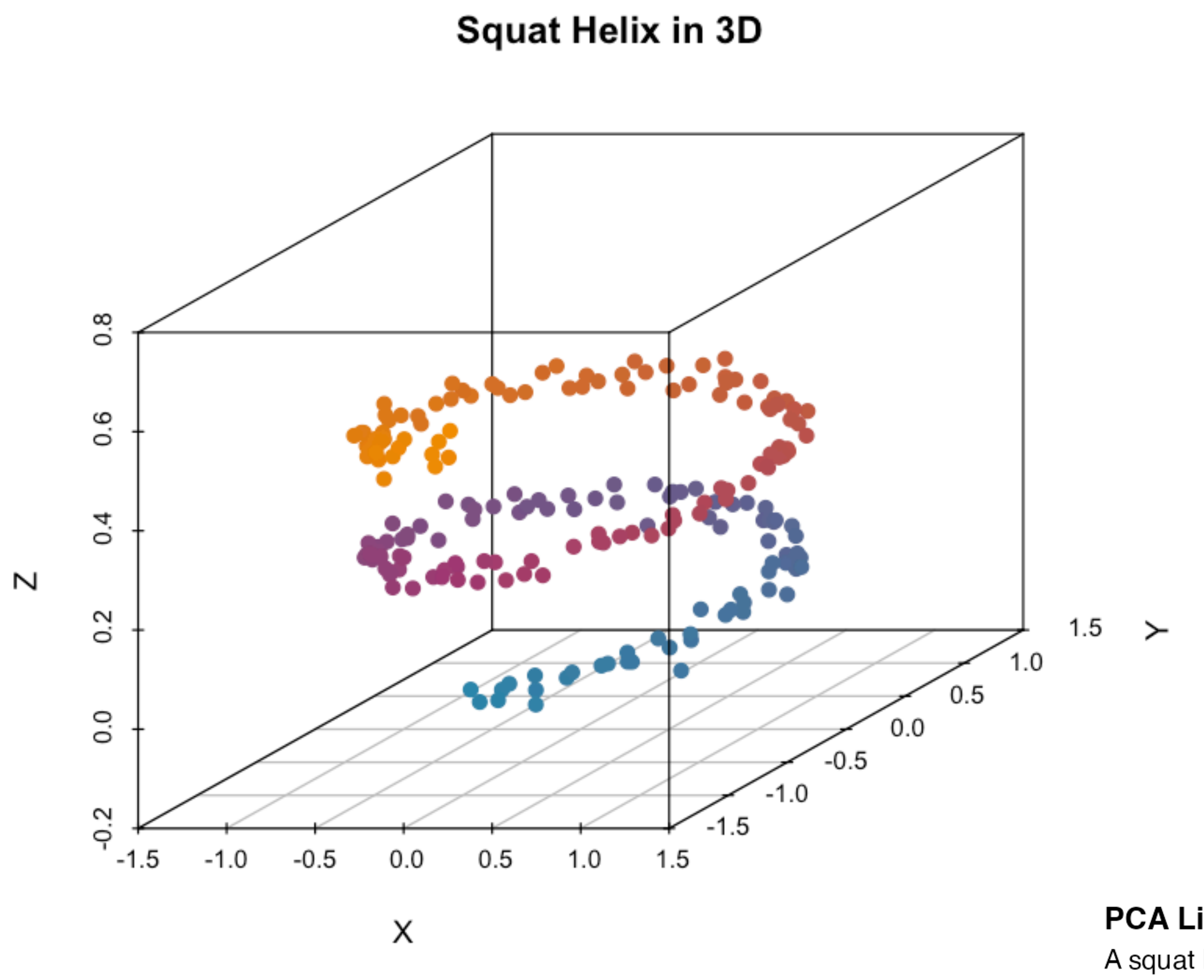


Where PCA Fails

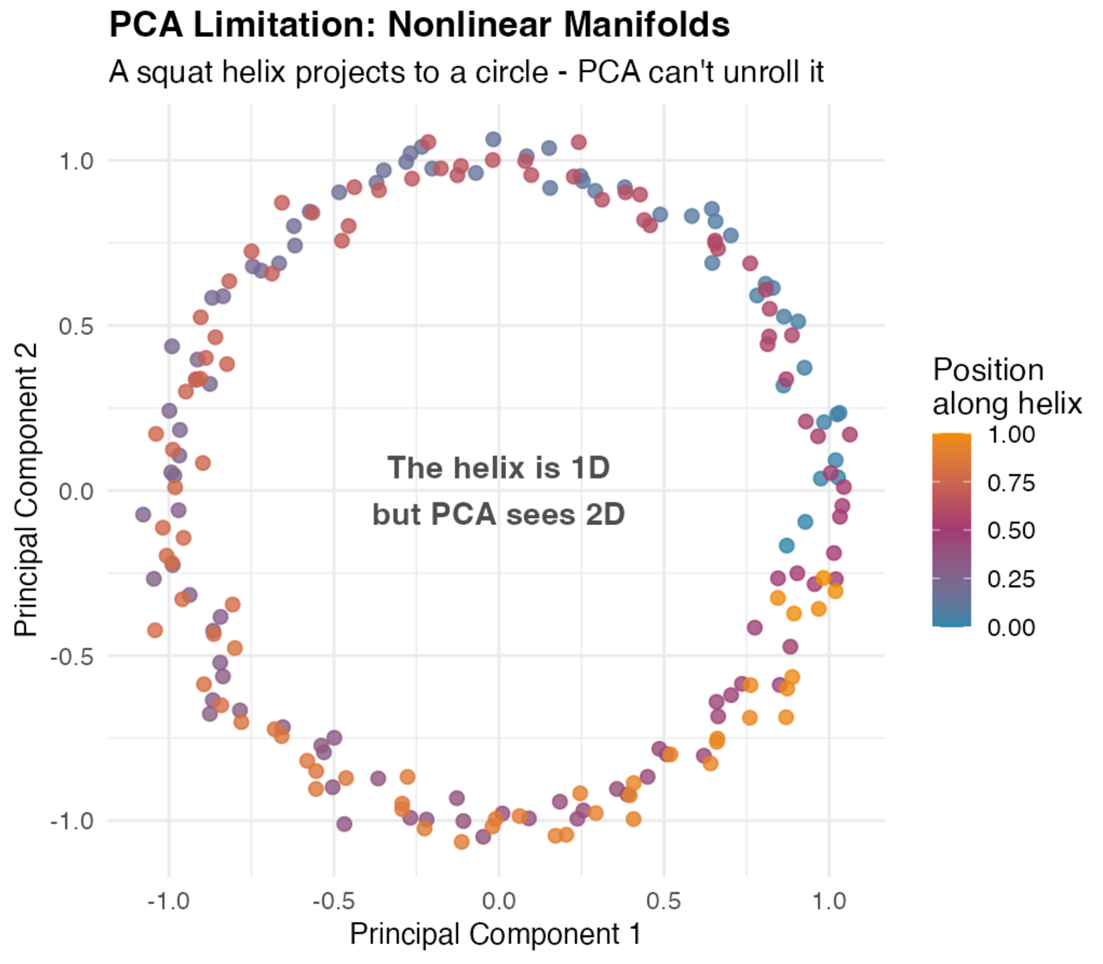


Where PCA Fails

- PCA assumes the structure that "matters" is a **linear manifold**
  - This is often **not the case!**

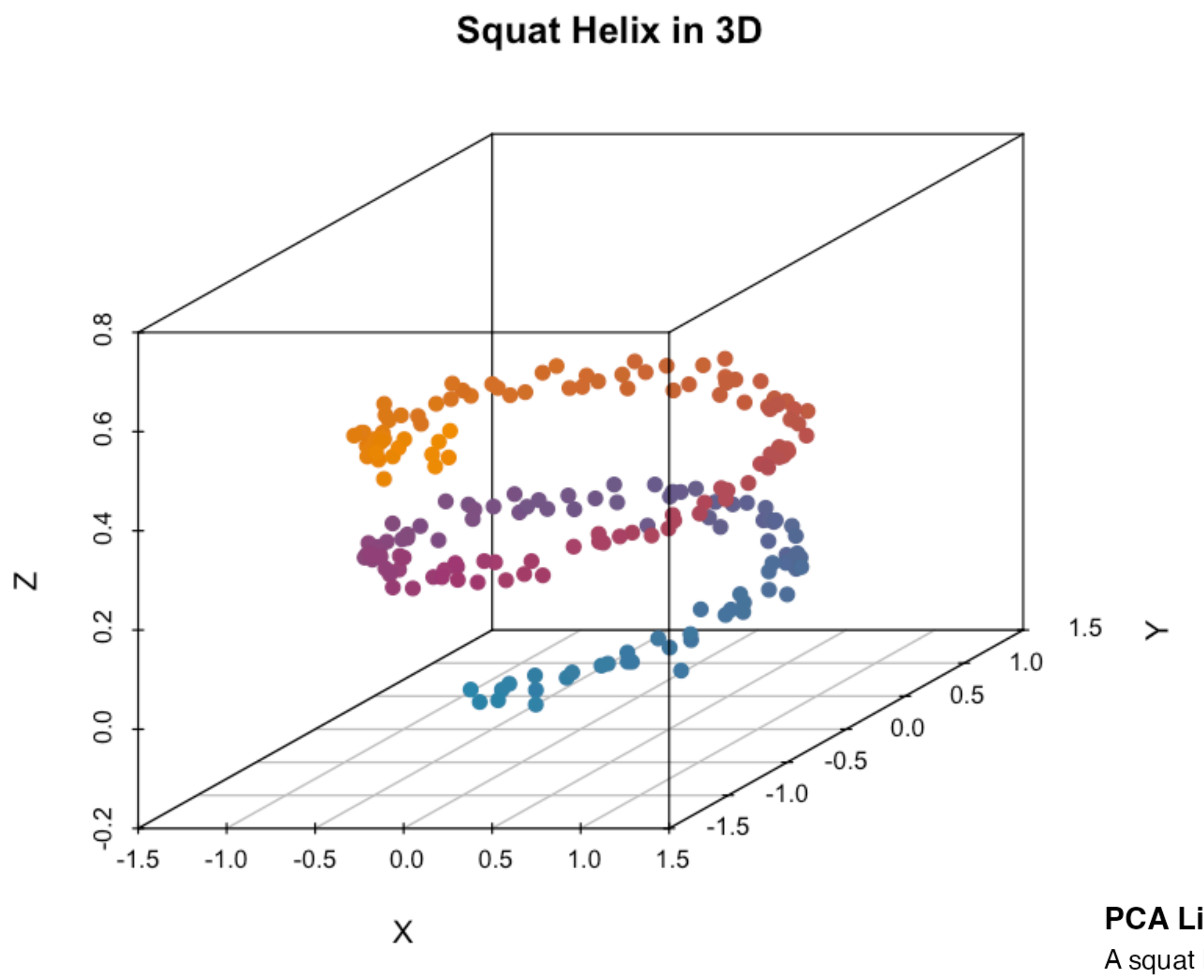


A 1D curve (the helix path) embedded in 3D space

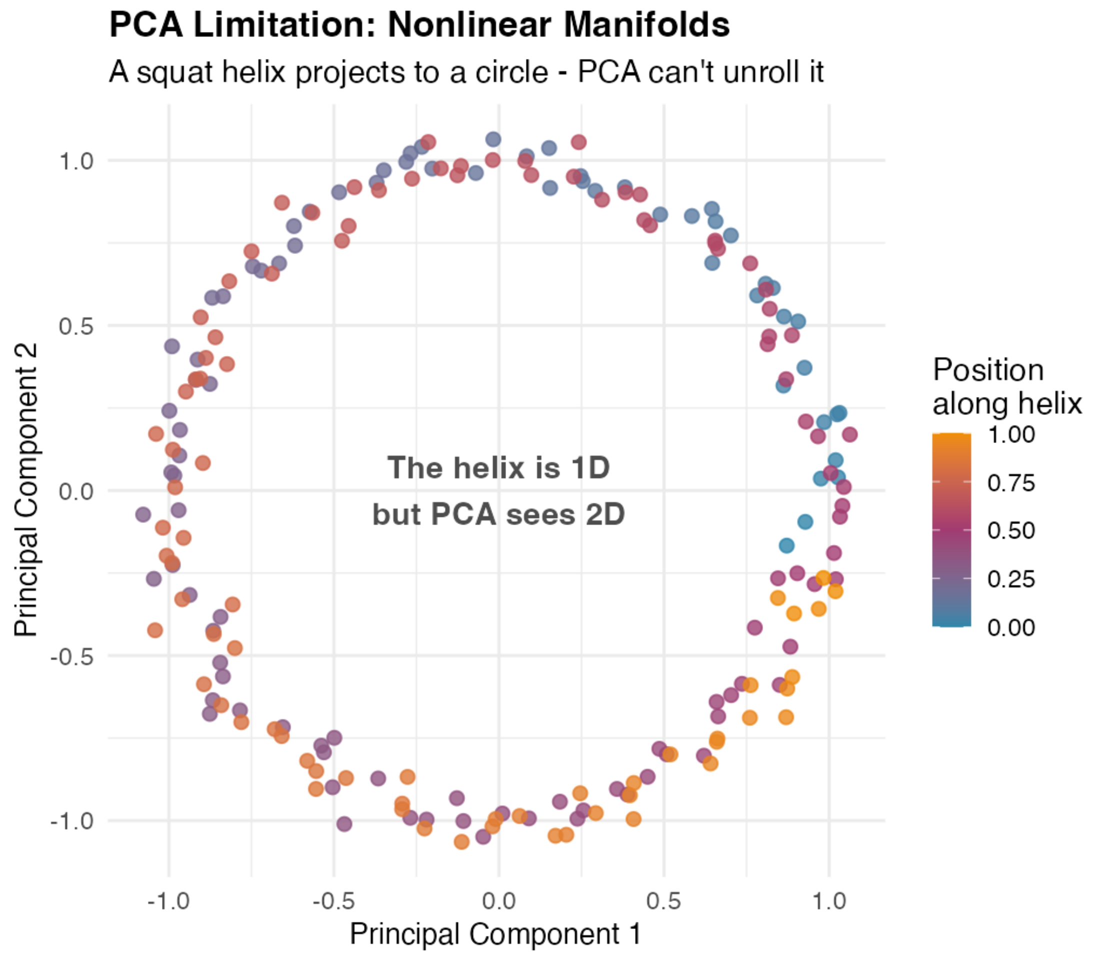


Where PCA Fails

- PCA assumes the structure that "matters" is a **linear manifold**
  - This is often **not the case!**
- "Squat helix": Z-dimension encodes **little variation**, but **most of the information!**
  - Technically a 1D manifold ("ascending" along the helix matters)

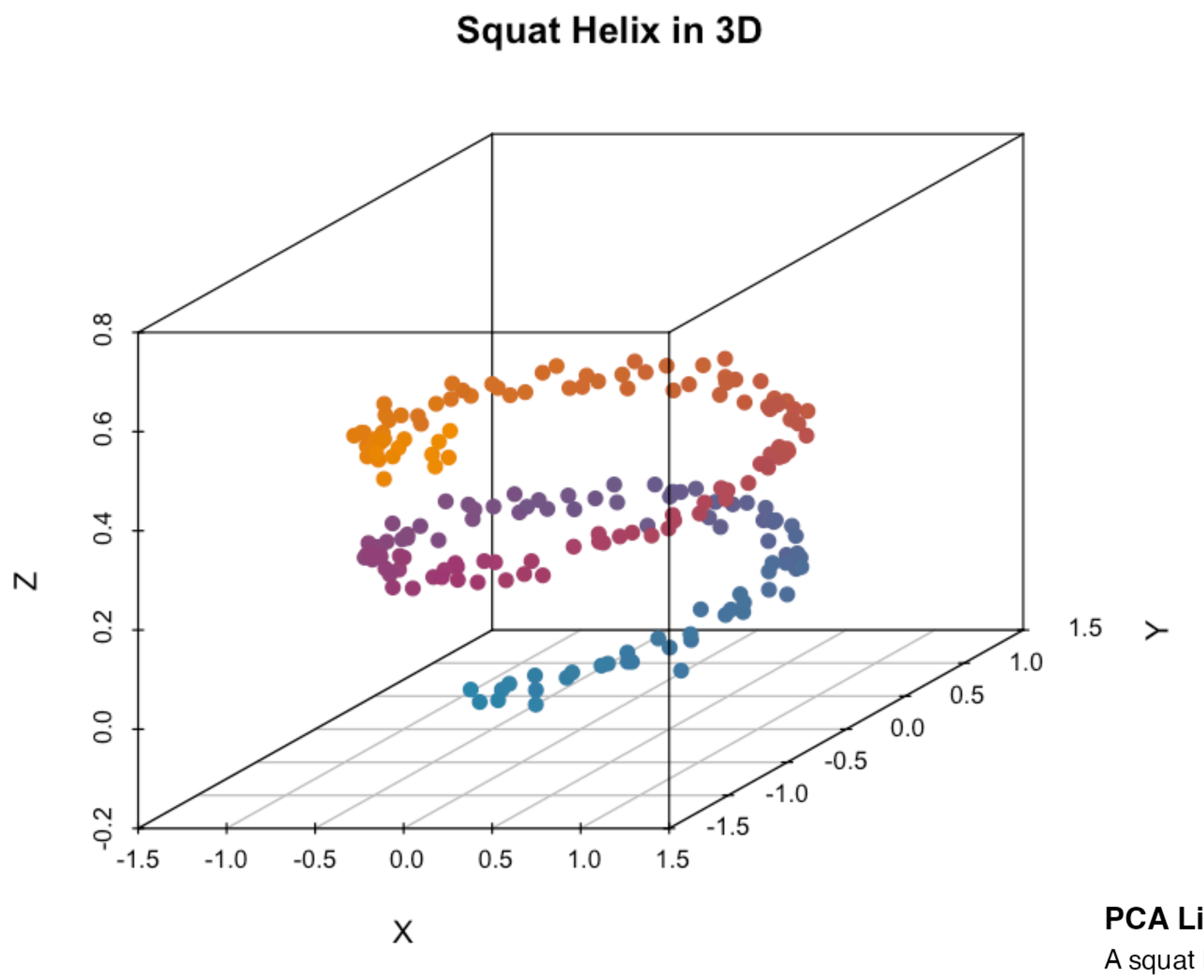


A 1D curve (the helix path) embedded in 3D space

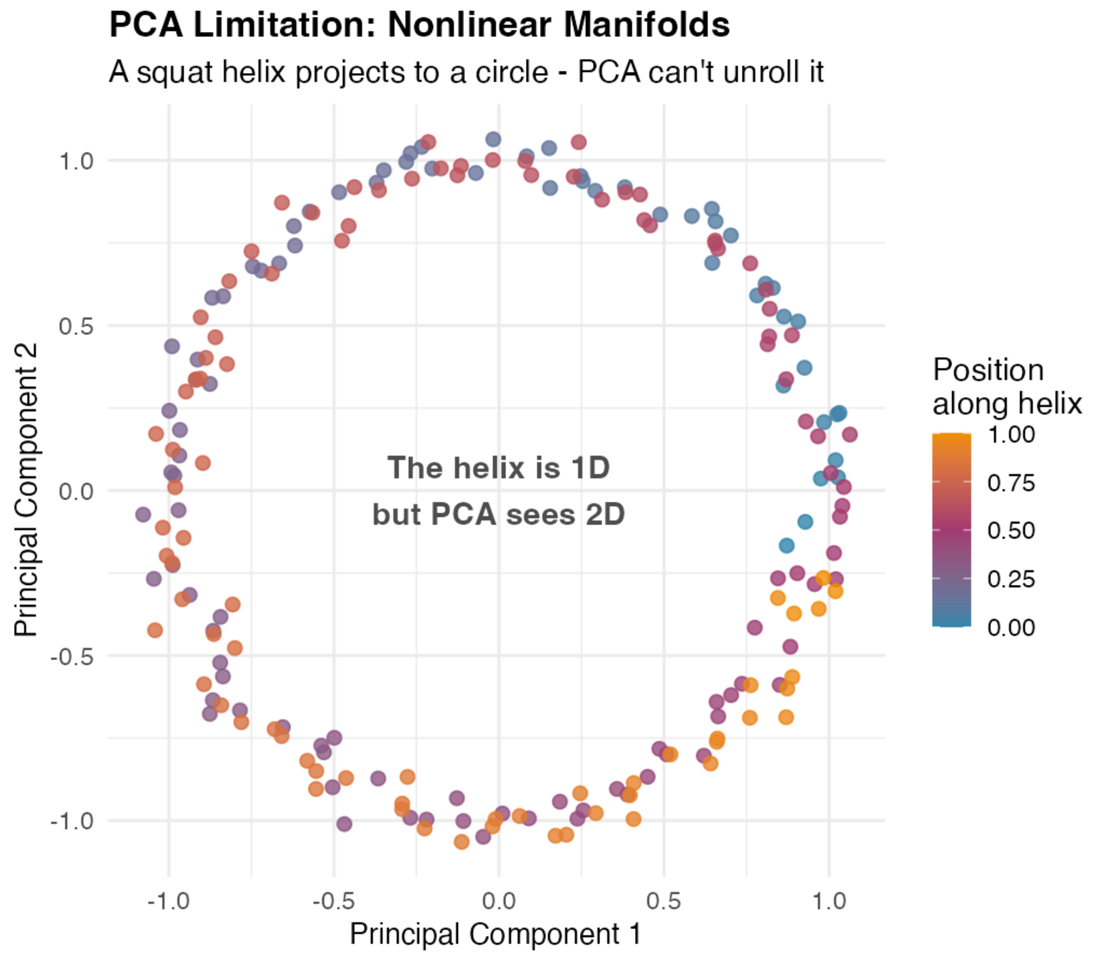


Where PCA Fails

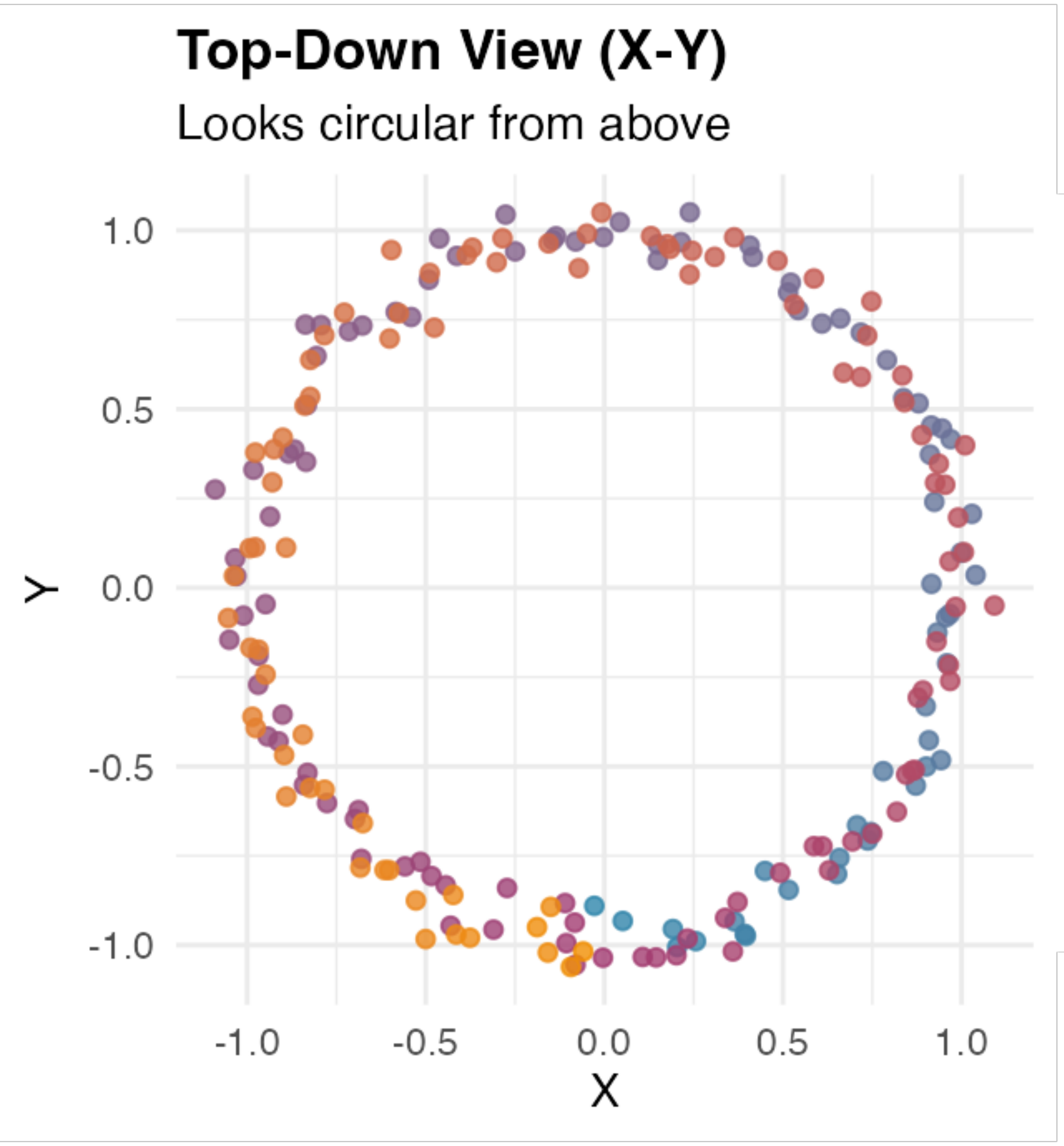
- PCA assumes the structure that "matters" is a **linear manifold**
  - This is often **not the case!**
- "Squat helix": Z-dimension encodes **little variation**, but **most of the information!**
  - Technically a 1D manifold ("ascending" along the helix matters)
- PCA will **flatten** this to a **circle**
  - **Erases** the helix information!

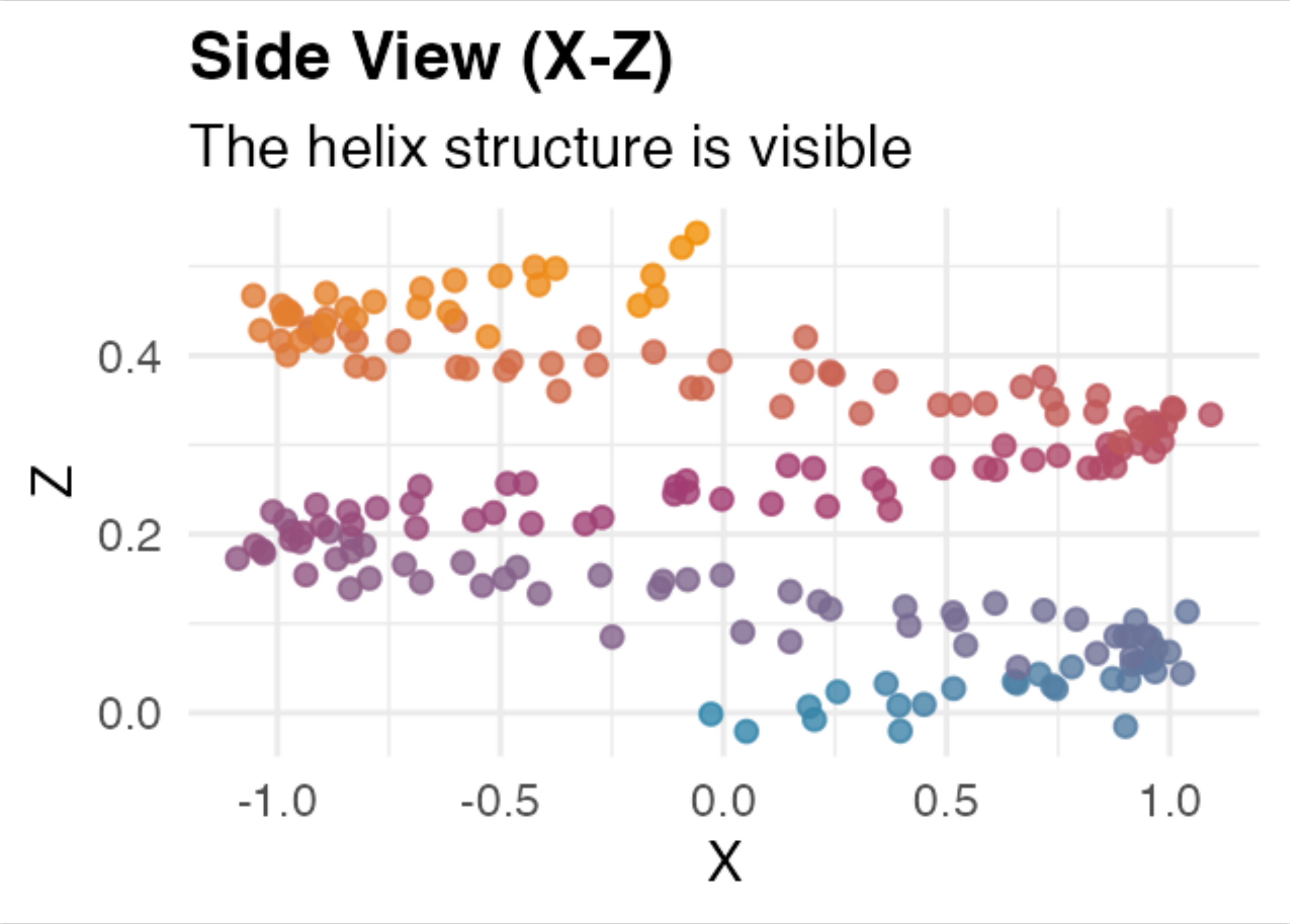


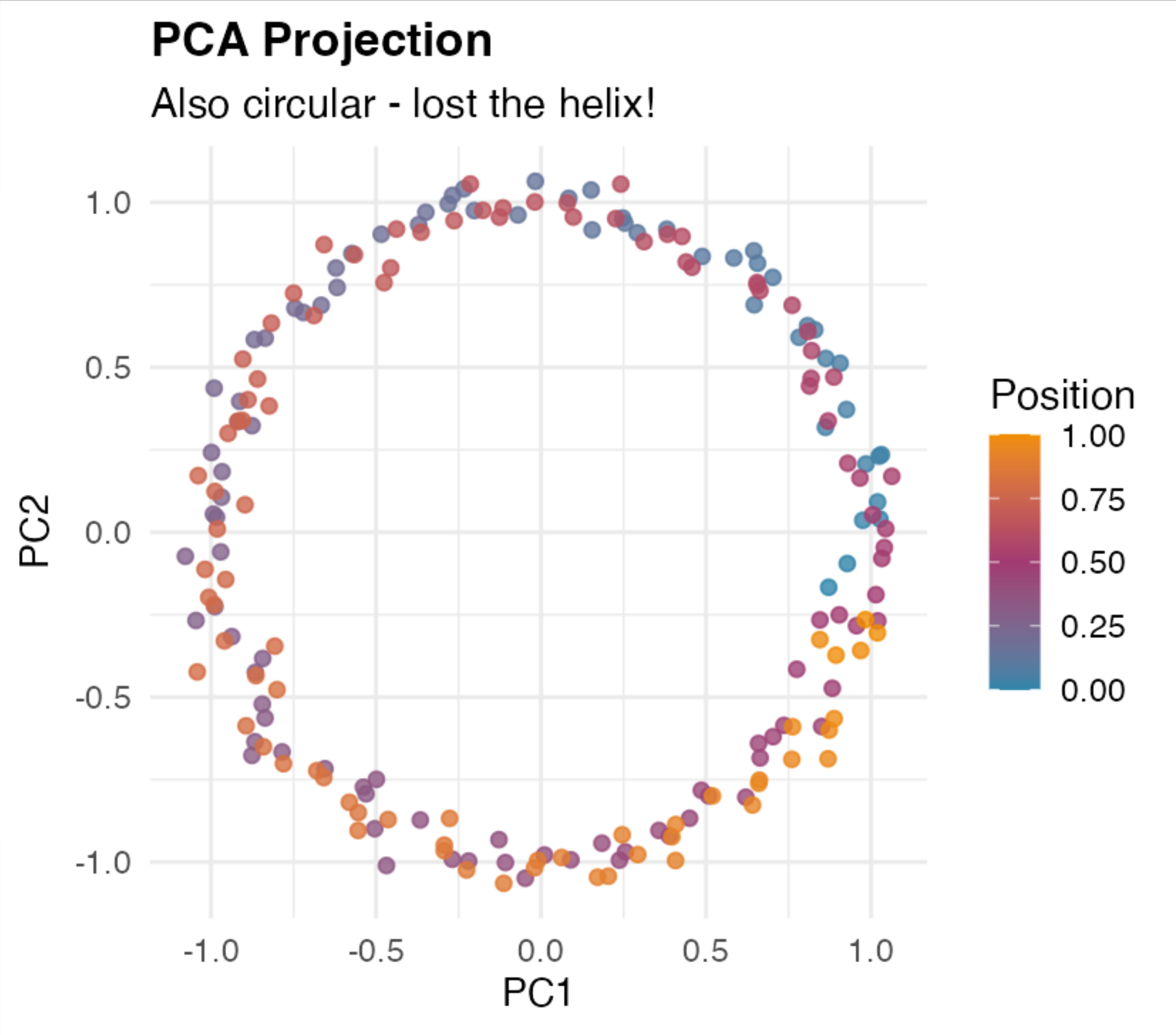
A 1D curve (the helix path) embedded in 3D space



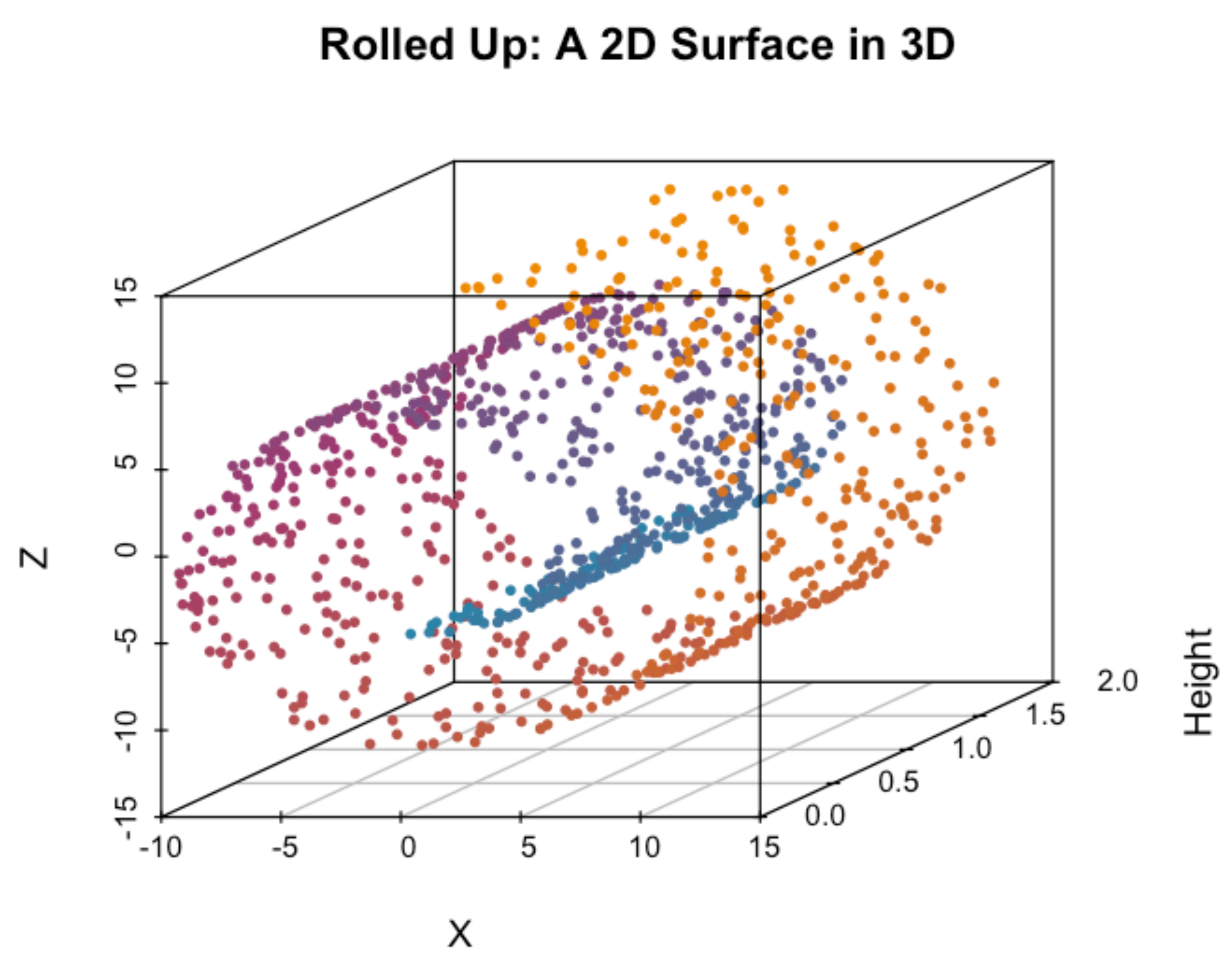
Helix Example

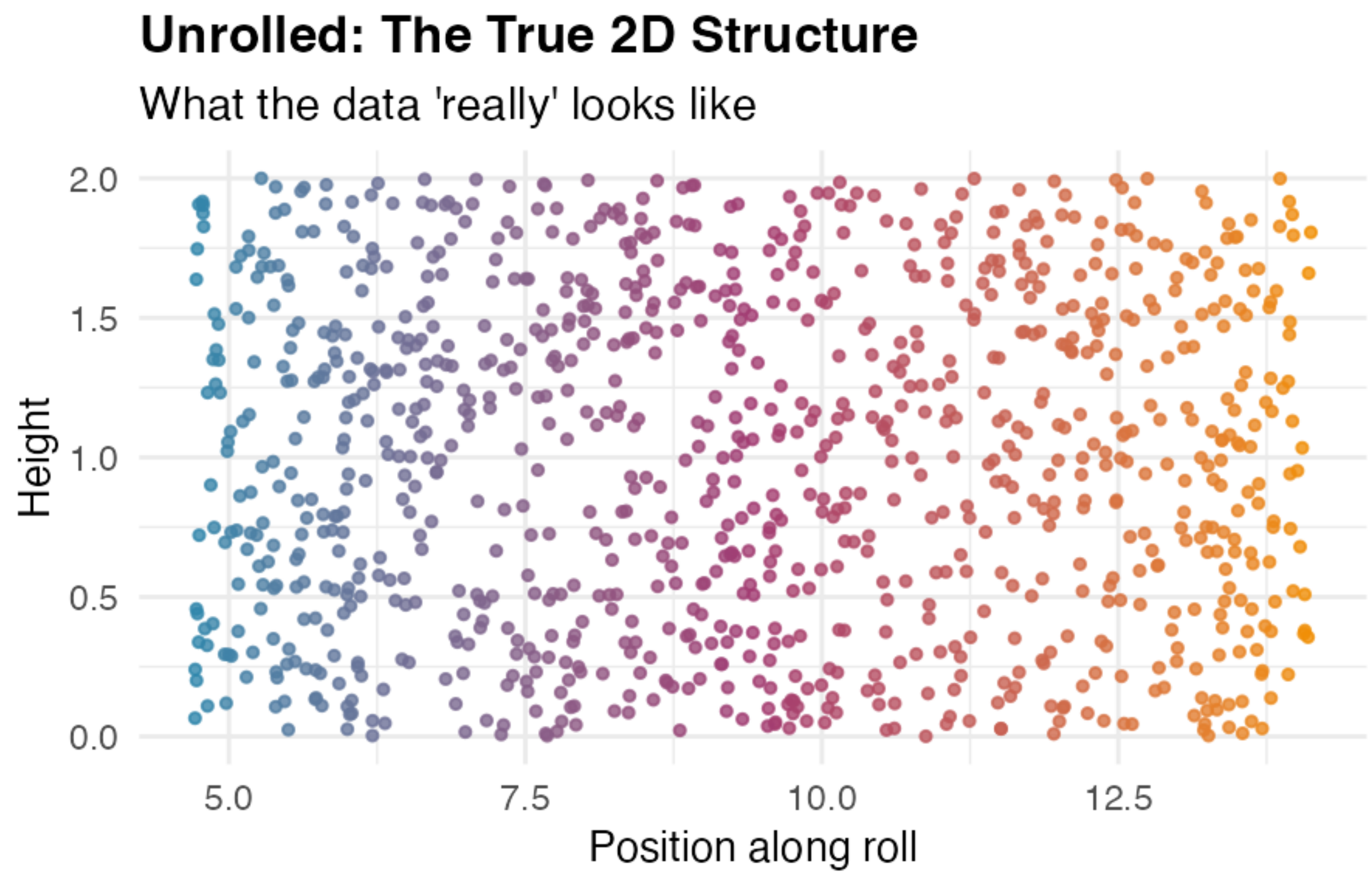






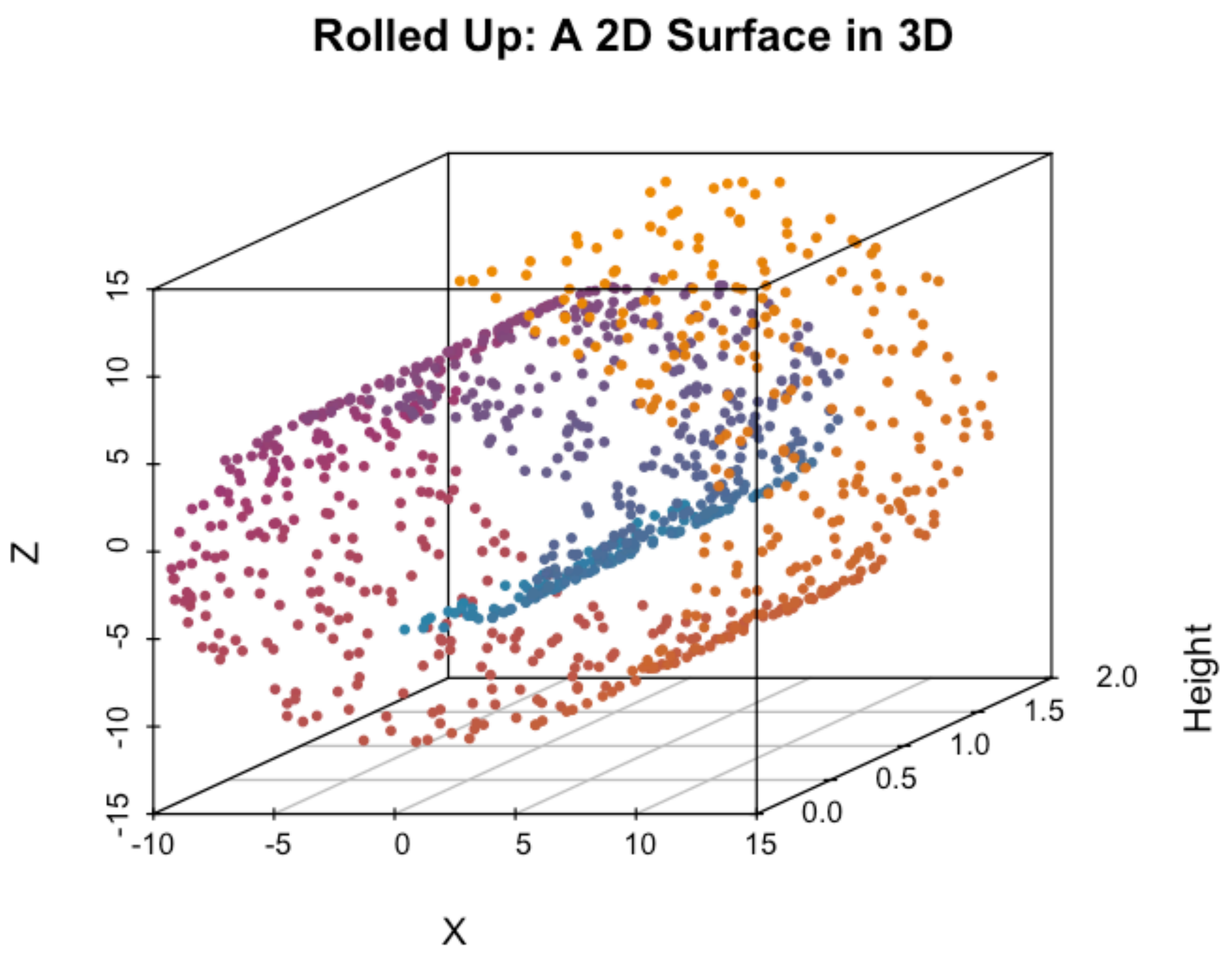
Confound: Non-Linear Manifolds

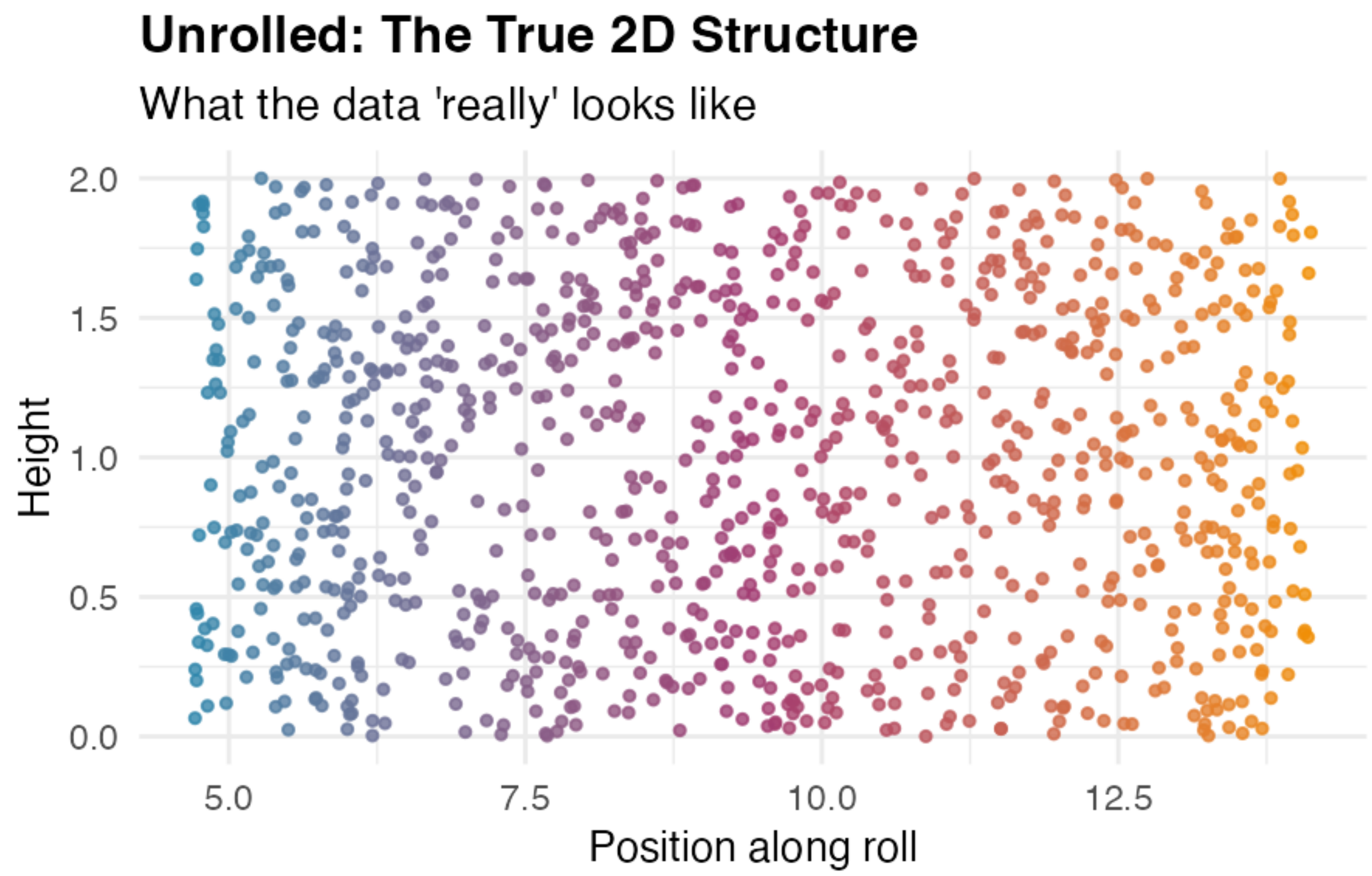




Confound: Non-Linear Manifolds

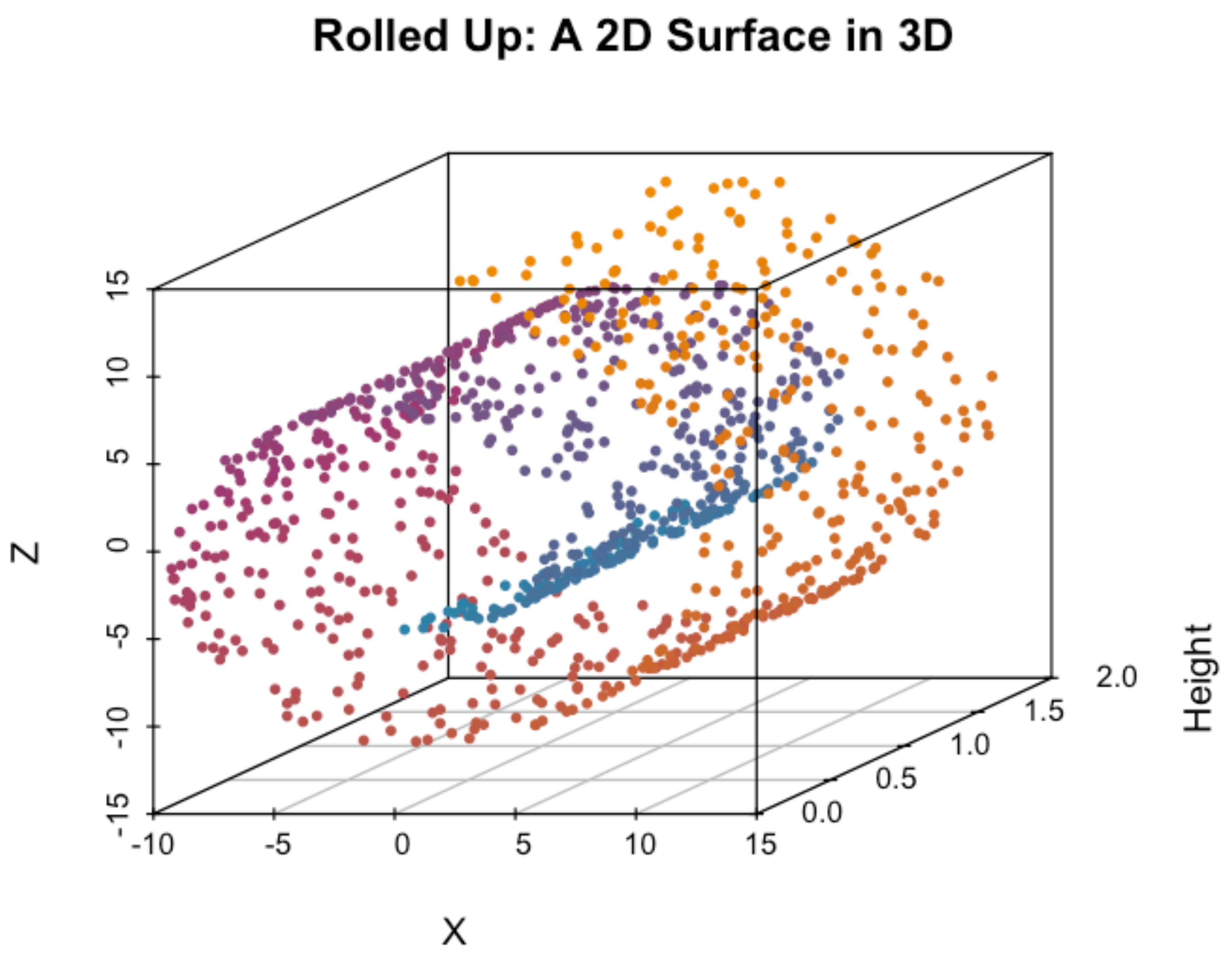
- PCA fails when the "actual" manifold is **non-linear**
  - I.e. the **inductive bias hurts us!**

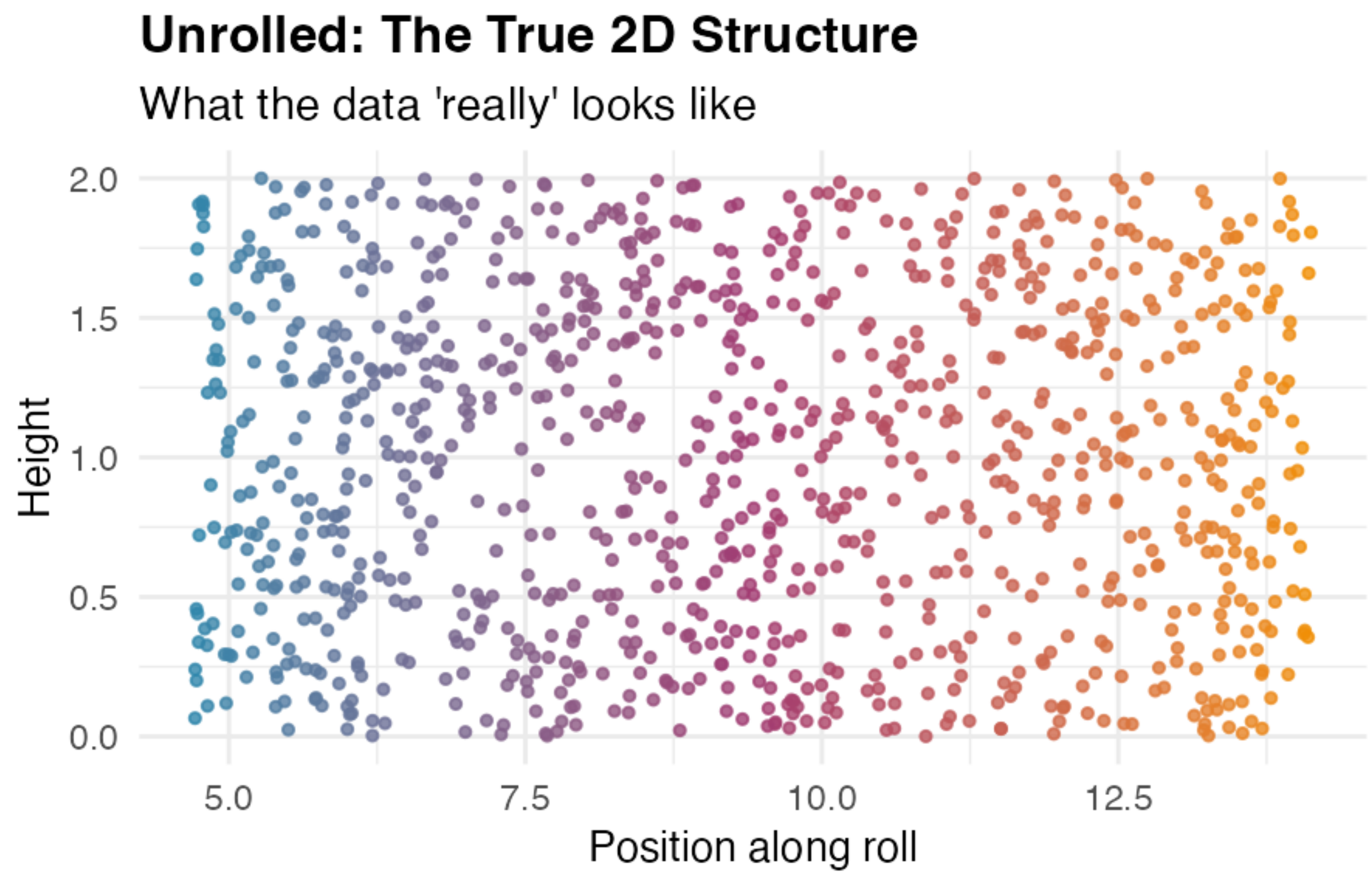




Confound: Non-Linear Manifolds

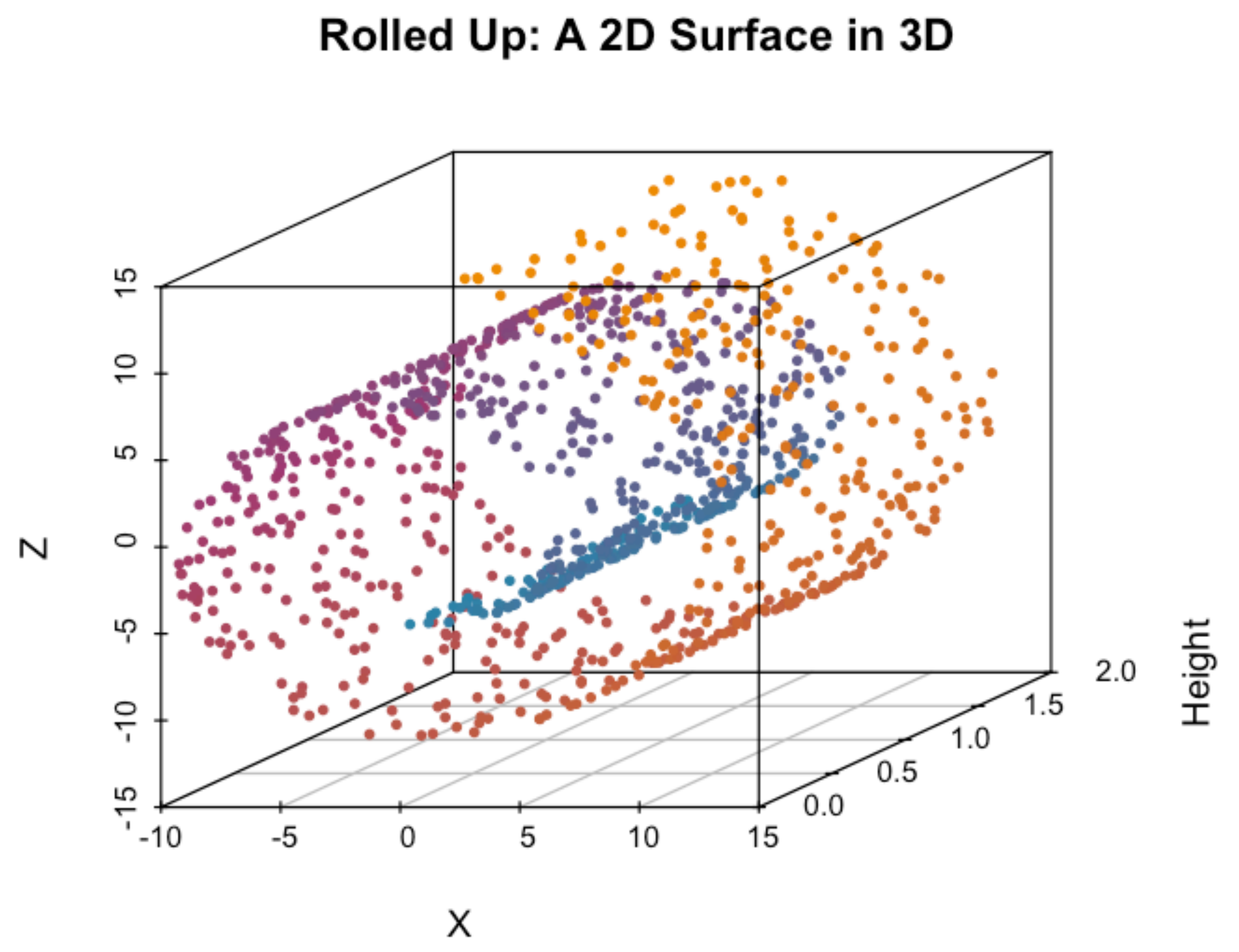
- PCA fails when the "actual" manifold is **non-linear**
  - I.e. the **inductive bias hurts us!**
- Manifolds can have **arbitrarily complex shapes**
  - E.g. "rolling up" a 2D surface

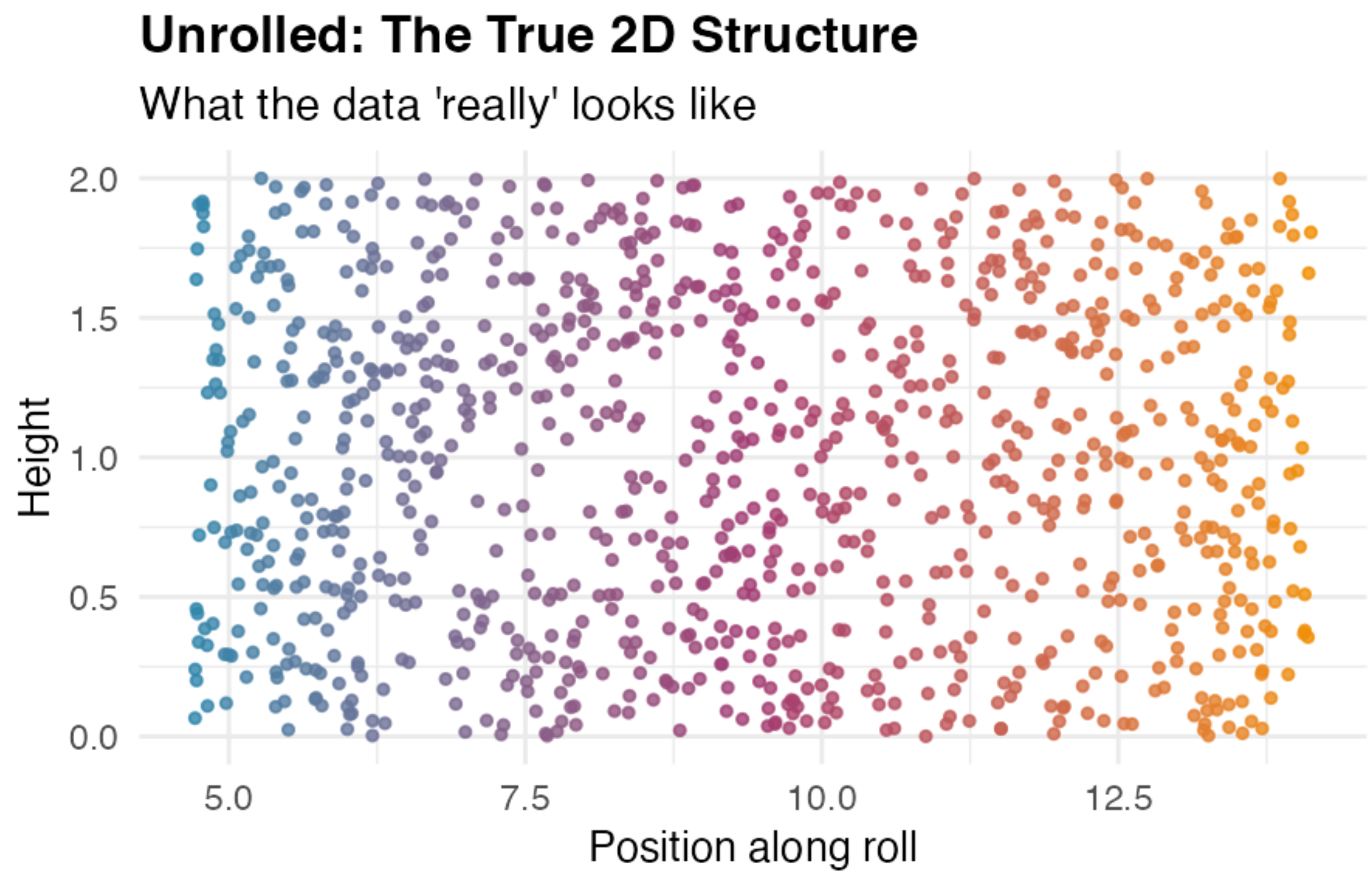




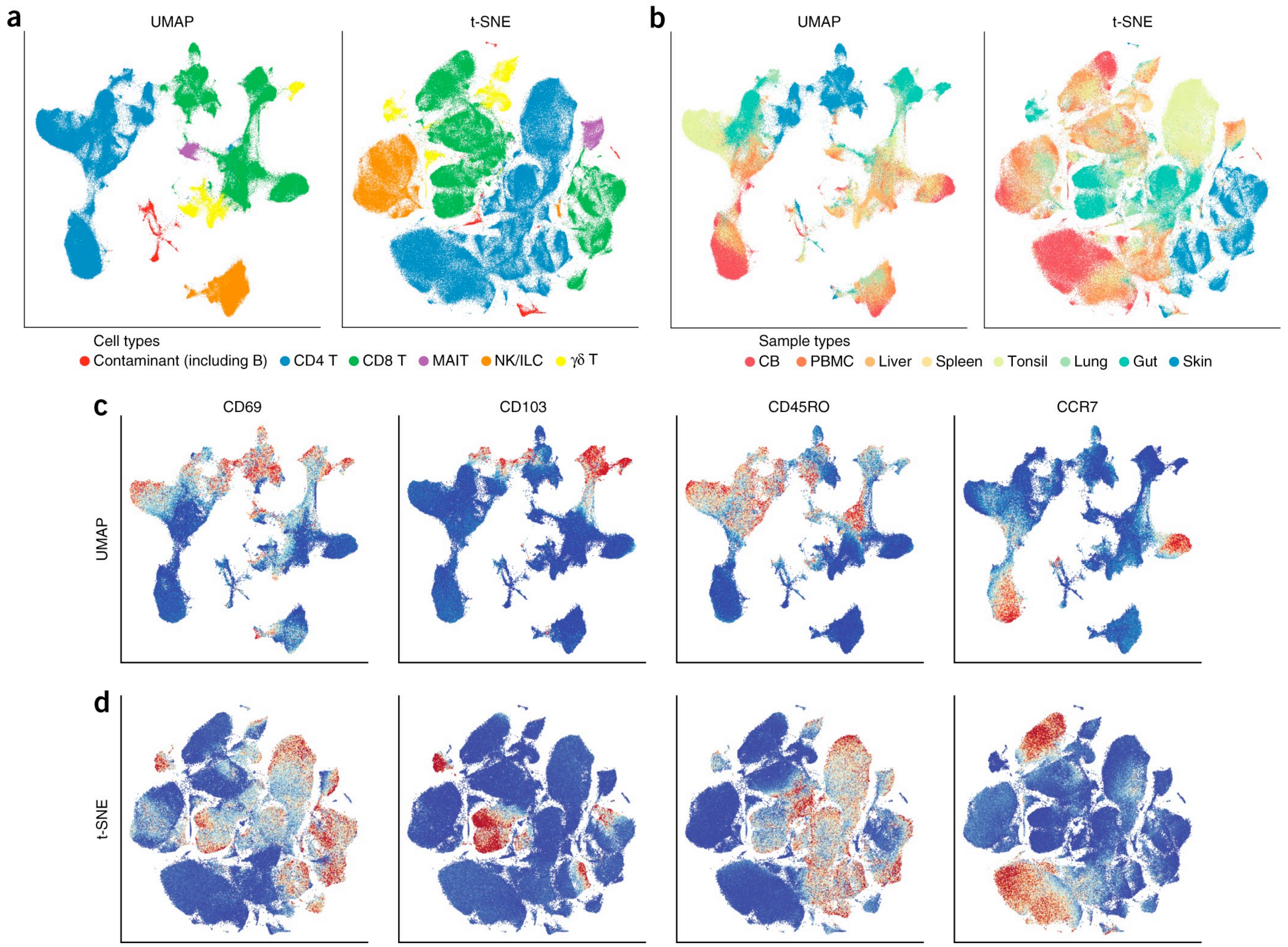
Confound: Non-Linear Manifolds

- PCA fails when the "actual" manifold is **non-linear**
  - I.e. the **inductive bias hurts us!**
- Manifolds can have **arbitrarily complex shapes**
  - E.g. "rolling up" a 2D surface
- How do we deal with this?
  - Algorithms to **learn non-linear manifolds**



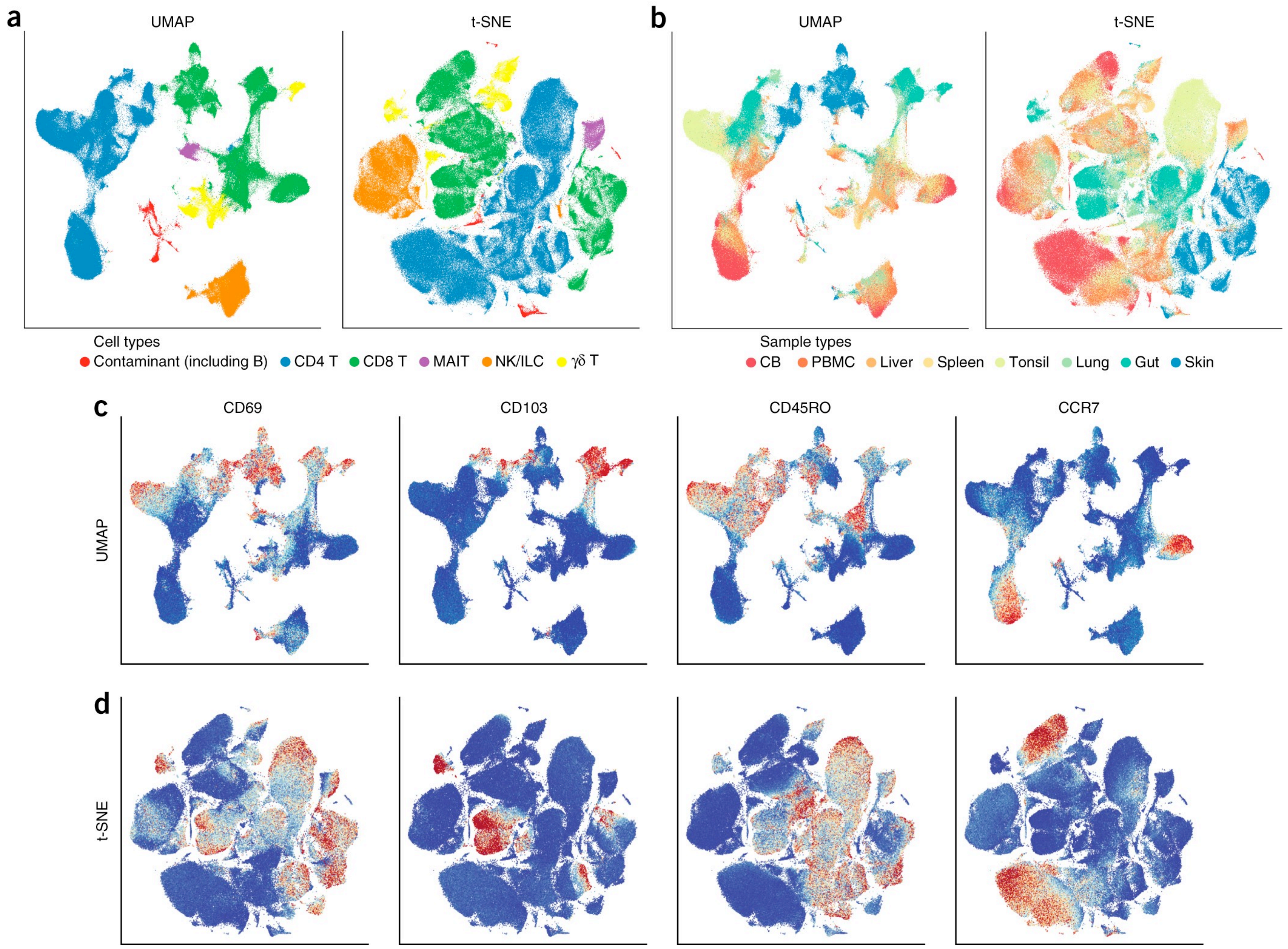


Related Non-Linear Algorithms



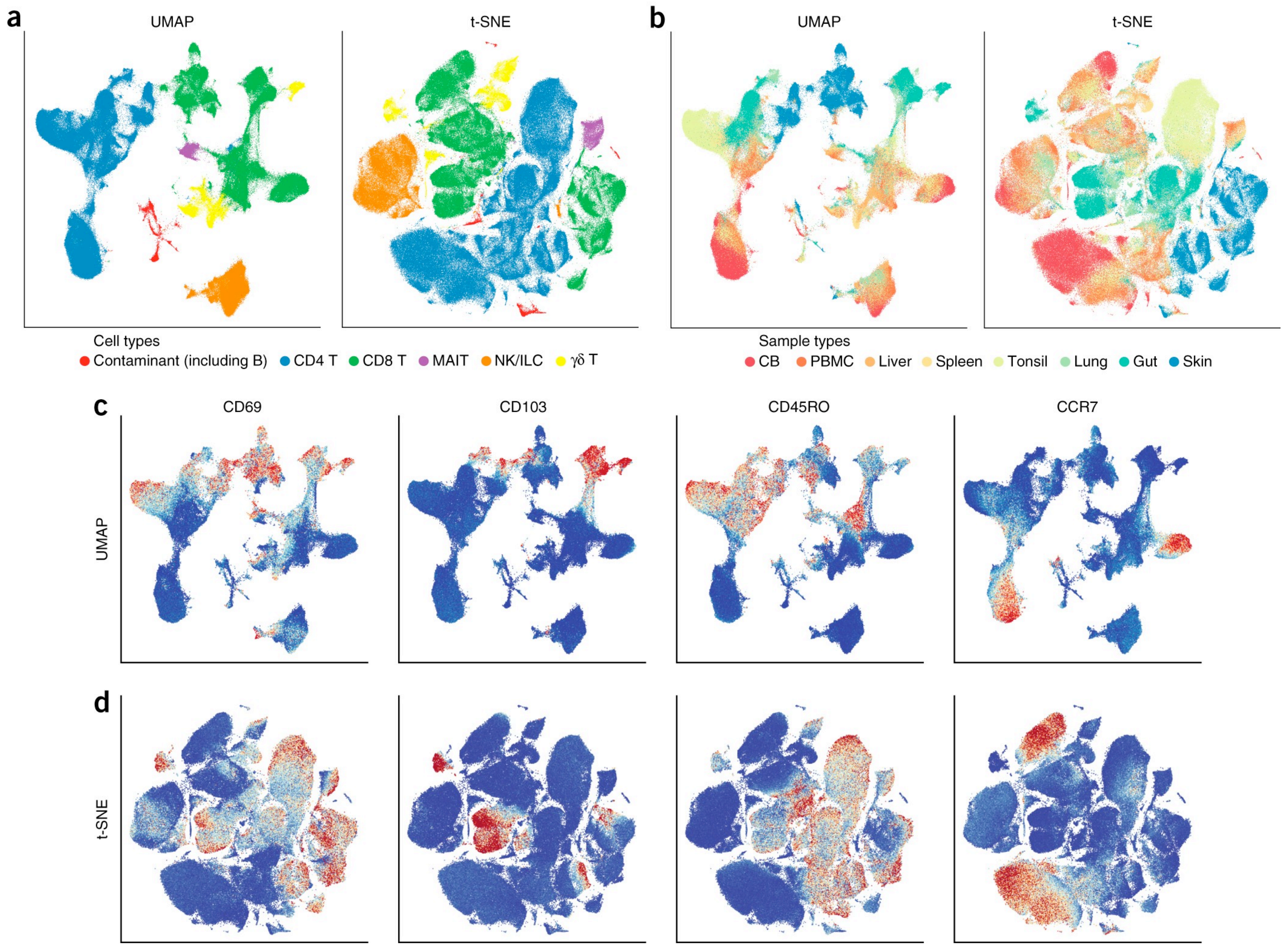
Related Non-Linear Algorithms

- **t-SNE** and **UMAP** resemble PCA input/output but...
  - They can learn **non-linear manifolds**
  - Focus more on preserving **local structure** where PCA preserves global structure



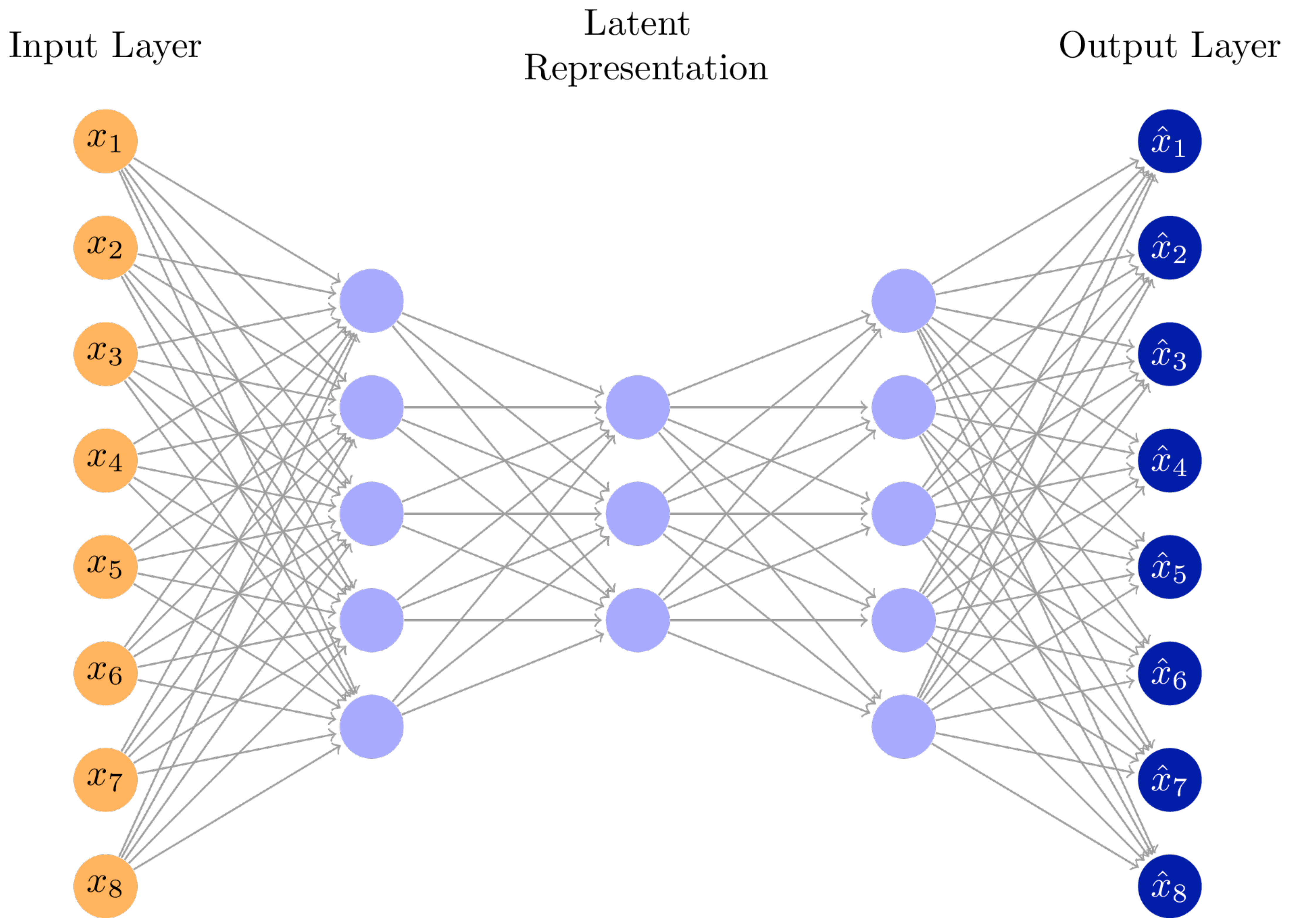
Related Non-Linear Algorithms

- **t-SNE** and **UMAP** resemble PCA input/output but...
  - They can learn **non-linear manifolds**
  - Focus more on preserving **local structure** where PCA preserves global structure
- Caveats: mostly good for **visualization** and **exploratory analysis**
  - The distance between output clusters is **not interpretable**
  - Very sensitive to **hyper-parameters**
  - Output features tend to be **not good for downstream ML**



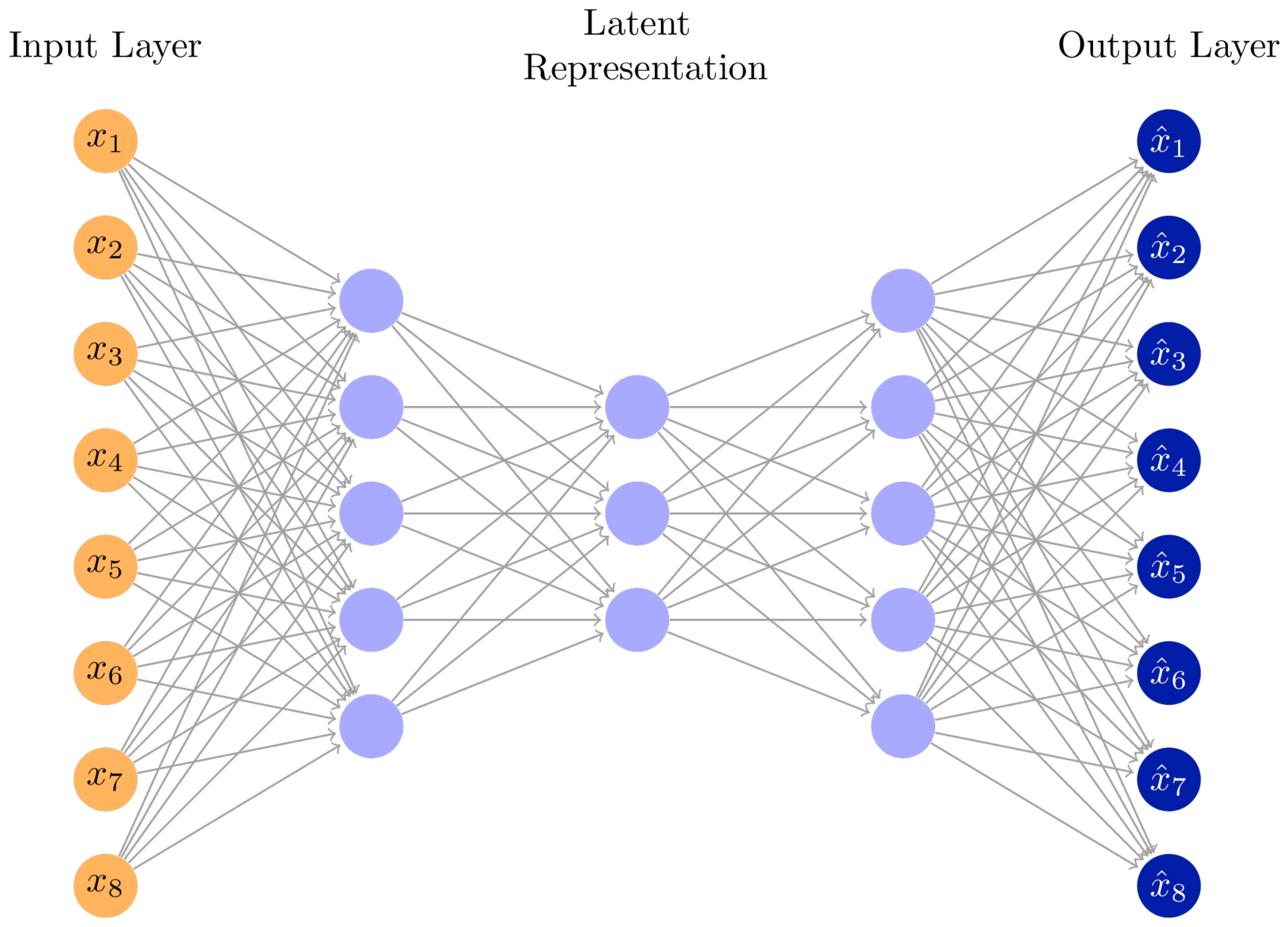
Autoencoders

Autoencoders: Basic Idea



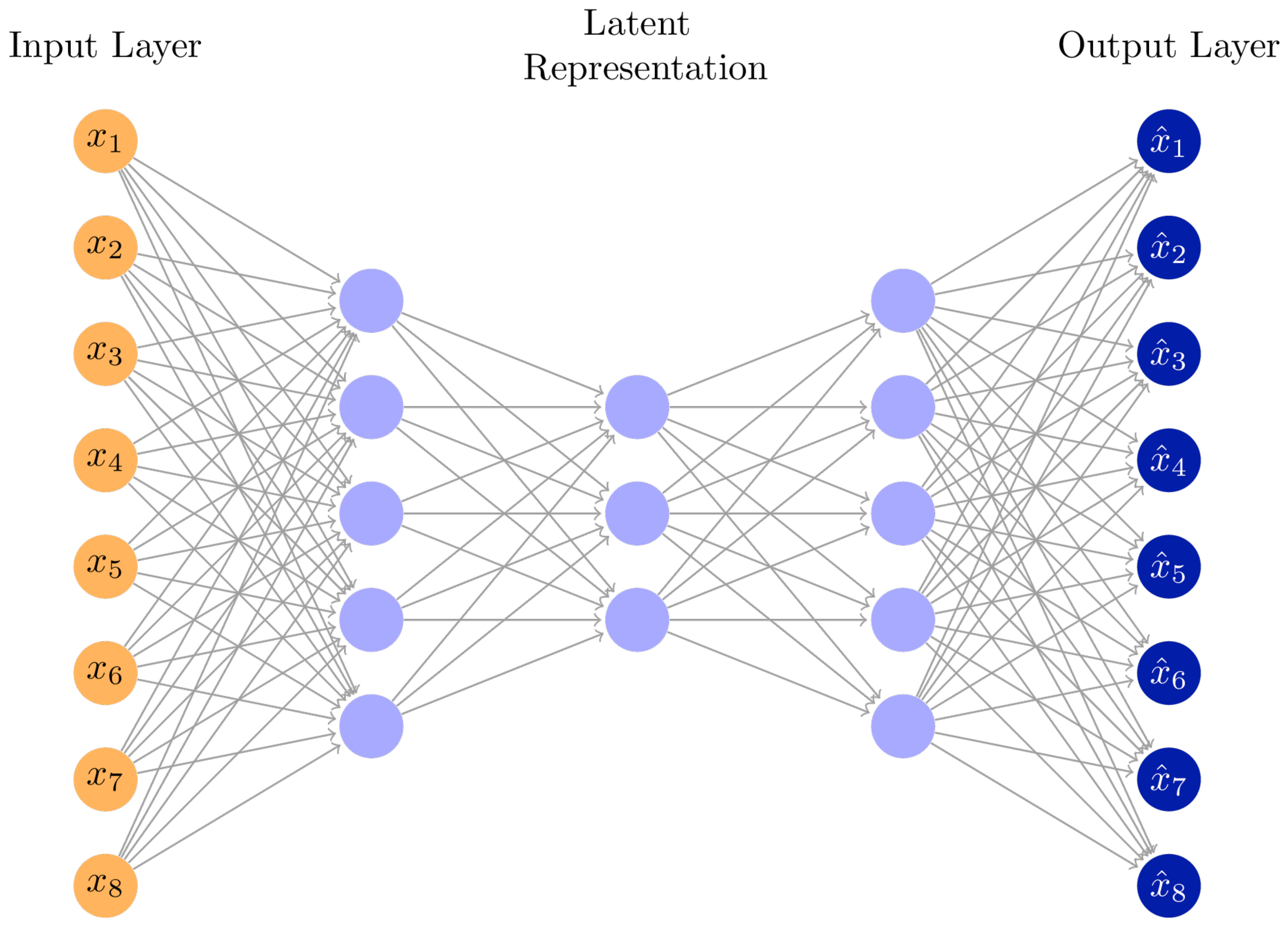
Autoencoders: Basic Idea

- Autoencoders are **Neural Network-based** algorithms that **learn to compress data**

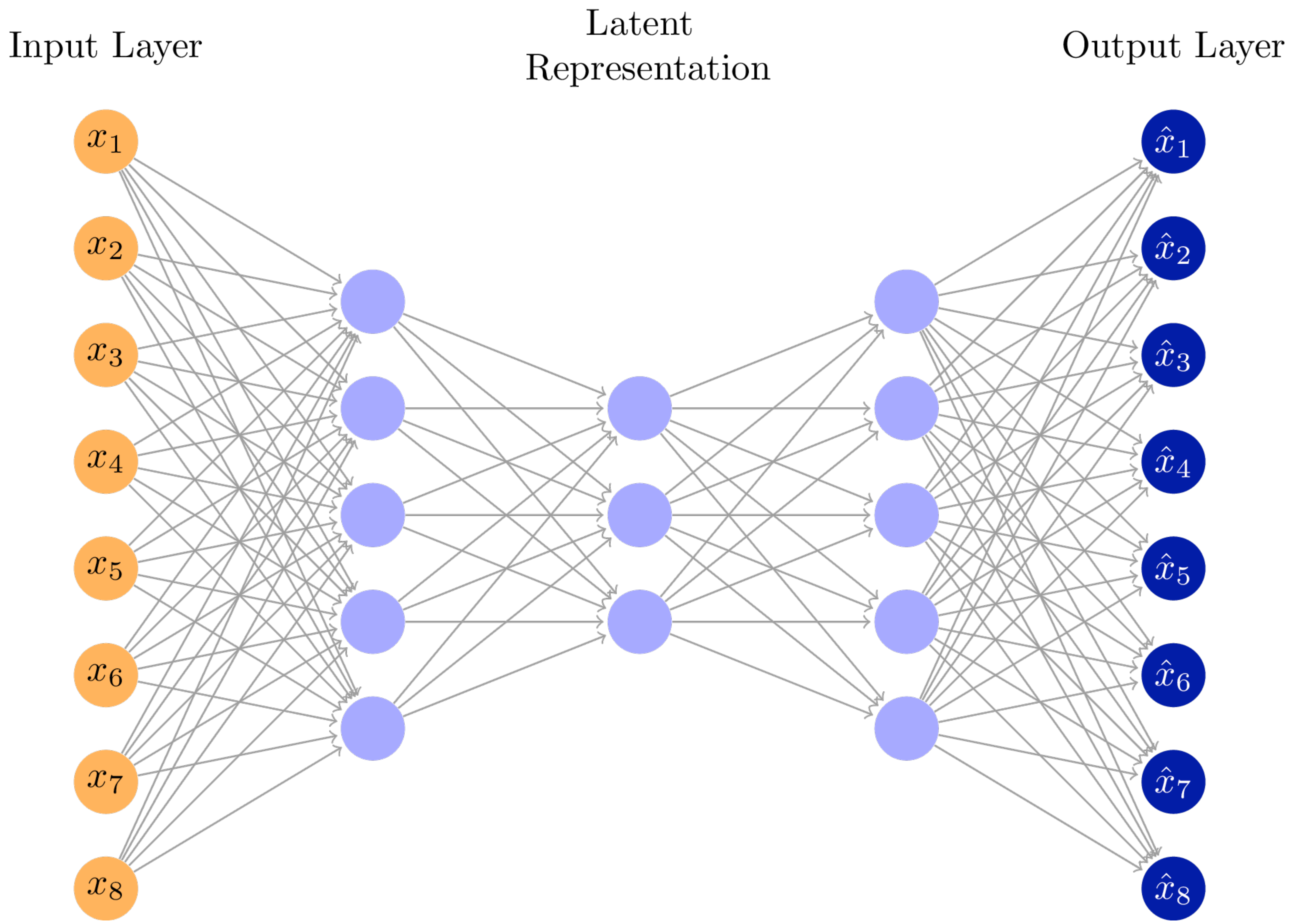


Autoencoders: Basic Idea

- Autoencoders are **Neural Network-based** algorithms that **learn to compress data**
- Learn a low-dimensional **bottleneck representation**, which can be used to **reconstruct the input**
  - This forces dimensionality reduction with **minimal loss of information**

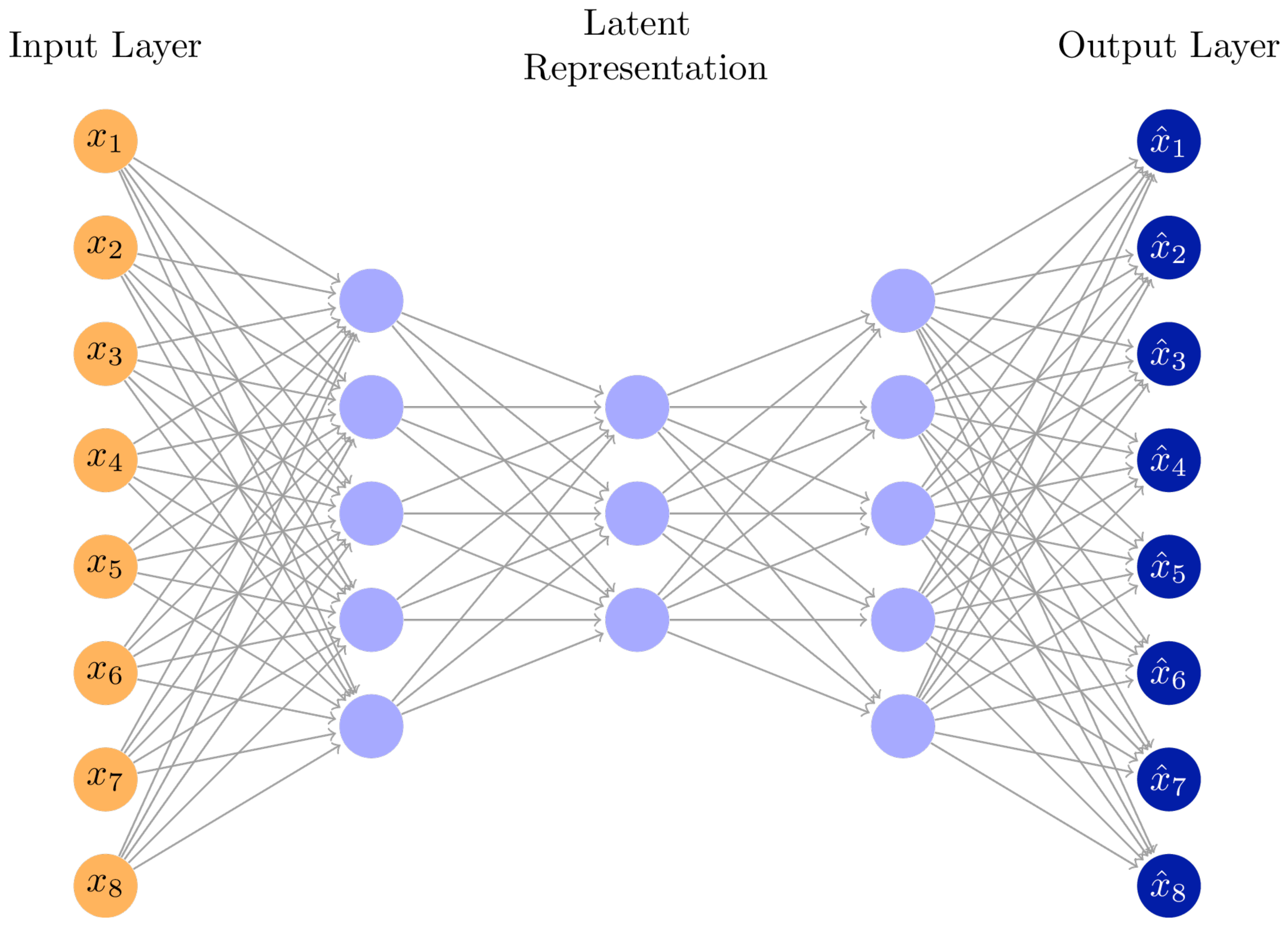


Why does this work?



Why does this work?

- Can learn non-linear manifolds because **NNs apply succcessive non-linear transformations**
  - Can think of it as NN learning to **"unravel"** or **"untangle"** the complex non-linear manifold



Why does this work?

- Can learn non-linear manifolds because **NNs apply succcessive non-linear transformations**
    - Can think of it as NN learning to **"unravel"** or **"untangle"** the complex non-linear manifold
- High-dimensional representation is **simplified** to what we hope is its "intrinsic" dimension