COMP4131: Data Modelling and Analysis

Lecture 5: Unsupervised Learning

Kian Ming Lim

University of Nottingham Ningbo China

kian-ming.lim@nottingham.edu.cn

March 12, 2025

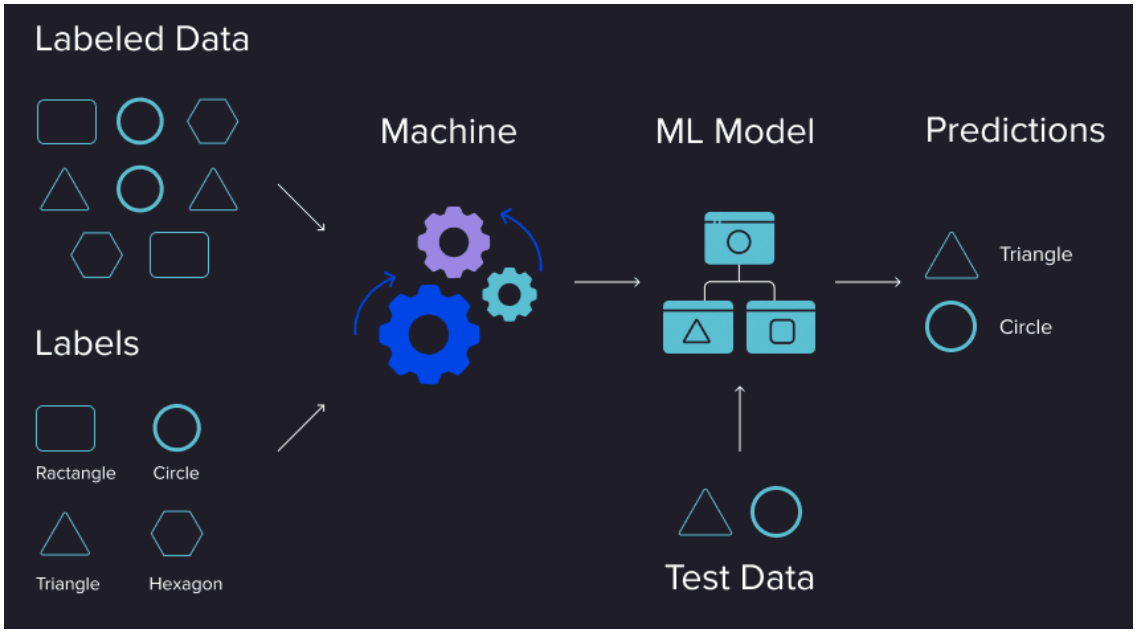
Outline

Introduction to Unsupervised Learning

What is Supervised Learning?

- **Definition**: A type of machine learning where the model learns from labeled data to predict outcomes.

- **Key Concepts**:
  - Explicit supervision with labeled data (input-output pairs).
  - The algorithm learns a mapping from inputs to outputs.
  - Commonly used for classification and regression tasks.

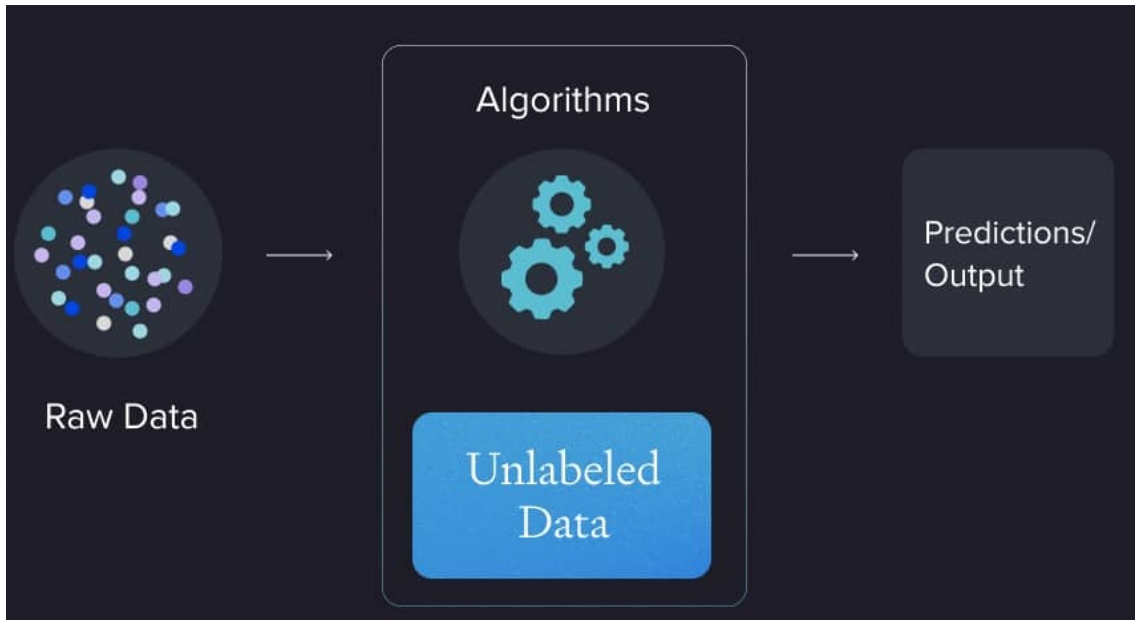


Source: Eleks

What is Unsupervised Learning?

- **Definition**: A type of machine learning where the model learns patterns from unlabeled data.

- **Key Concepts**:
  - No explicit supervision.
  - The algorithm identifies hidden structures or patterns.
  - Commonly used for clustering and dimensionality reduction.



Source: Eleks

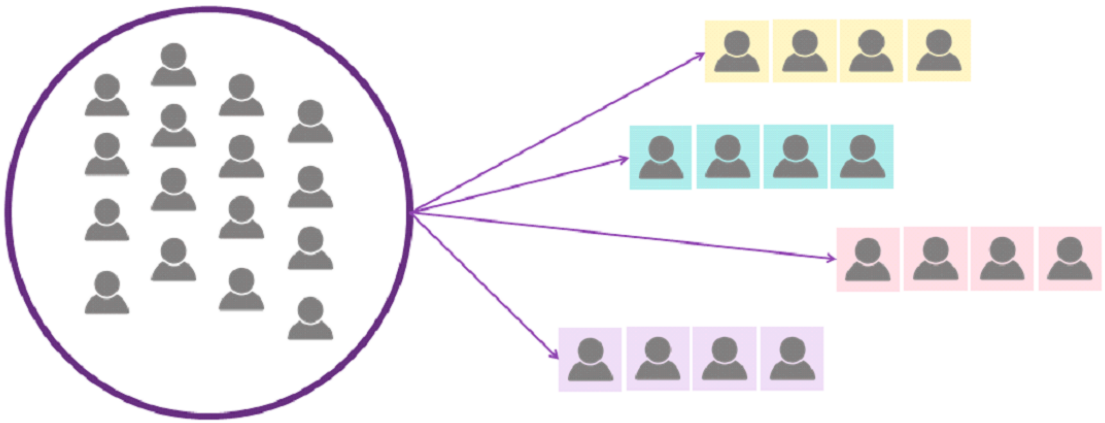
Supervised vs. Unsupervised Learning

Aspect	**Supervised Learning**	**Unsupervised Learning**
Data	Labeled	Unlabeled
Goal	Predict outcomes	Discover patterns
Examples	Classification, Regression	Clustering, Dimensionality Reduction

Table: Comparison of Supervised and Unsupervised Learning

Applications of Unsupervised Learning

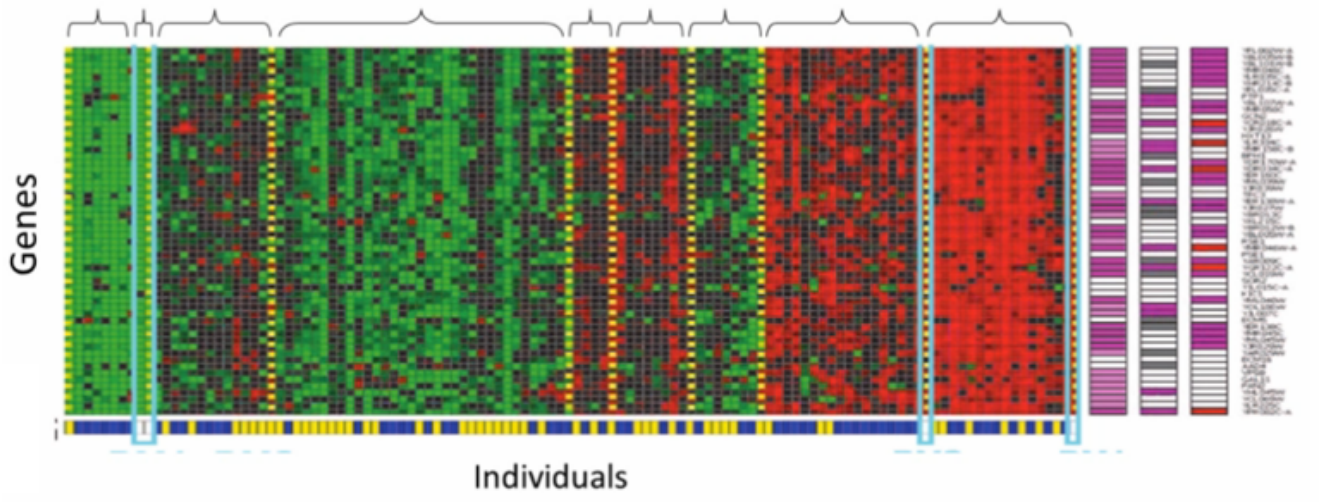
- **Customer Segmentation**: Grouping customers based on purchasing behavior.
- **Anomaly Detection**: Identifying unusual patterns in data (e.g., fraud detection).
- **Image Compression**: Reducing the size of images using clustering.
- **Recommendation Systems**: Suggesting products based on user behavior.



Source: Aspire System, Arun Lakshmanan

Applications of Unsupervised Learning

- **Google News**: Groups news stories into cohesive groups.
- **Genomics Microarray Data**: Cluster individual's genes into types of people.
- **Organize Computer Clusters**: Identify potential weak spots or distribute workload effectively.
- **Social network analysis**: Uncover valuable insights from the structure and dynamics of social networks.



Types of Unsupervised Learning

- **Clustering**: Grouping similar data points (e.g., K-Means, Hierarchical Clustering).
- **Dimensionality Reduction**: Reducing the number of features (e.g., PCA, t-SNE).
- **Association Rule Learning**: Discovering relationships between variables (e.g., Apriori Algorithm).

Source: Machine Learning Crash Course, Google

Clustering

What is Clustering?

- **Definition**: Grouping similar data points together based on their features.

- **Intuition**:
  - Unsupervised learning technique.
  - No predefined labels; the algorithm identifies natural groupings.
  - Used for exploratory data analysis and pattern discovery.

Types of Clustering

- **Partitional Clustering**: Divides data into non-overlapping subsets (e.g., K-Means).
- **Hierarchical Clustering**: Builds a tree of clusters (e.g., Agglomerative, Divisive).
- **Density-Based Clustering**: Groups data based on density (e.g., DBSCAN).
- **Model-Based Clustering**: Assumes data is generated from a mixture of distributions (e.g., Gaussian Mixture Models).

K-Means Clustering

- **Definition**: A type of unsupervised learning used for unlabeled data.
- **Goal**: Find groups in the data, with the number of groups represented by K.
- **Algorithm**:
  - Works iteratively to assign each data point to one of K groups.
  - Clusters data points based on feature similarity.
- **Results**:
  - Centroids of the K clusters (used to label new data).
  - Labels for the training data (each point assigned to a single cluster).

K-means Clustering: Problem and Objective

Given a dataset $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ where each $\mathbf{x}_i \in \mathbb{R}^d$, the goal of K-means is to partition the dataset into k clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ such that the within-cluster sum of squares is minimized.

Objective Function:

$$J(\mathcal{C}, \boldsymbol{\mu}) = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

where:

- $\boldsymbol{\mu}_i$ is the centroid of cluster C_i.
- $\|\mathbf{x} - \boldsymbol{\mu}_i\|^2$ is the squared Euclidean distance between a data point $\mathbf{x}$ and the centroid $\boldsymbol{\mu}_i$.

K-means Clustering: Algorithm Steps

The K-means algorithm proceeds as follows:

1. **Initialization:** Randomly initialize k cluster centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$.
2. **Assignment Step:** Assign each data point $\mathbf{x}_j$ to the nearest centroid:

$$C_i = \{\mathbf{x}_j : \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 \leq \|\mathbf{x}_j - \boldsymbol{\mu}_l\|^2 \, \forall l \neq i\}$$

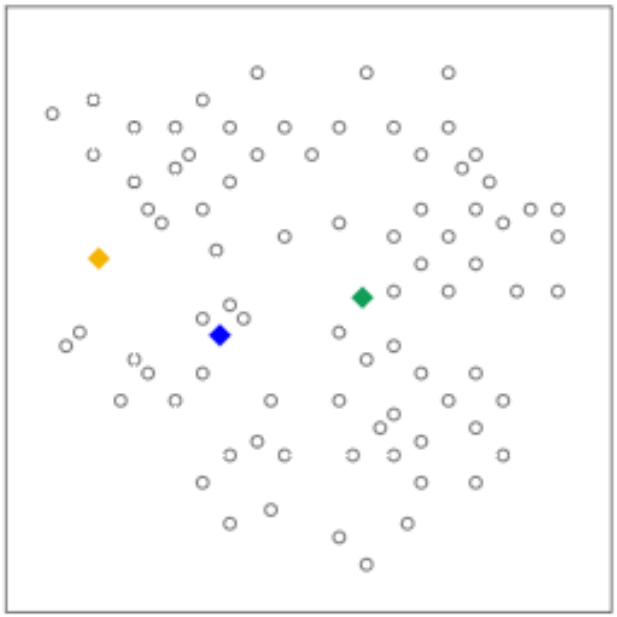
3. **Update Step:** Recompute the centroids as the mean of all points assigned to each cluster:

$$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

4. **Repeat:** Repeat the assignment and update steps until convergence (i.e., when the centroids no longer change significantly or a maximum number of iterations is reached).

K-Means Algorithm - Example

1. Provide an initial guess for K. For this example, we choose $K = 3$
2. Randomly choose K centroids.

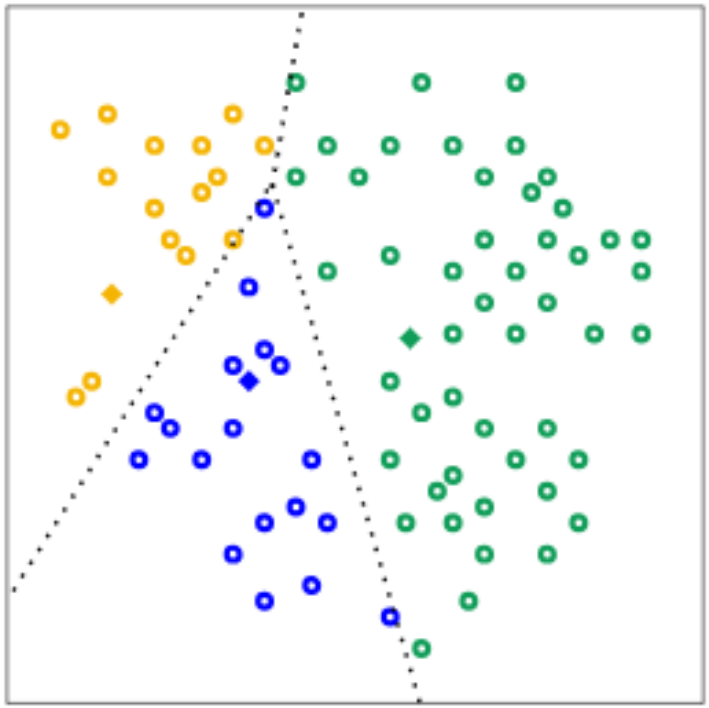


k-means at initialization

Source: Machine Learning Crash Course, Google

K-Means Algorithm - Example

3. Assign each point to the nearest centroid to get K initial clusters.

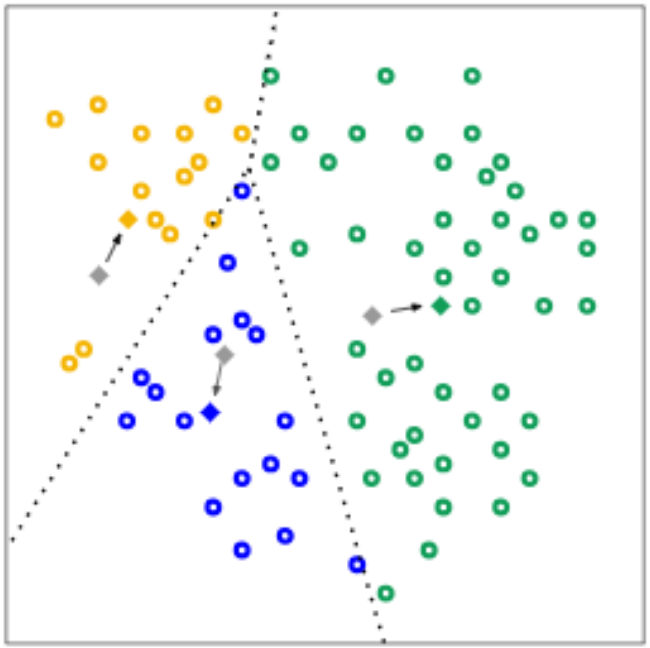


Initial clusters

Source: Machine Learning Crash Course, Google

K-Means Algorithm - Example

4. For each cluster, calculate a new centroid by taking the mean position of all points in the cluster. The arrows in the figure show the change in centroid positions.

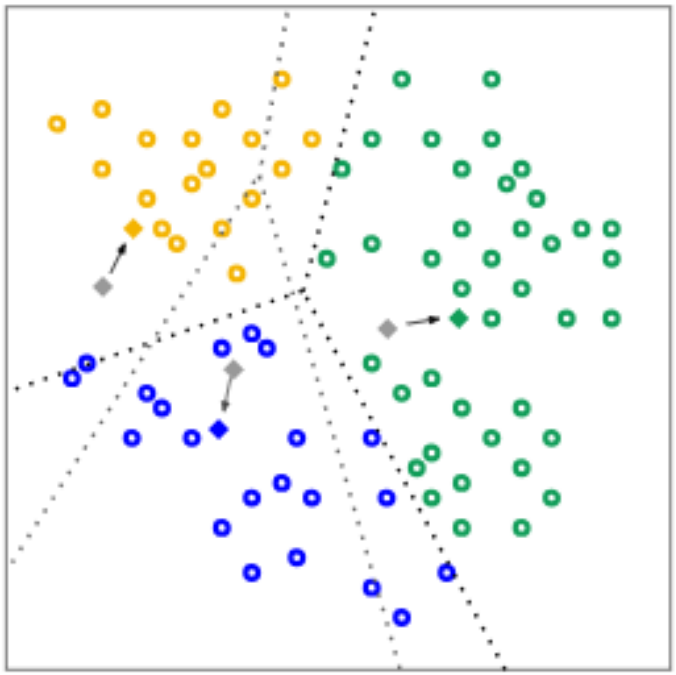


Recomputed centroids

Source: Machine Learning Crash Course, Google

K-Means Algorithm - Example

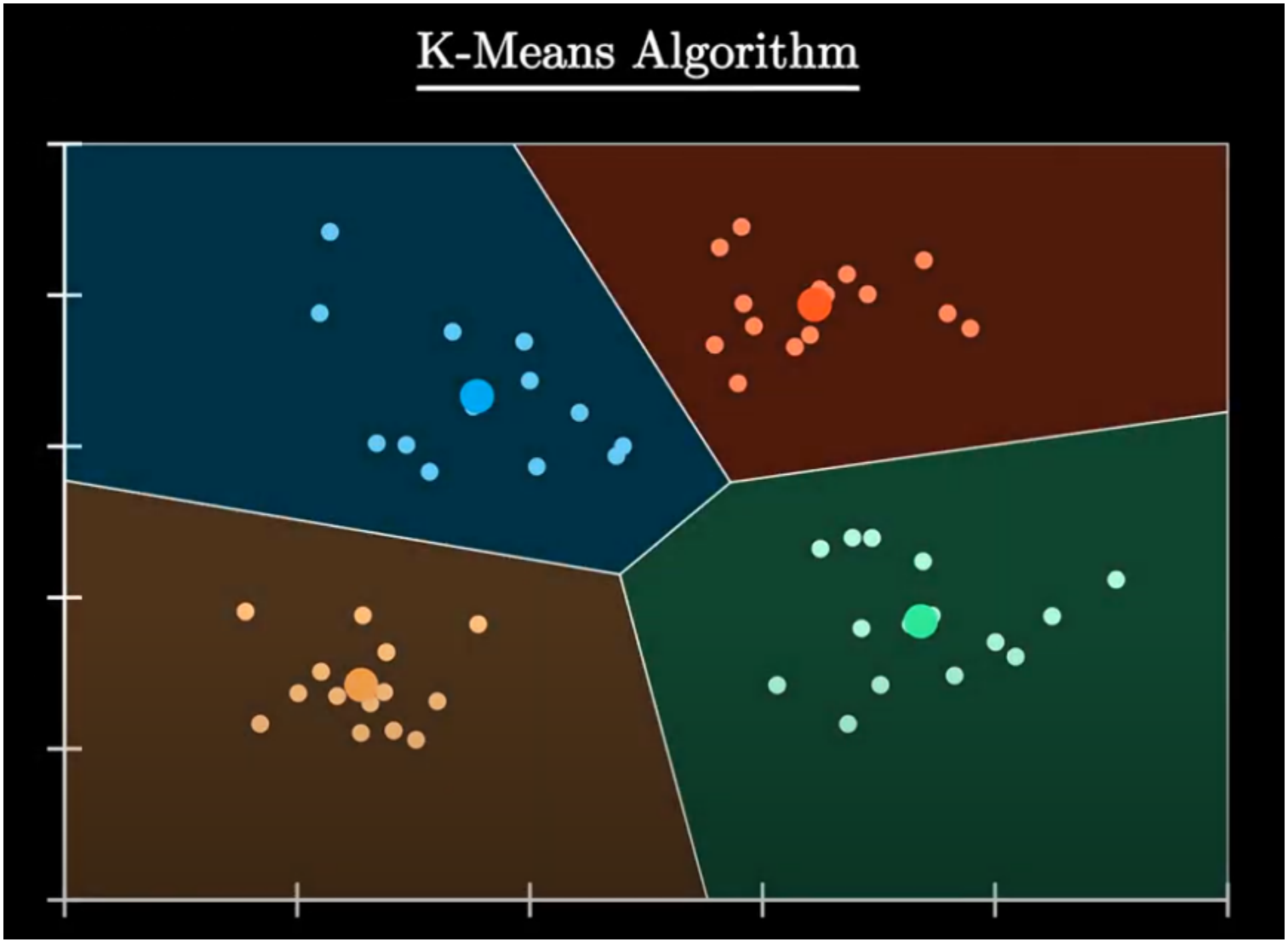
5. Reassign each point to the nearest new centroid.
6. Repeat steps 4 and 5, recalculating centroids and cluster membership, until points no longer change clusters.



Clusters after reassignment.

Source: Machine Learning Crash Course, Google

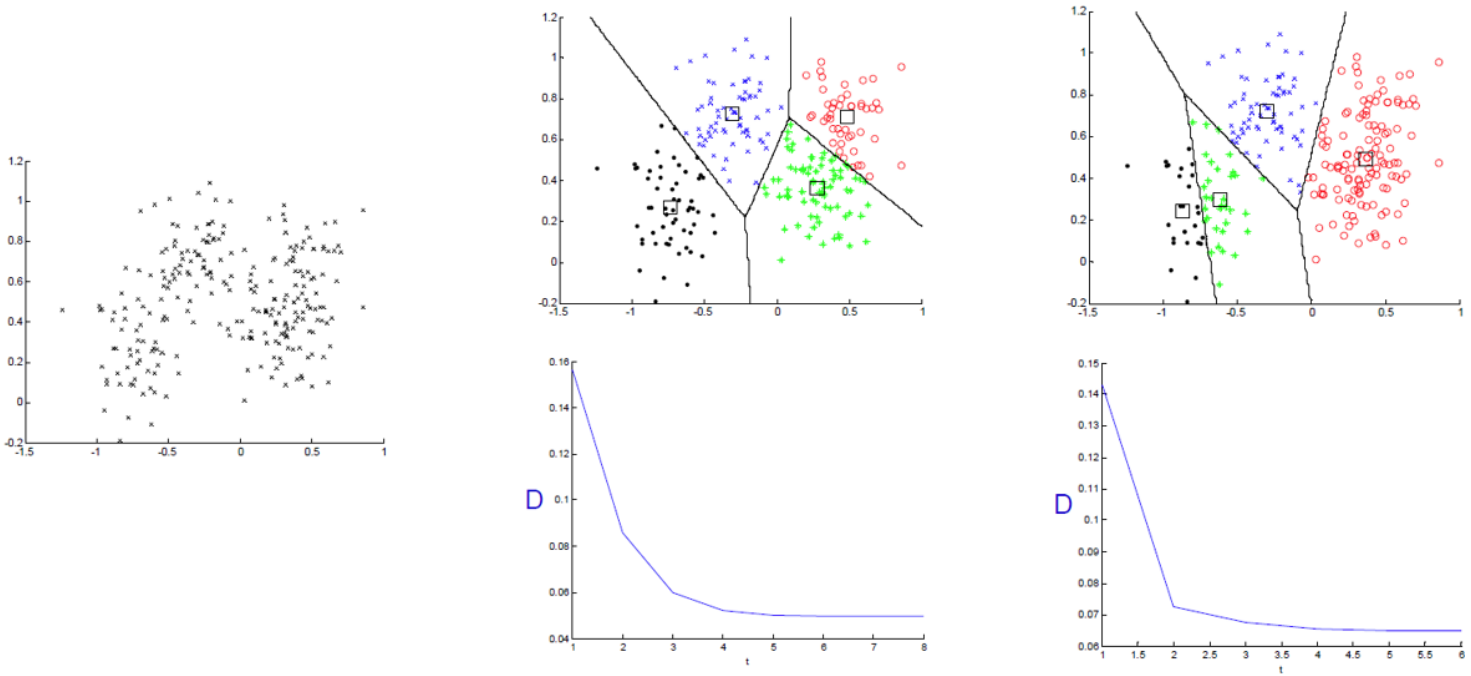
K-Means Algorithm Visualization



Video: K-Means Algorithm Visualization

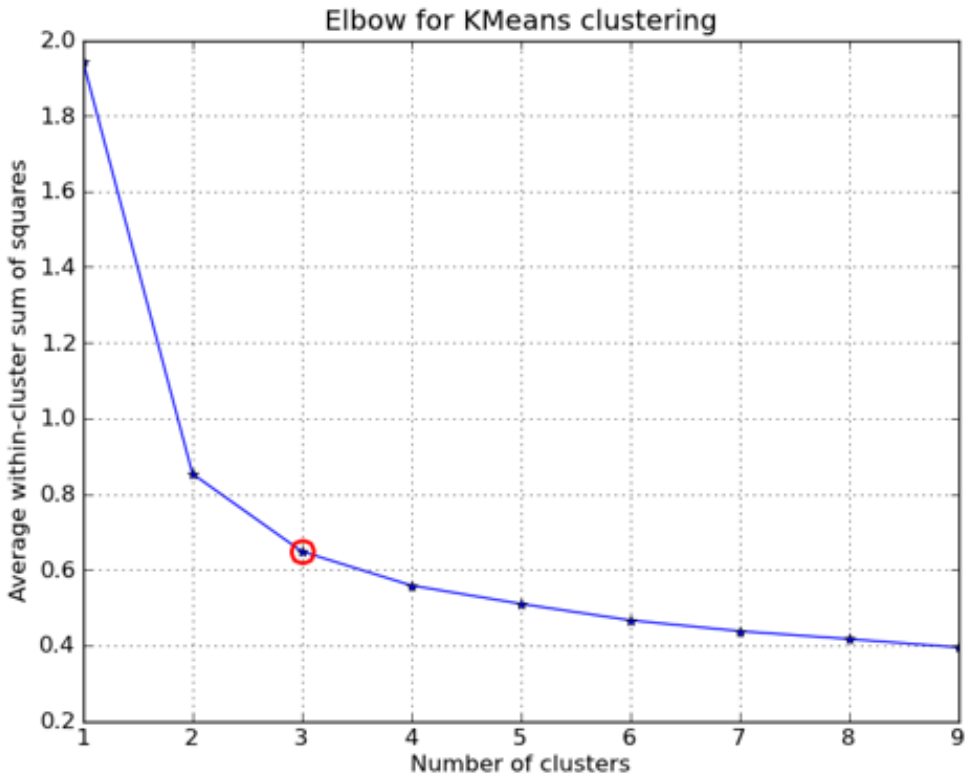
Limitations of K-Means

- Sensitive to initial centroid positions.
- Requires the number of clusters to be specified in advance.
- Struggles with non-spherical clusters and outliers.
- Not suitable for clusters of varying densities.



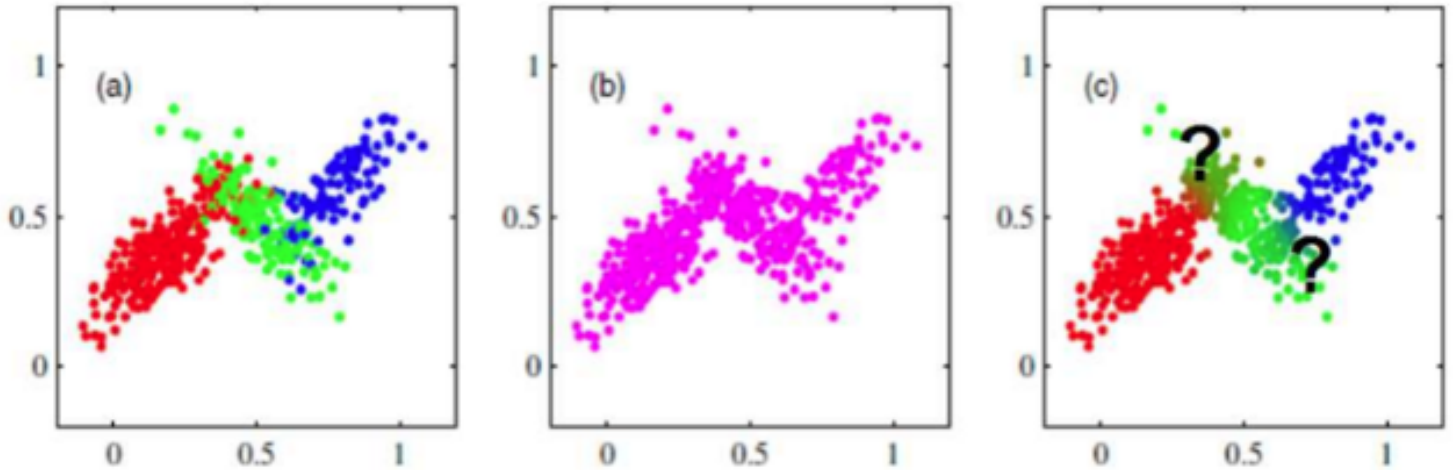
Choosing the Right Number of Clusters

- **Elbow Method**: Plot the sum of squared errors (SSE) vs. the number of clusters and look for the "elbow" point.
- **Silhouette Score**: Measures how similar a data point is to its own cluster compared to other clusters (range: -1 to 1).



Hard vs Soft Assignments

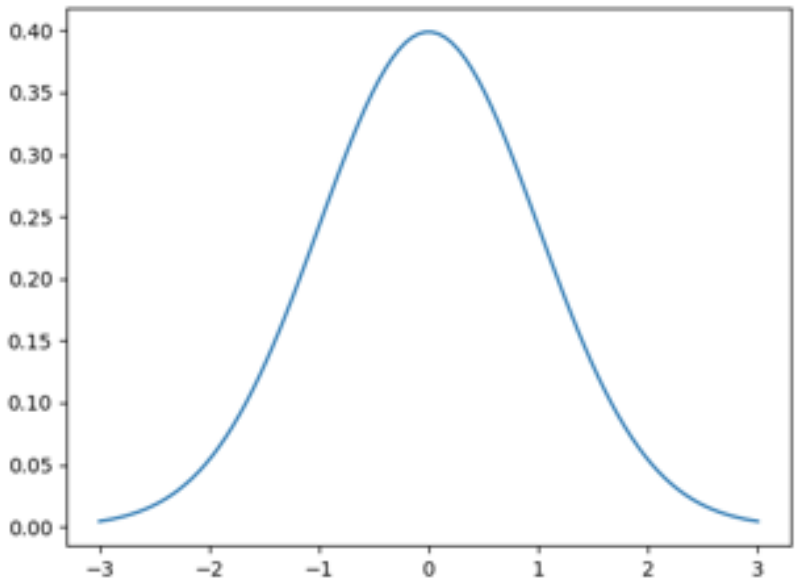
- In K-means, there is a hard assignment of feature vectors to a cluster.
- However, for feature vectors near the boundary this may be a poor representation.
- Instead, we can consider a soft-assignment, where the strength of the assignment depends on distance.
- To address these problems, we use soft clustering - Gaussian Mixture Model.
- Soft clustering is a form of clustering where observations may belong to multiple clusters.



Gaussian Distribution

- Gaussian / normal distribution / bell curve / probability density function (pdf). The function that describes the normal distribution is the following:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



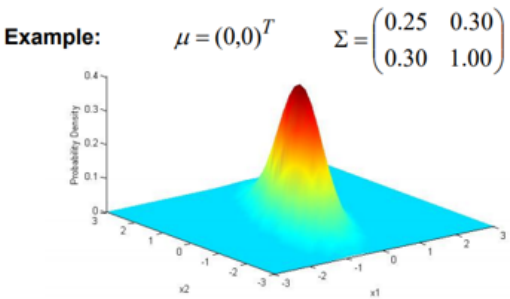
Multivariate Gaussians

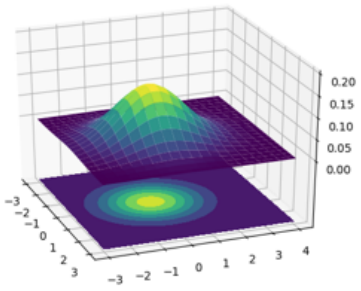
- For d dimensions, the Gaussian distribution of a vector $x = (x_1, x_2, \ldots, x_d)^T$ is defined by:

- **Formula**:

$$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

- **Parameters**:
  - μ: Mean vector of the Gaussian.
  - Σ: Covariance matrix of the Gaussian.





Introduction to Gaussian Mixture Model (GMM)

- GMM is a probabilistic model for clustering based on Gaussian distributions.
- Unlike K-Means, it assigns a probability to each data point belonging to every clusters.
- Uses the **Expectation-Maximization (EM)** algorithm for parameter estimation.
- **Key Concepts**:
  - Each cluster is represented by a Gaussian distribution.
  - The algorithm estimates the parameters of these distributions.

Expectation-Maximization (EM) Algorithm

Steps:

1. Initialize Gaussian parameters (means, variances, weights).
2. Expectation Step (E-Step): Compute probabilities for each data point.
3. Maximization Step (M-Step): Update parameters to maximize likelihood.
4. Repeat until convergence.

Expectation-Maximization (EM) Algorithm

- **Expectation Step**: Probabilistic assignment of data points to clusters.

- After initializing k random Gaussian models:
  - Calculate $E[z_{i,j}]$, the probability that x_i belongs to cluster j.
  - z_i: A vector of probabilities for x_i belonging to each cluster.

- **Formula**:

$$E[z_{i,j}] = \frac{P(x = x_i | \mu = \mu_j)}{\sum_{m=1}^{k} P(x = x_i | \mu = \mu_m)}$$

- **Gaussian Probability**:

$$P(x = x_i | \mu = \mu_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x_i - \mu_j)^2}{2\sigma_j^2}}$$

Expectation-Maximization (EM) Algorithm

- **Maximization Step**: Recalculate Gaussian models using weights from the Expectation Step.
- **Update Mean**:

$$\mu_j = \frac{\sum_i E[z_{i,j}]\, x_i}{\sum_i E[z_{i,j}]}$$

- **Update Covariance**:

$$\sigma_j = \frac{\sum_i E[z_{i,j}]\, (x_i - \mu_j)\, (x_i - \mu_k)}{\sum_i E[z_{i,j}]}$$

- **Key Idea**:
  - Use the probabilistic assignments $E[z_{i,j}]$ to update the parameters of the Gaussian models.
  - Iteratively refine the model to better fit the data.

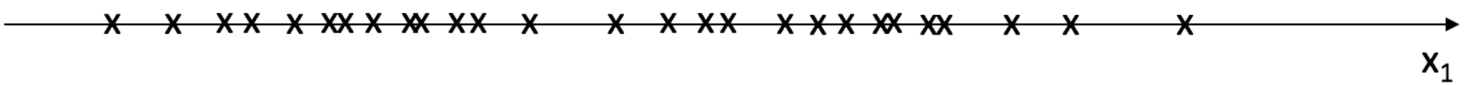
Expectation-Maximization (EM) Algorithm

Defining a stopping criterion:

- With k-means clustering, we iteratively recalculated means and reassigned observations until convergence, where observations stopped moving between clusters.
- However, since we're now dealing with soft clustering that involves continuous probabilities, we can't rely on this same type of convergence.
- Instead, we'll set a stopping criterion to end the iterative cycle. The cycle will stop once the observation probabilities stop changing by more than some threshold.

GMM - Example

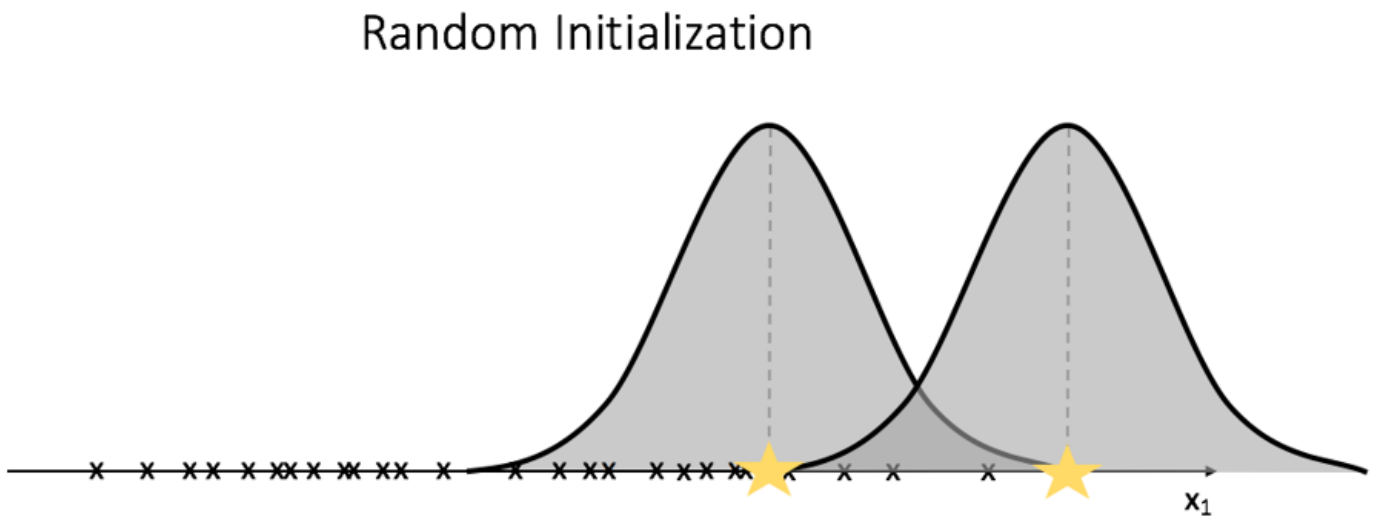
- Suppose we have data for a set of observations with one feature, x_1.
- We can use this data to build a Gaussian model for each class. This would allow us to calculate the probability of a new observation belonging to each class.



Source: Jeremy Jordan

GMM - Example

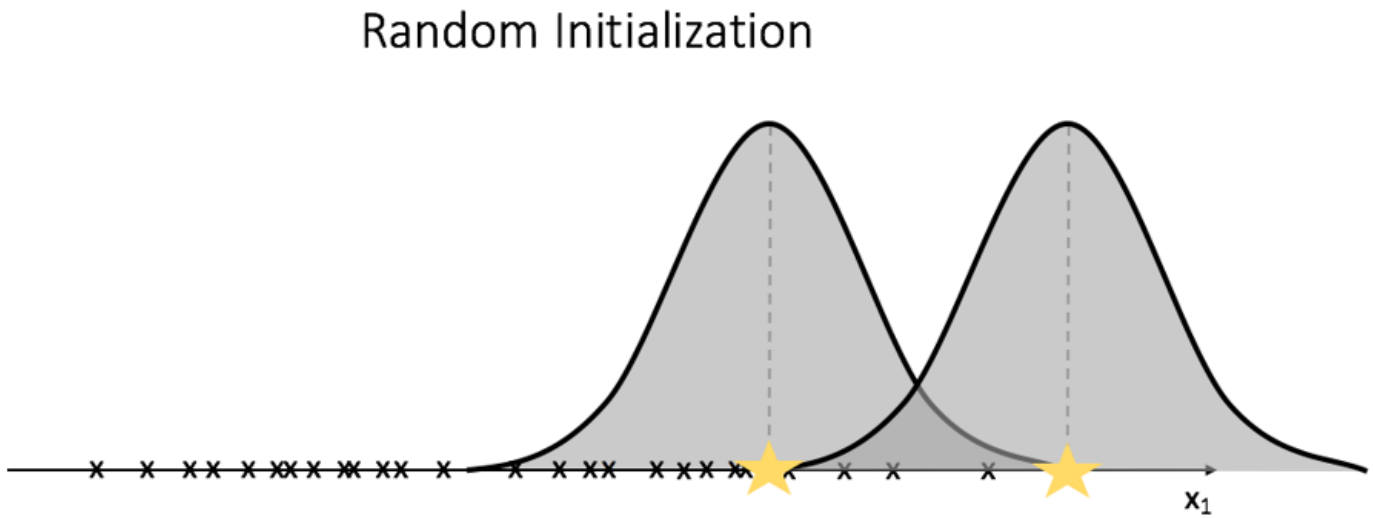
- Since we don't know which class each observation belongs to, we don't have an easy way to build multiple Gaussian models to partition the data.
- Let's start with a random initialization of our Gaussian models and then iteratively optimize the attributes.



Source: Jeremy Jordan

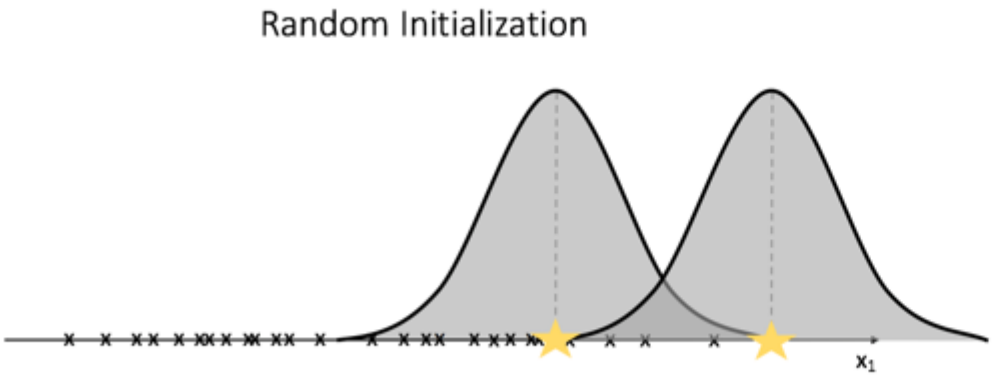
GMM - Example

- After initializing two random Gaussian distributions, we will compute the likelihood of each observation under both Gaussian models. This likelihood is calculated using the probability density function of the Gaussian distribution.
- We will then recalculate the parameters of the Gaussian models. When calculating the mean and variance for each Gaussian, we will use all of the observations. However, each observation will be weighted by the likelihood of it belonging to the given model.



GMM - Example

- Rather than assigning observations to a class, we'll assume there's a possibility that it can belong to any of the k clusters.
- Thus, for any given observation we'll calculate the probability of it belonging to each of the k clusters. Let's attach a hidden variable to each observation which contains a vector, z_i, to store these probabilities.

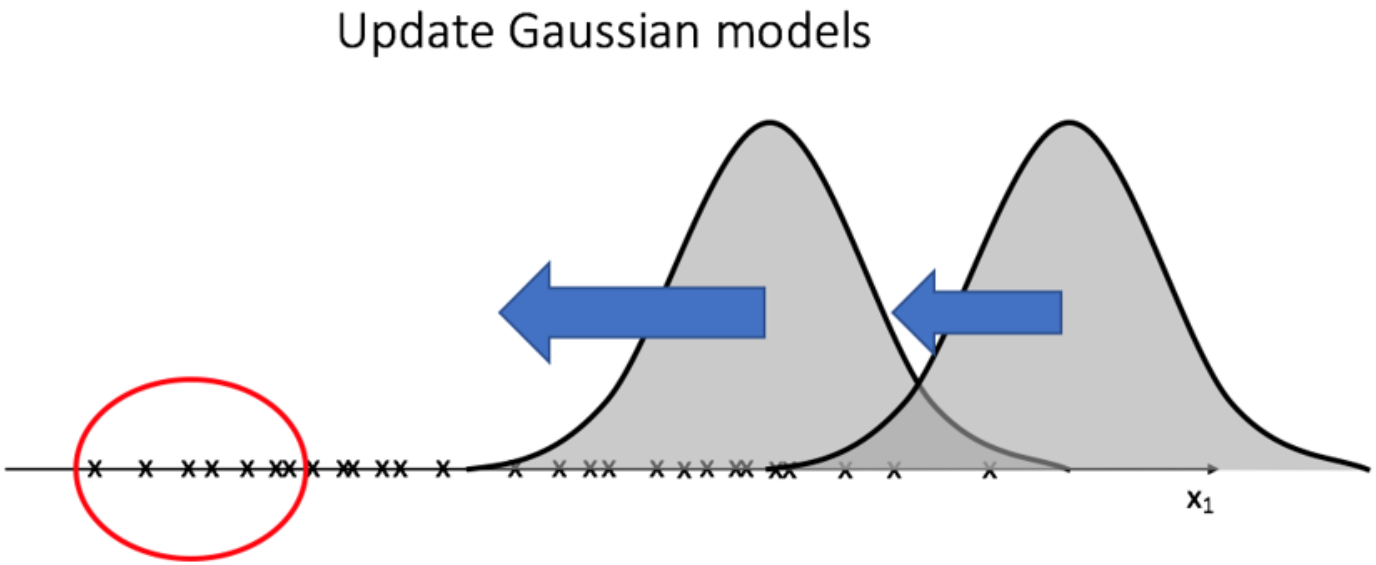


$$z_i = \begin{bmatrix} P\left(x_i \text{ belongs to Cluster } 1\right) \\ P\left(x_i \text{ belongs to Cluster } 2\right) \\ \vdots \\ P\left(x_i \text{ belongs to Cluster } k\right) \end{bmatrix}$$

Source: Jeremy Jordan

GMM - Example

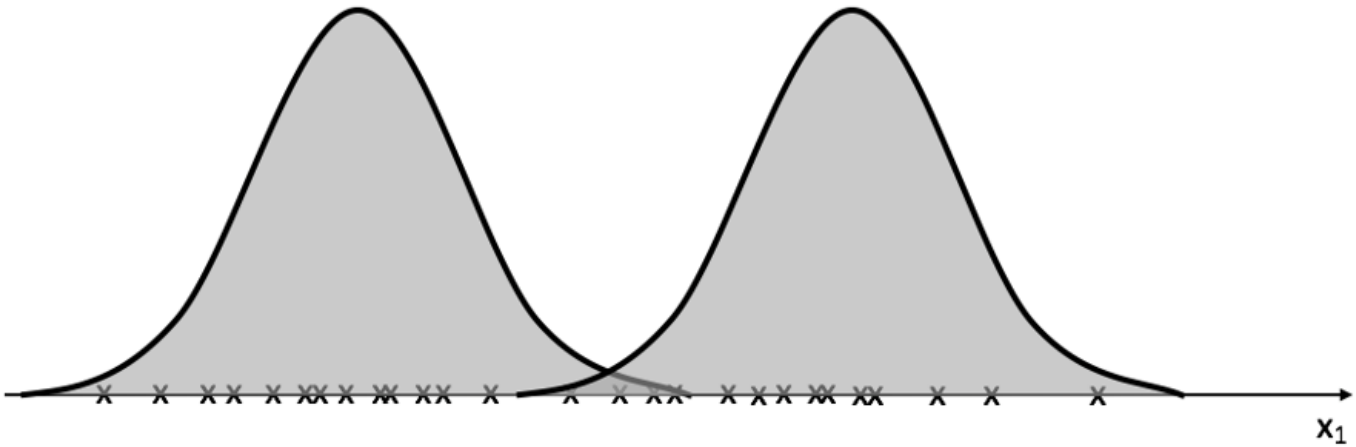
- After using z_i to calculate the weighted parameters (mean and variance) for each cluster, we update the Gaussian models.
- You'll see that the leftmost data (circled below) will pull both models in it's direction, but it will have a larger effect on the left Gaussian since the observations will have a higher weight for the model that it's closest to.



Source: Jeremy Jordan

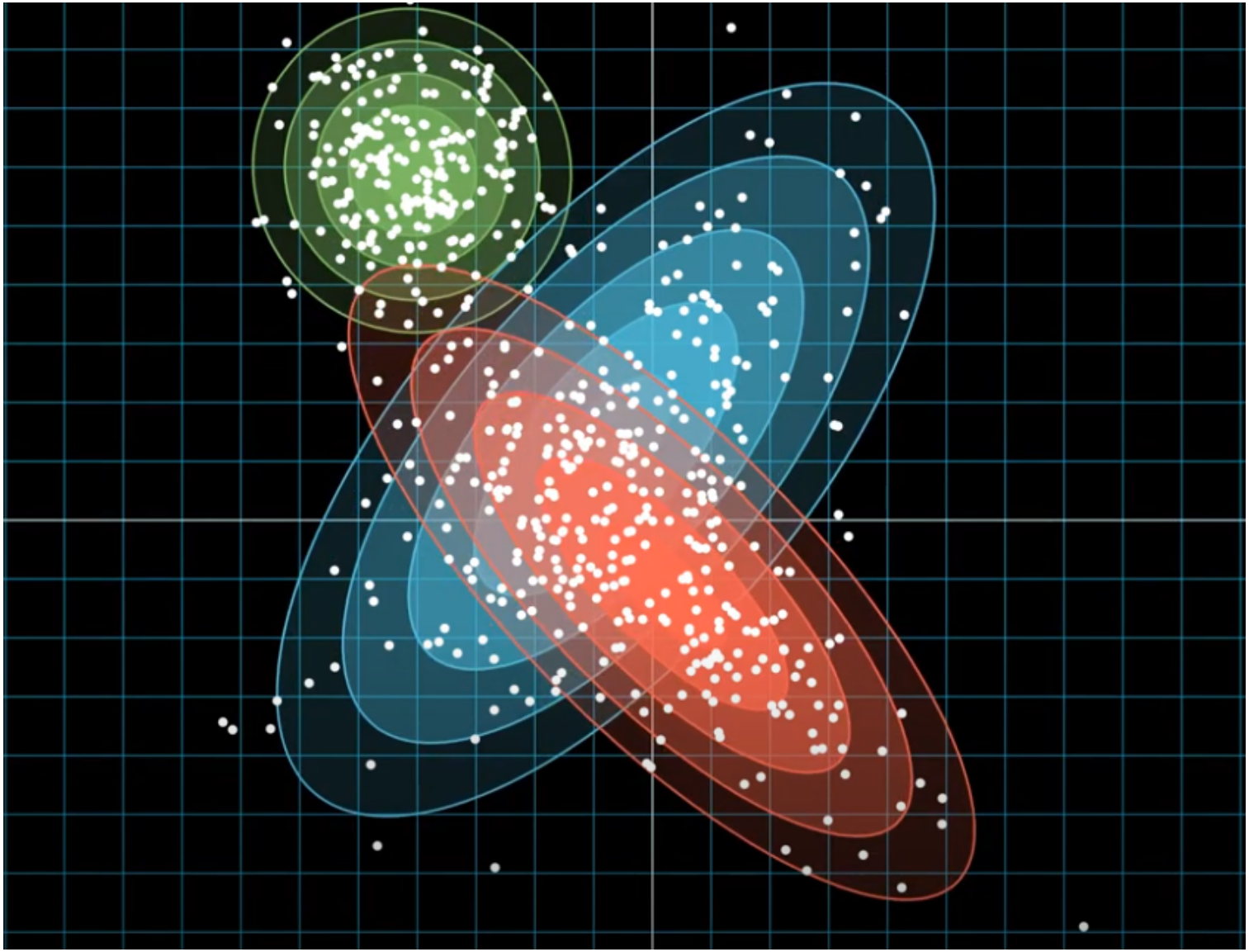
GMM - Example

- We'll continue this cycle of recalculating z_i and then using it to update the Gaussians until we "converge" on optimal clusters.
- The example visualizes a univariate Gaussian example, we can extend this logic to accommodate multivariate datasets.



Source: Jeremy Jordan

GMM Visualization



Video: GMM Visualization

GMM vs K-Means

- **K-Means**:
  - Hard assignment (each point belongs to one cluster).
  - Assumes spherical clusters of equal size.

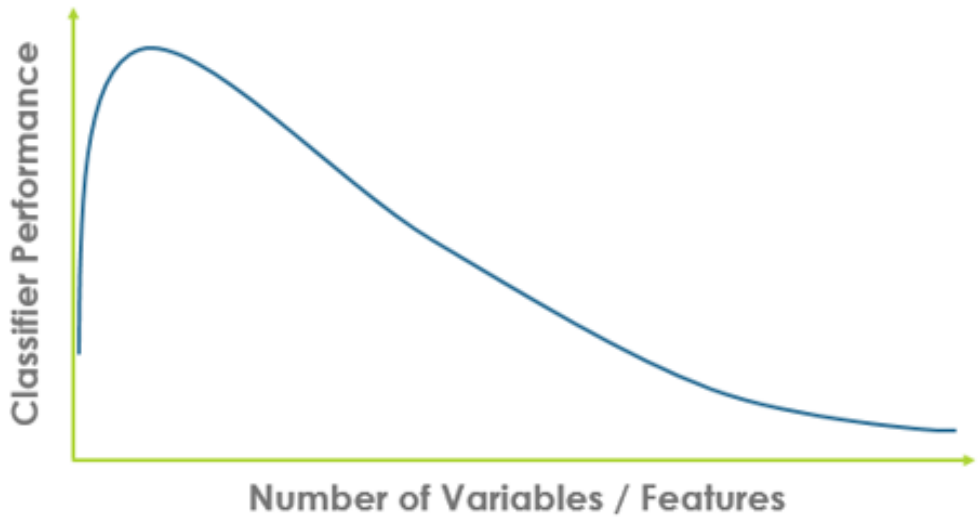
- **GMM**:
  - Soft assignment (probabilistic membership in clusters).
  - Can model clusters of different shapes and sizes.

Dimensionality Reduction

Curse of Dimensionality

- **Explanation**: As the number of dimensions increases, data becomes sparse, and distance metrics lose meaning.

- **Examples**:
  - High computational cost.
  - Overfitting in machine learning models.
  - Difficulty in visualizing and interpreting data.



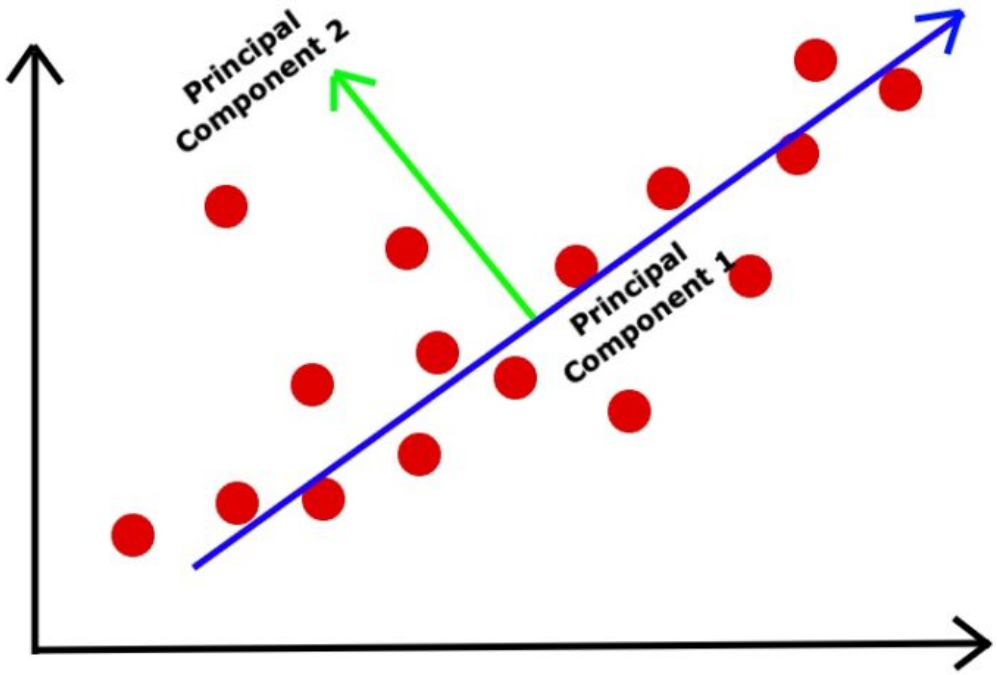
What is Dimensionality Reduction?

- **Definition**: A technique to reduce the number of features (dimensions) in a dataset while preserving its structure.

- **Motivation**:
  - Improves computational efficiency.
  - Reduces noise and redundancy.
  - Enables visualization of high-dimensional data.

Principal Component Analysis (PCA)

- **Theory**: A linear dimensionality reduction technique that projects data onto orthogonal axes (principal components).



Source: INFOARYAN

PCA Algorithm: Steps 1–2

Given a dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ where each $\mathbf{x}_i \in \mathbb{R}^d$, the goal of PCA is to find a lower-dimensional representation of the data while preserving as much variance as possible.

1. **Standardize the Data**
   - Compute the mean of the dataset:

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

   - Center the data:

$$\mathbf{X}_{centered} = \mathbf{X} - \bar{\mathbf{x}}$$

2. **Compute the Covariance Matrix**
   - Covariance matrix captures the variance and correlation between features:

$$\mathbf{C} = \frac{1}{N-1} \mathbf{X}_{centered}^{\top} \mathbf{X}_{centered}$$

PCA Algorithm: Steps 3–4

3. **Compute the Eigenvalues and Eigenvectors**
   - Solve the eigenvalue problem:

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

   - λ_i is the eigenvalue, $\mathbf{v}_i$ is the eigenvector.

4. **Sort Eigenvectors by Eigenvalues**
   - Order eigenvectors by decreasing eigenvalues:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$$

   - The eigenvectors corresponding to the largest eigenvalues capture the most variance.

PCA Algorithm: Steps 5–6

5. **Select the Top K Eigenvectors**
   - Form a projection matrix $\mathbf{W}$:

$$\mathbf{W} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K]$$

   - K is the number of principal components you choose to keep.

6. **Project the Data onto the New Subspace**
   - Transform the data into the new K-dimensional subspace:

$$\mathbf{Z} = \mathbf{X}_{centered}\mathbf{W}$$

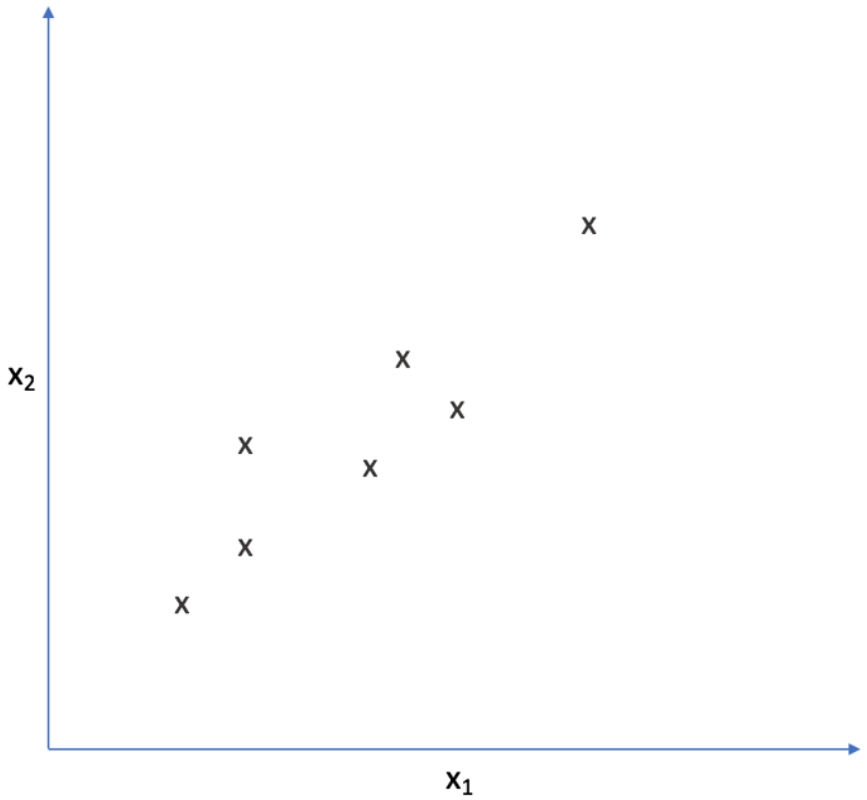
   - $\mathbf{Z}$ is the reduced representation of the original data.

PCA Algorithm Summary

1. Standardize the data
2. Compute the covariance matrix
3. Calculate eigenvalues and eigenvectors
4. Sort eigenvectors by eigenvalues
5. Select top K eigenvectors
6. Project data onto new subspace

Principal Components Analysis (PCA)

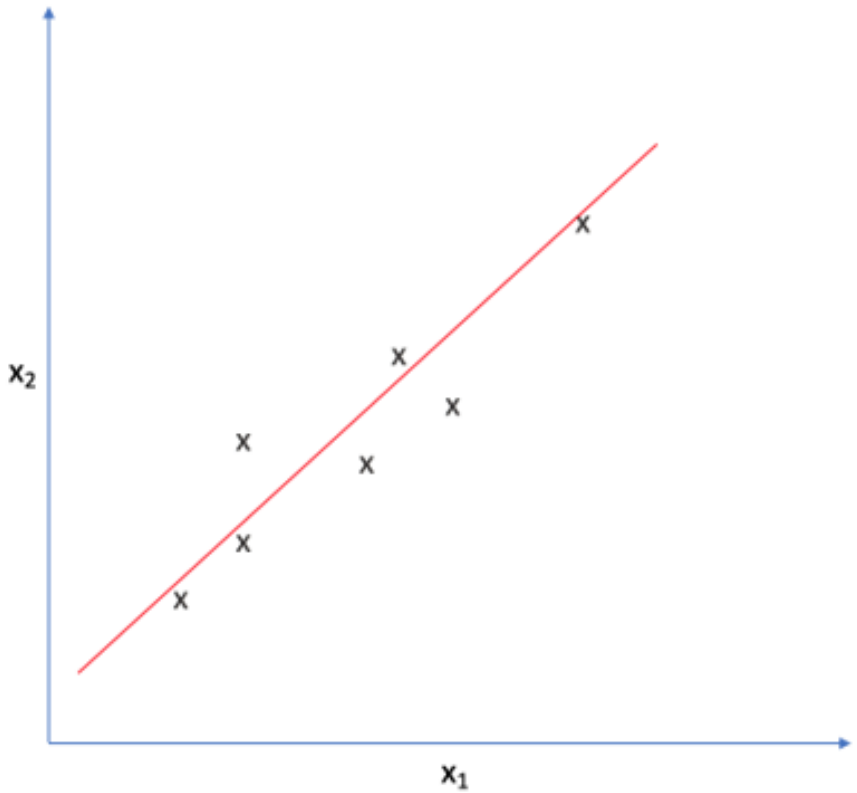
- Intuition: Let's look at the following two-dimensional feature-space.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

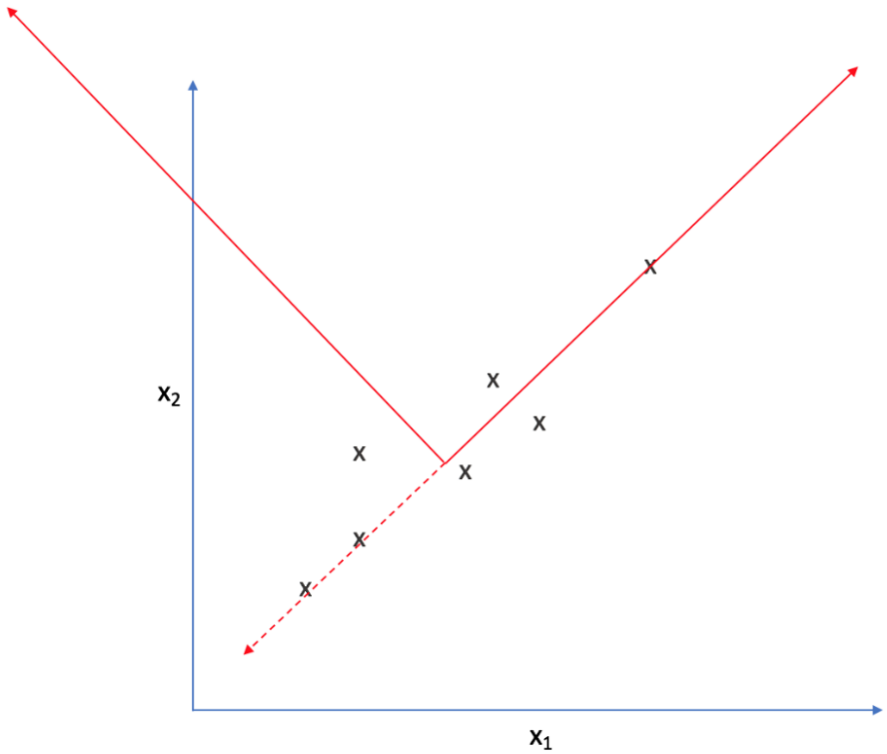
- It appears that most of the points on this scatterplot lie along the following line. This suggests some degree of correlation between x_1 and x_2.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

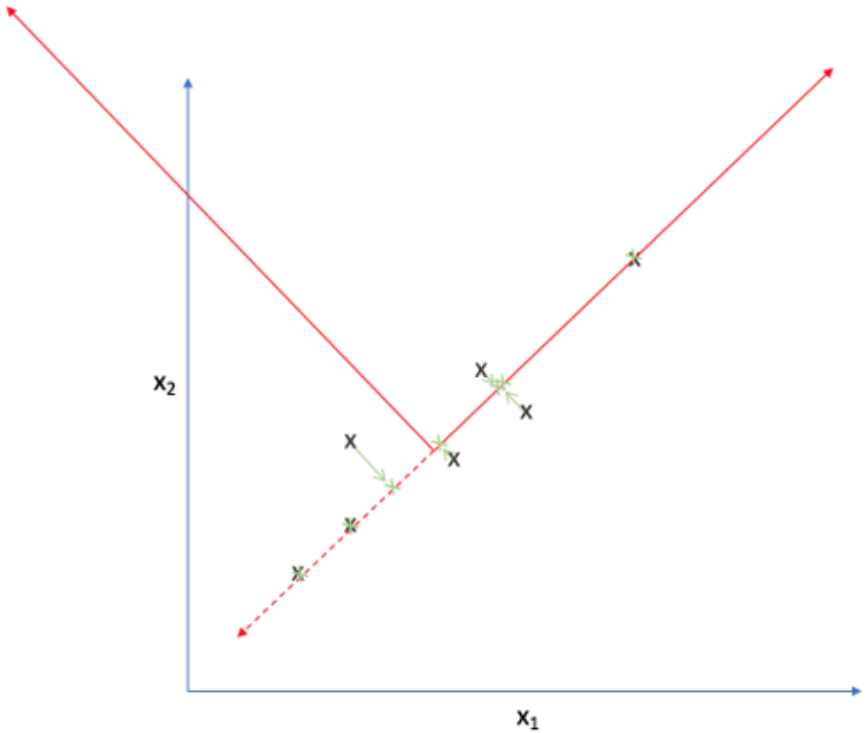
- As such, we could reorient the axes to be centered on the data and parallel to the line above.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

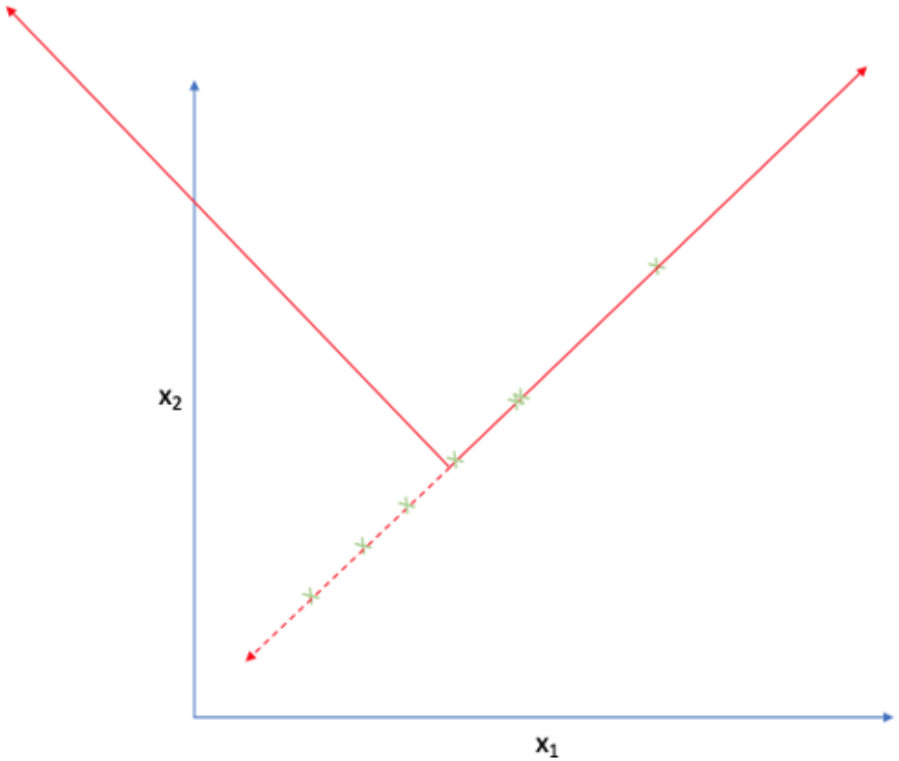
- Strictly speaking, we're still dealing with two dimensions. However, let's project each observation onto the primary axis.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

- Now, every observation lies on the primary axis.



Source: Jeremy Jordan

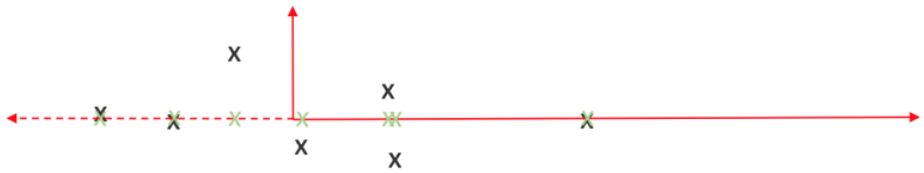
Principal Components Analysis (PCA)

- We just compressed a two dimensional dataset into one dimension by translating and rotating our axes! After this transformation, we only really have one relevant dimension and thus we can discard the second axis.

Source: Jeremy Jordan

Principal Components Analysis (PCA)

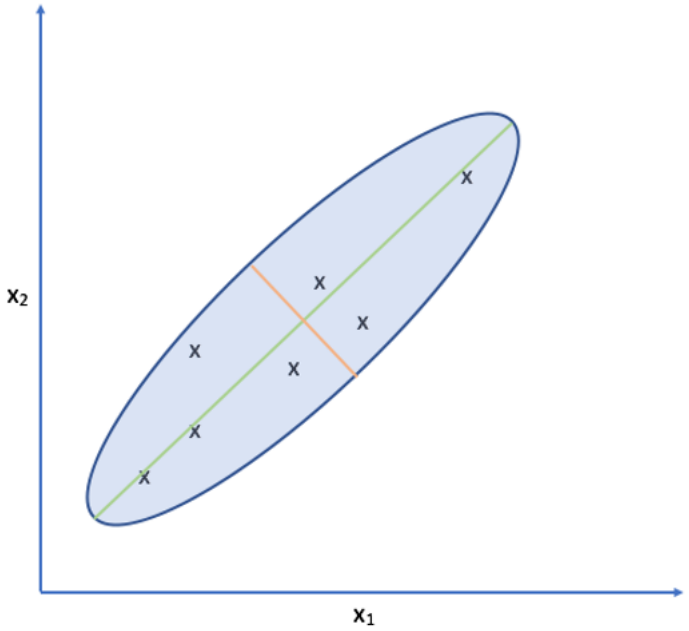
- Comparing the original observations with our new projections, we can see that it's not an exact representation of our data. However, one could argue it does capture the essence of our data - not exact, but enough to be meaningful.



Source: Jeremy Jordan

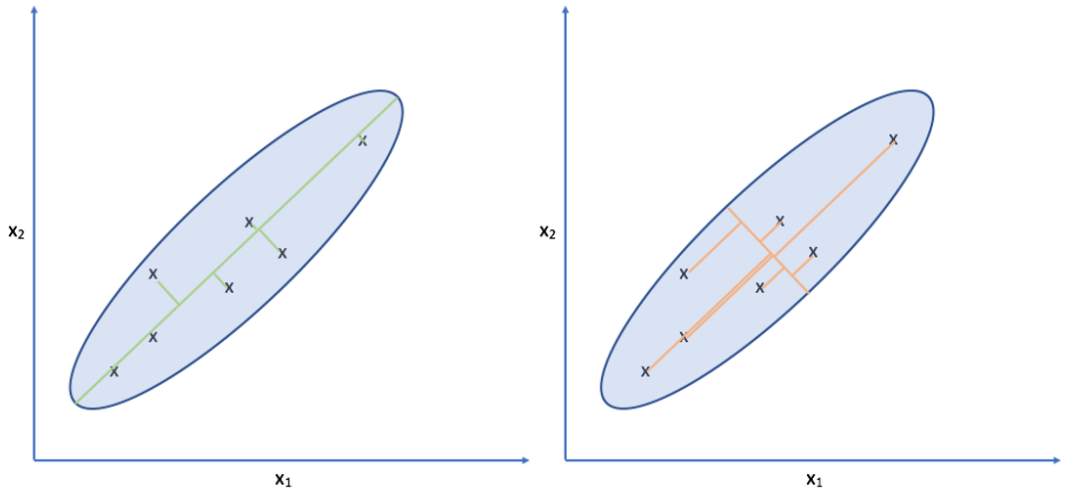
Principal Components Analysis (PCA)

- We can view the cumulative distance between our observations and projected points as a measure of information loss. Thus we'd like to orient the axes in a manner which minimizes this. How do we do this? That's where variance comes into play.
- Another example: look at the spread of the data along the green and orange directions. Notice that there's a much more deviation along the green direction than there is along the orange direction.



Principal Components Analysis (PCA)

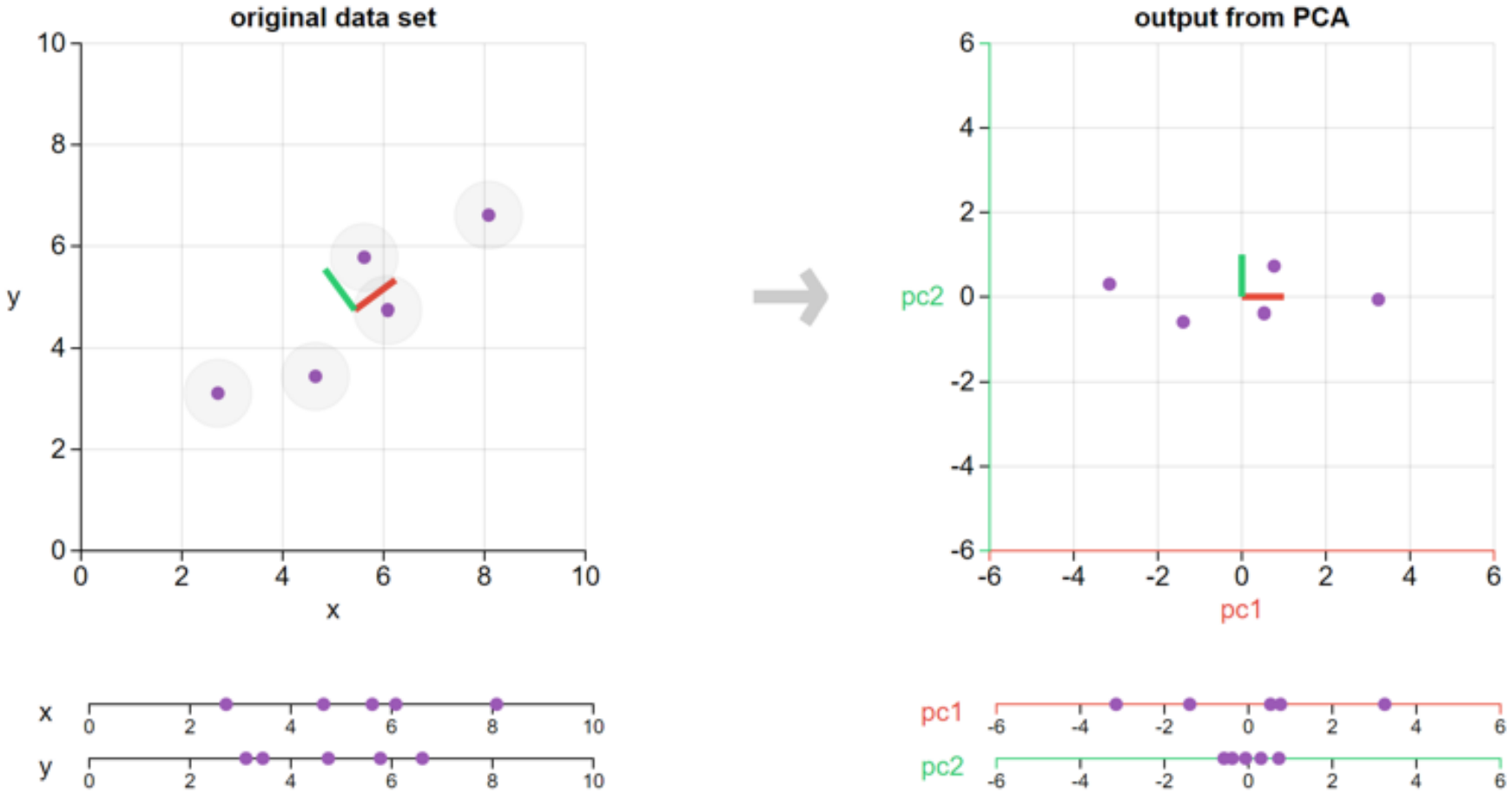
- Projecting our observations onto the orange vector requires us to move much further than we would need to for projecting onto the green vector. It turns out, the vector which is capable of capturing the maximum variance of the data minimizes the distance required to move our observations as projection onto the vector.



Source: Jeremy Jordan

Principal Components Analysis (PCA)

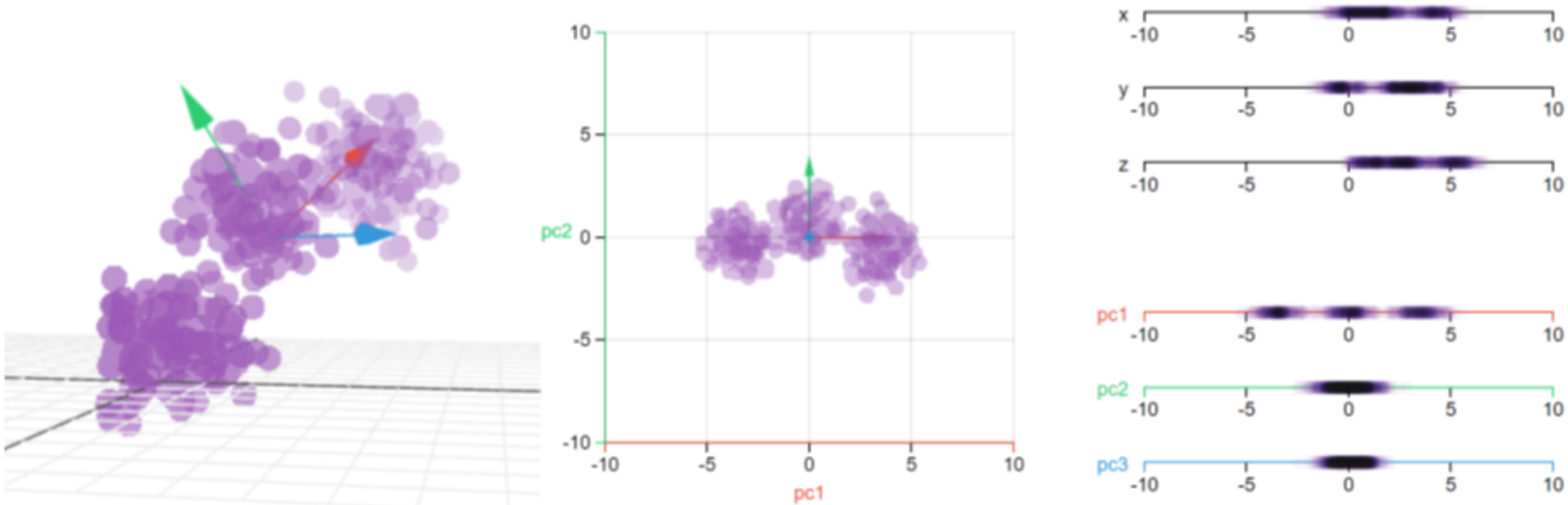
2D example



Source: Principal Component Analysis Explained Visually

Principal Components Analysis (PCA)

3D example

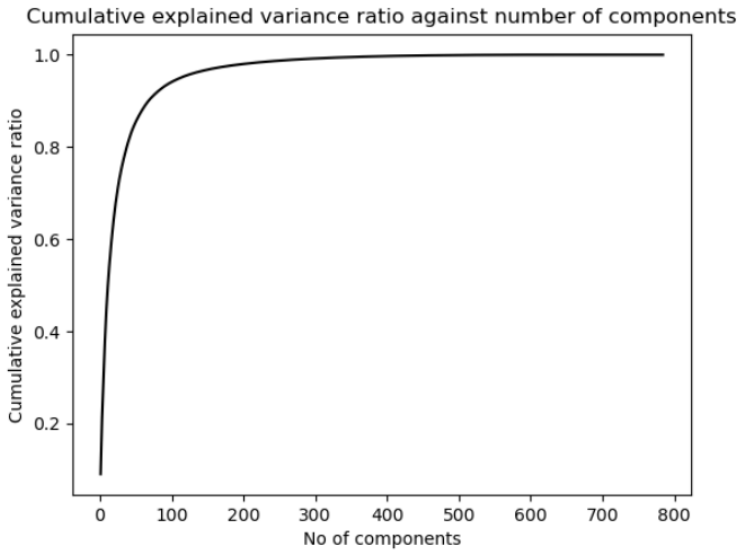


Source: Principal Component Analysis Explained Visually

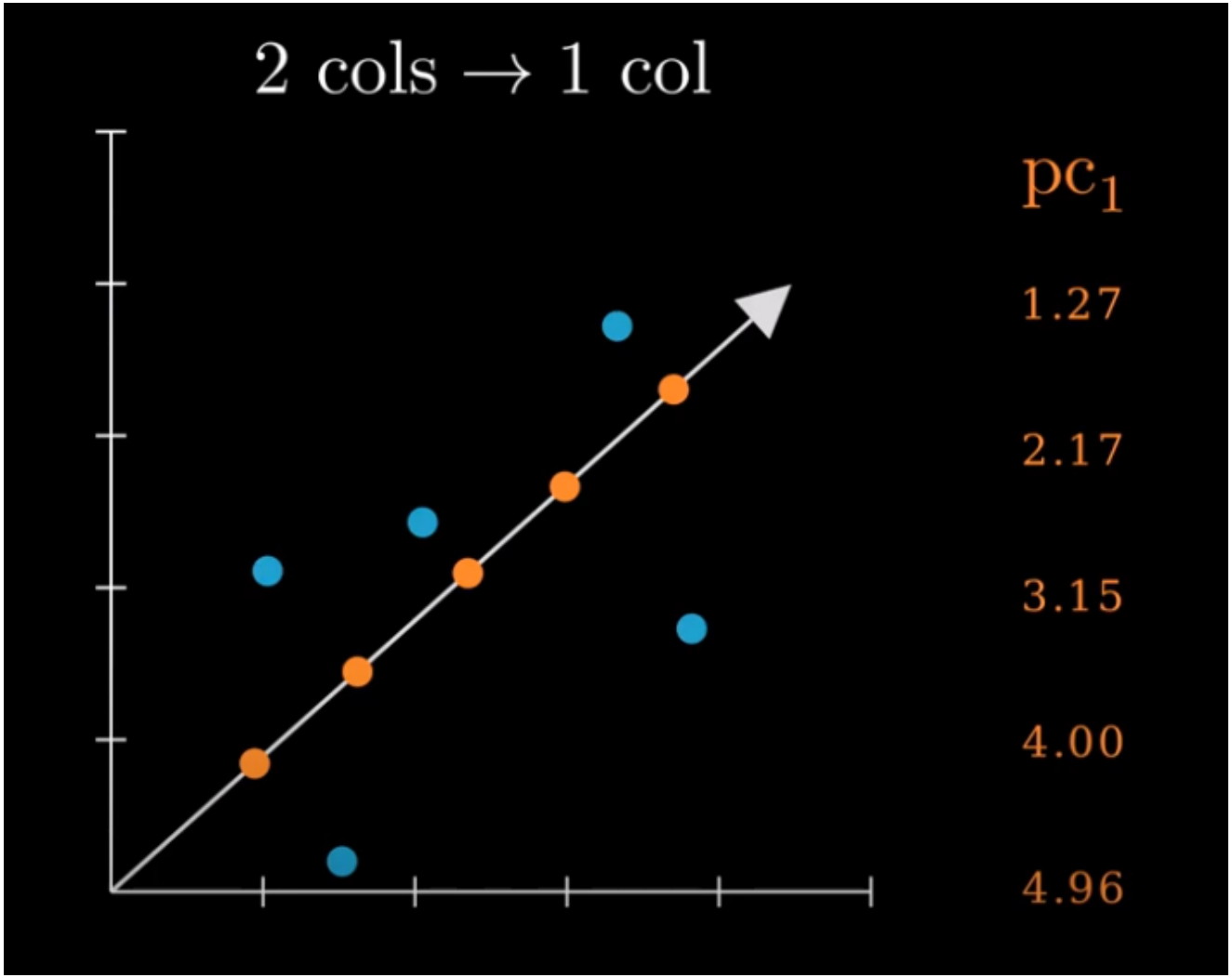
Explained Variance Ratio

- **Definition**: The proportion of variance explained by each principal component. It determines how much information is retained in each principal component and helps in choosing the optimal number of components.

- **How to Choose**:
  - Plot the cumulative explained variance ratio.
  - Select the number of components that explain a significant portion of the variance (e.g., 95%).



PCA Visualization



Video: PCA Visualization

Limitations of PCA

- **Linearity Assumption**: PCA assumes linear relationships between variables.
- **Sensitivity to Scaling**: Requires standardized data.
- **Loss of Information**: May discard important variance in lower components.