Implementing Clustering and Dimensionality Reduction in Spark ML

Janani Ravi

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Unsupervised learning is used to find patterns within the data itself

K-means clustering is a popular technique to find logical groupings of data

Elbow and silhouette methods are used to find the best value of hyperparameter k

Dimensionality reduction is used to discover latent factors in underlying data

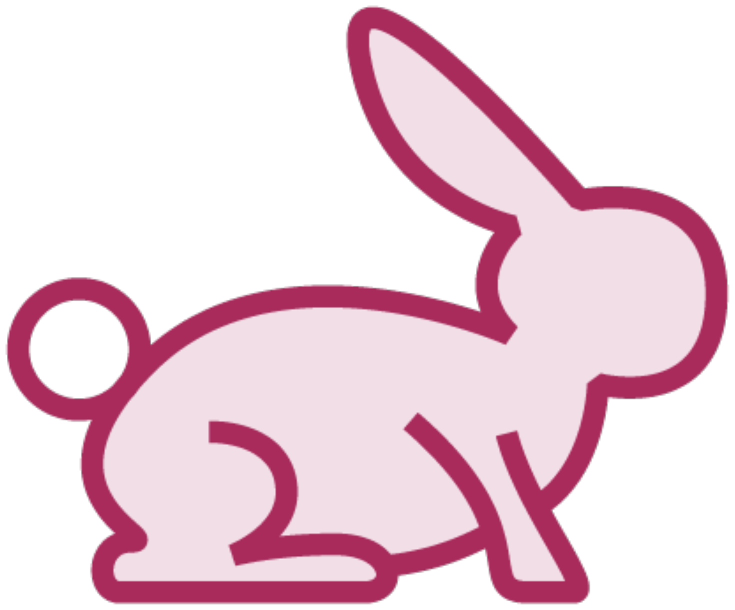
PCA is very commonly used method for dimensionality reduction

Supervised and Unsupervised Learning

“What lies behind us and what lies ahead of us are tiny matters compared to what lives within us”

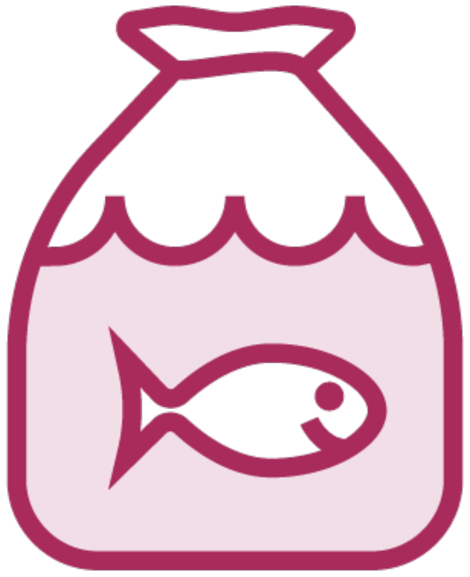
Henry David Thoreau

Whales: Fish or Mammals?



Mammals

Members of the infraorder *Cetacea*



Fish

Look like fish, swim like fish, move with fish

Whales: Fish or Mammals?

ML-based Classifier

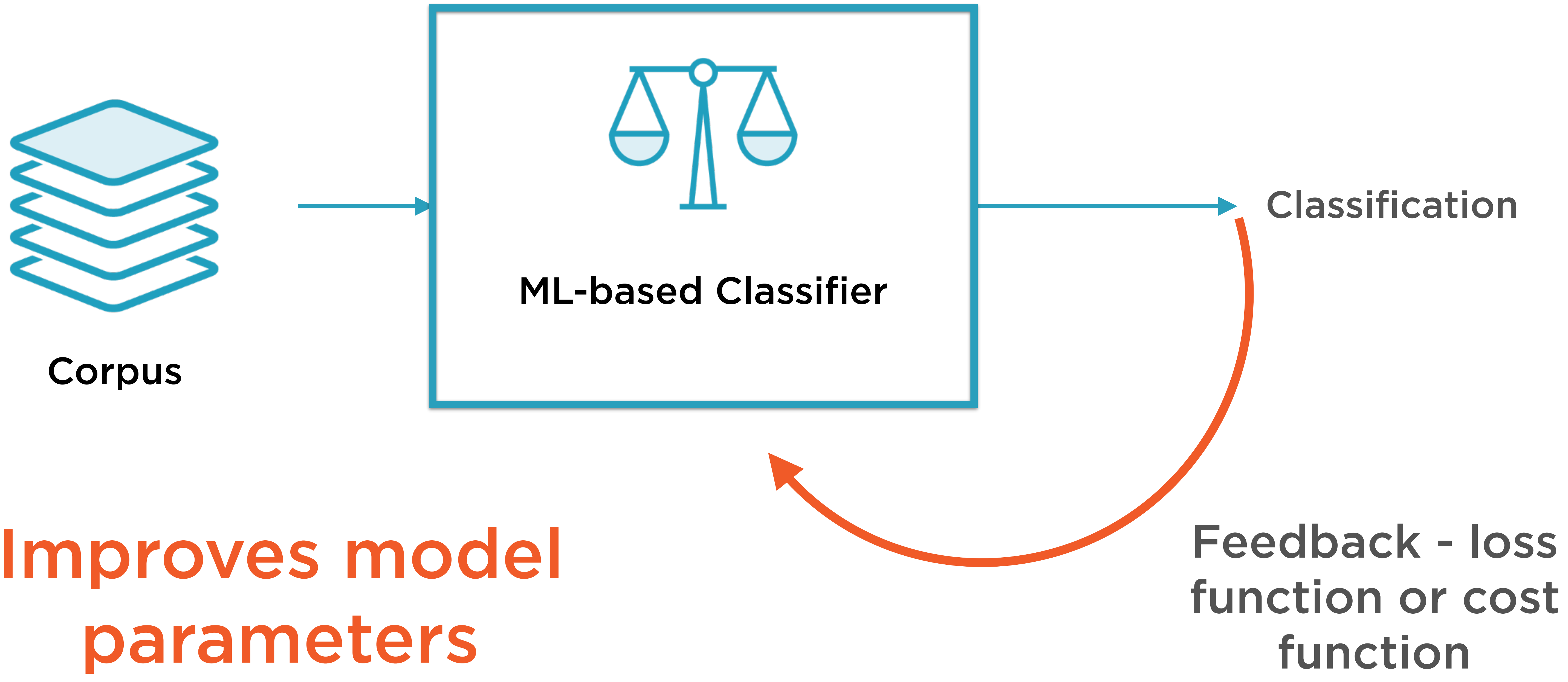
Training

Feed in a large corpus of data classified correctly

Prediction

Use it to classify new instances which it has not seen before

Training the ML-based Classifier



$$y = f(x)$$

Supervised Machine Learning

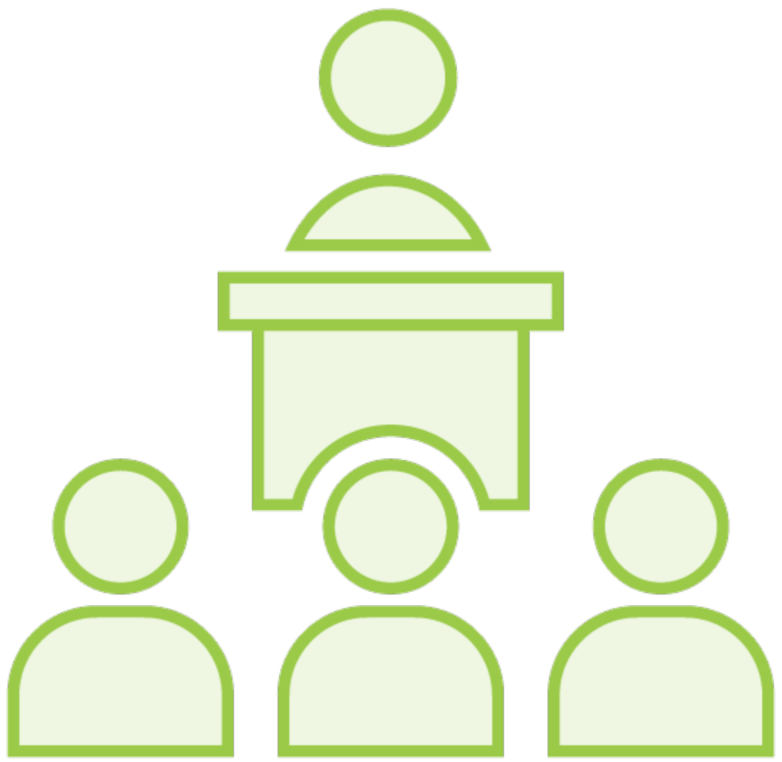
Most machine learning algorithms seek to "learn" the function f that links the features and the labels

Everything so far discussed really applied only to **Supervised Learning**

Unsupervised Learning does not have:

- y variables
- a labeled corpus

Types of ML Algorithms

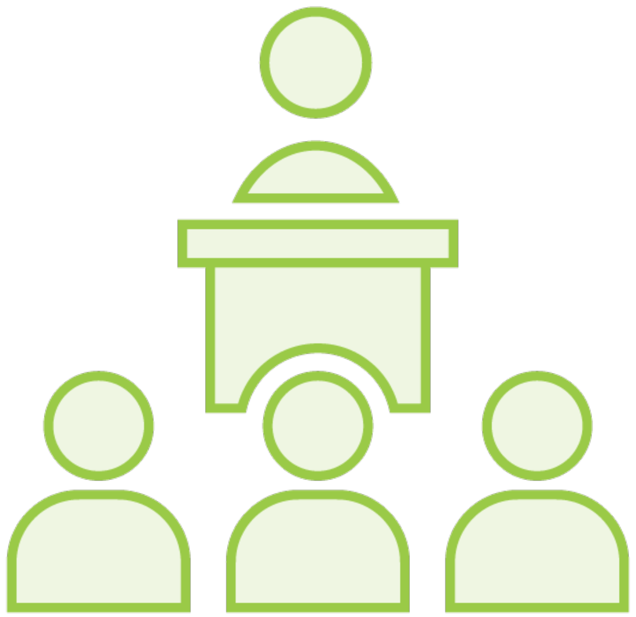


Supervised

Labels associated with the training data is used to correct the algorithm

Unsupervised

The model has to be set up right to learn structure in the data



Supervised Learning

Input variable x and output variable y

Learn the mapping function `y = f(x)`

Approximate the mapping function so for new values of x we can predict y

Use existing dataset to correct our mapping function approximation

Unsupervised Learning

Only have input data x – no output data

Model the underlying structure to learn more about data

Algorithms self discover the patterns and structure in the data

Unsupervised Learning Use-cases

ML Technique	Use-case
To make unlabelled data self-sufficient	Identify photos of a specific individual
Latent factor analysis	Find common drivers of 200 stocks
Clustering	Find relevant document in a corpus
Anomaly detection	Flag fraudulent credit card transactions
Quantisation	Compress true color (24 bit) to 8 bit
Pre-training for supervised learning problems (classification, regression)	All of the above!

Unsupervised ML Algorithms

Clustering

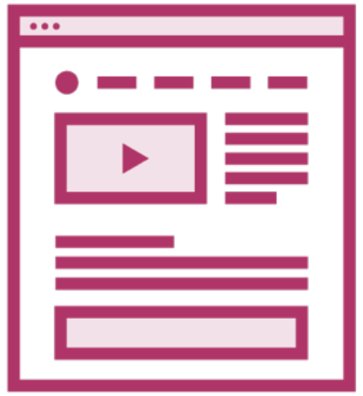
Identify patterns in data items e.g. K-means clustering

Dimensionality reduction

Identify latent factors that drive data e.g. PCA

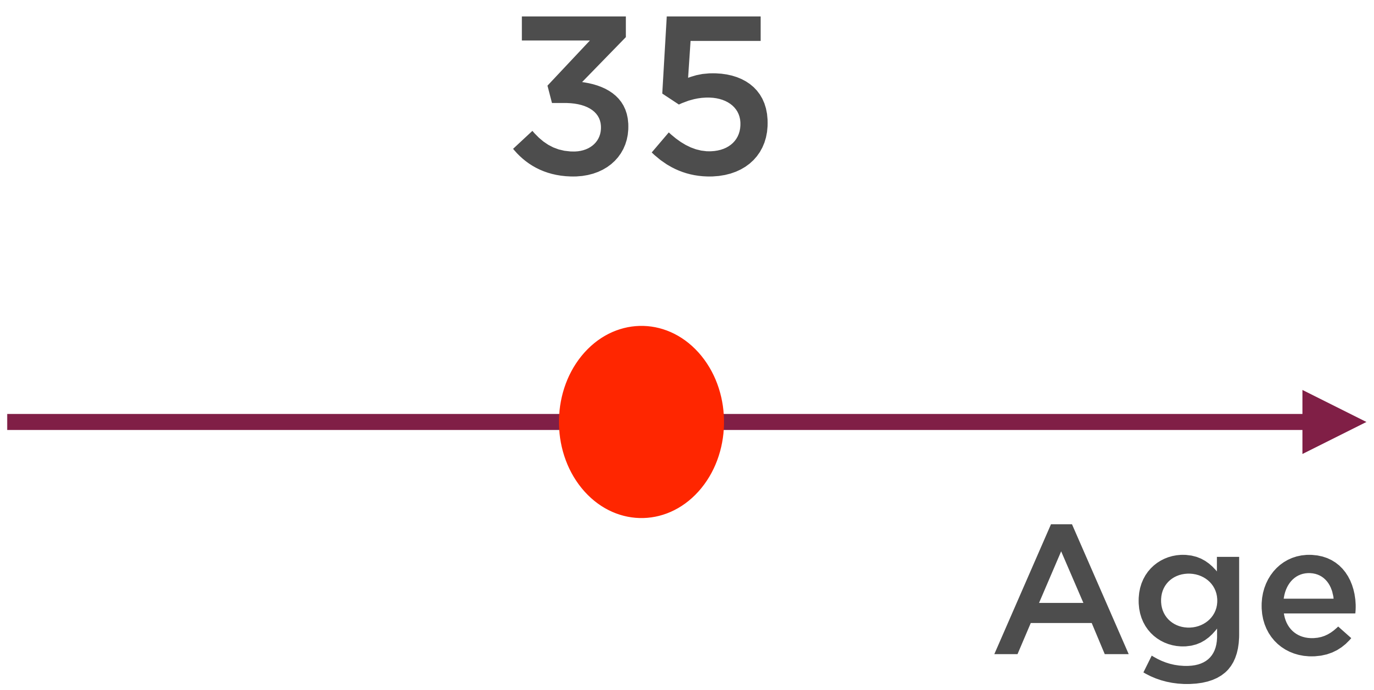
Intuitively Understanding K-Means Clustering

Clustering

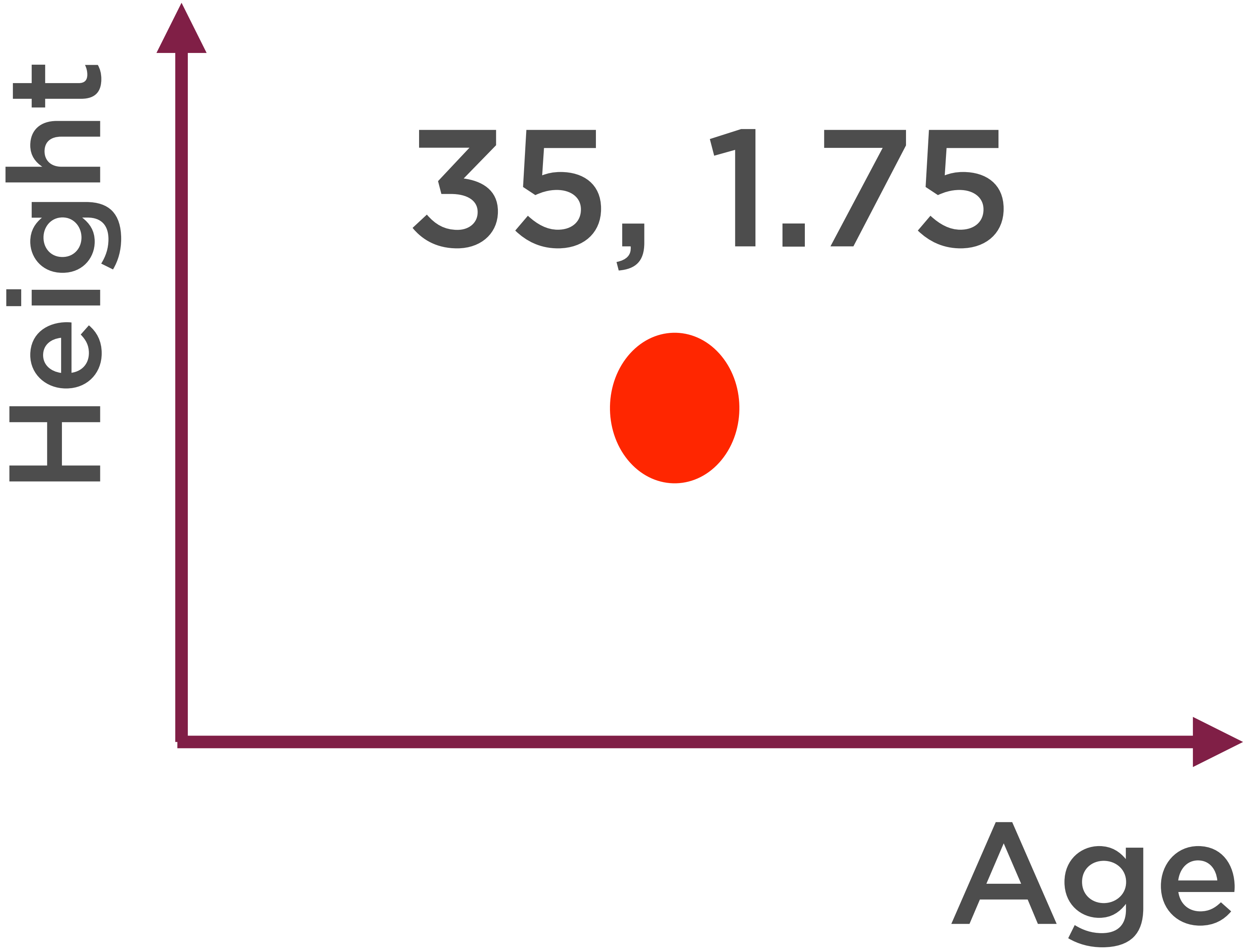


Anything can be represented by a set of numbers

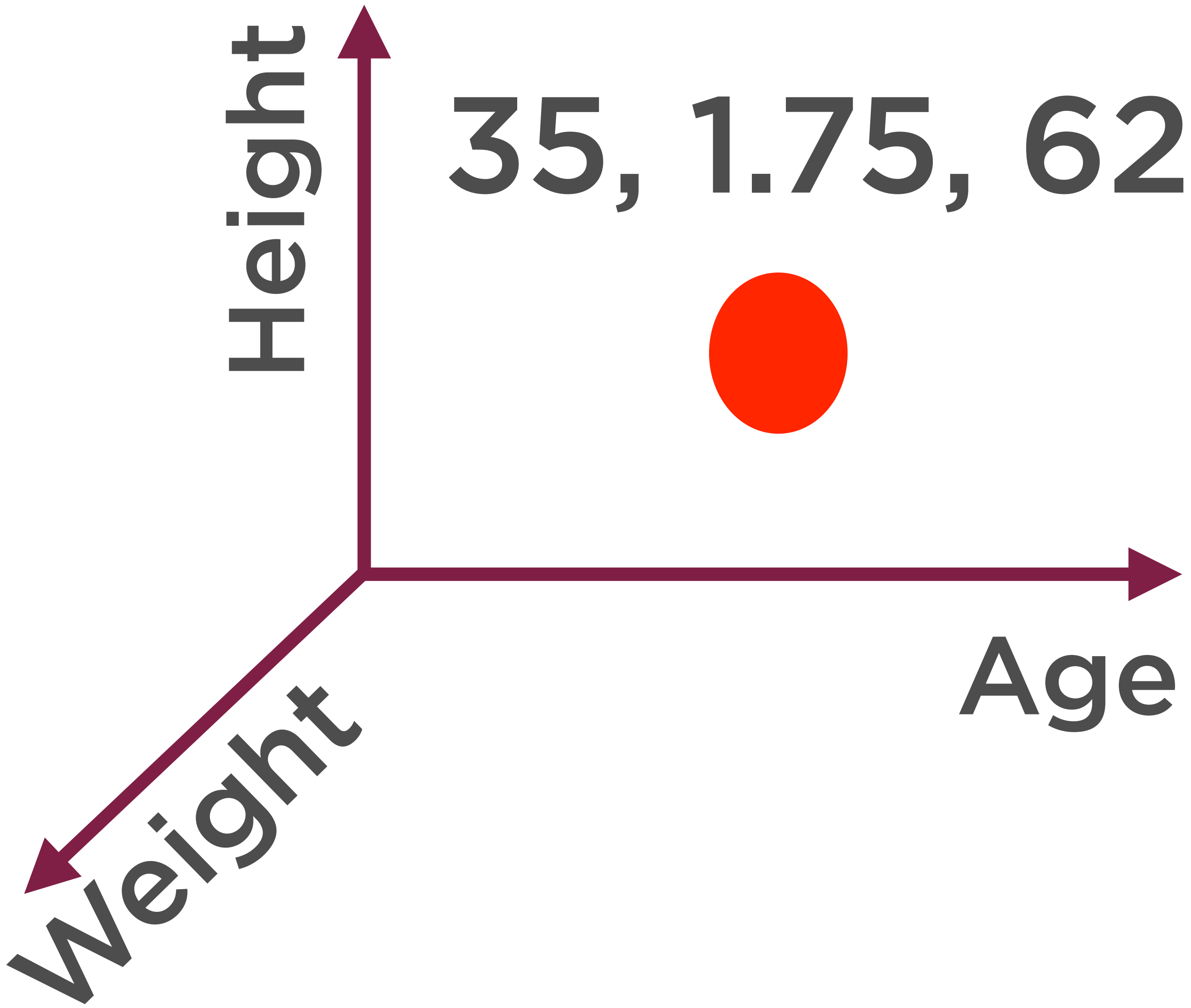
Age, Height, Weight



Age, Height, Weight



Age, Height, Weight

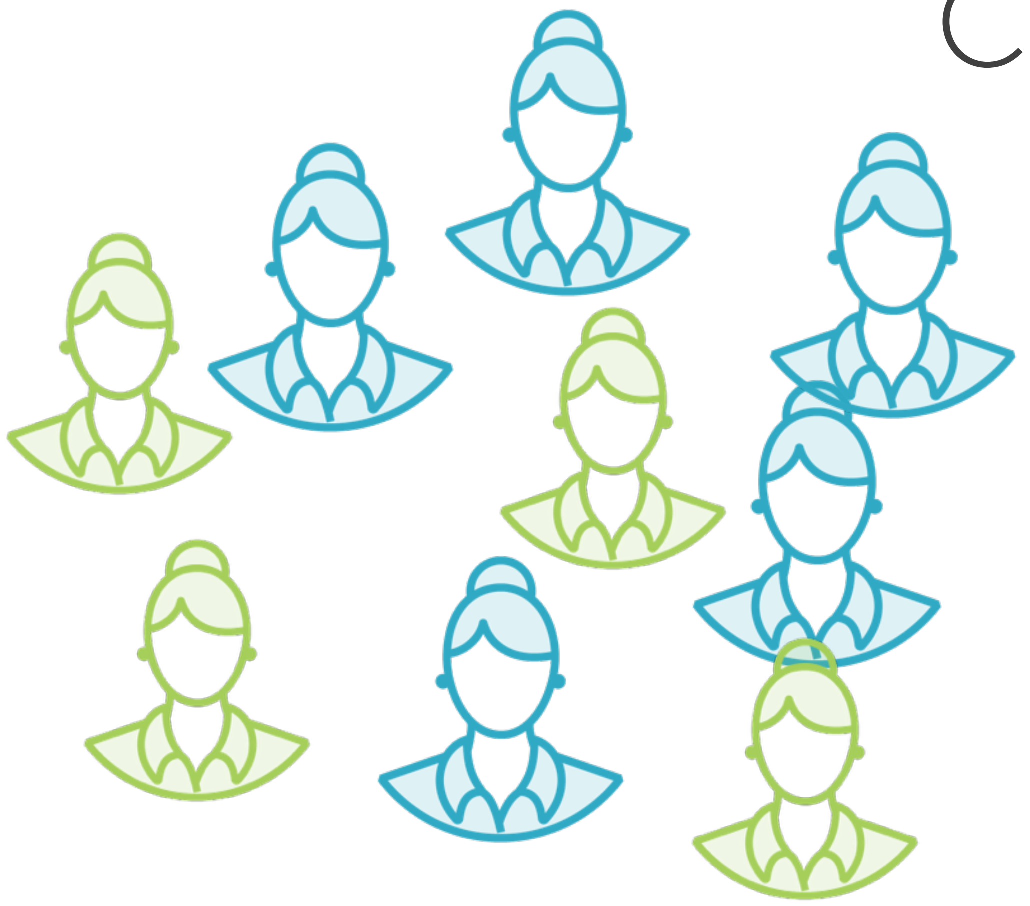


A set of N numbers represents a point in an **N-dimensional Hypercube**

Clustering

A set of points, each representing a Facebook user

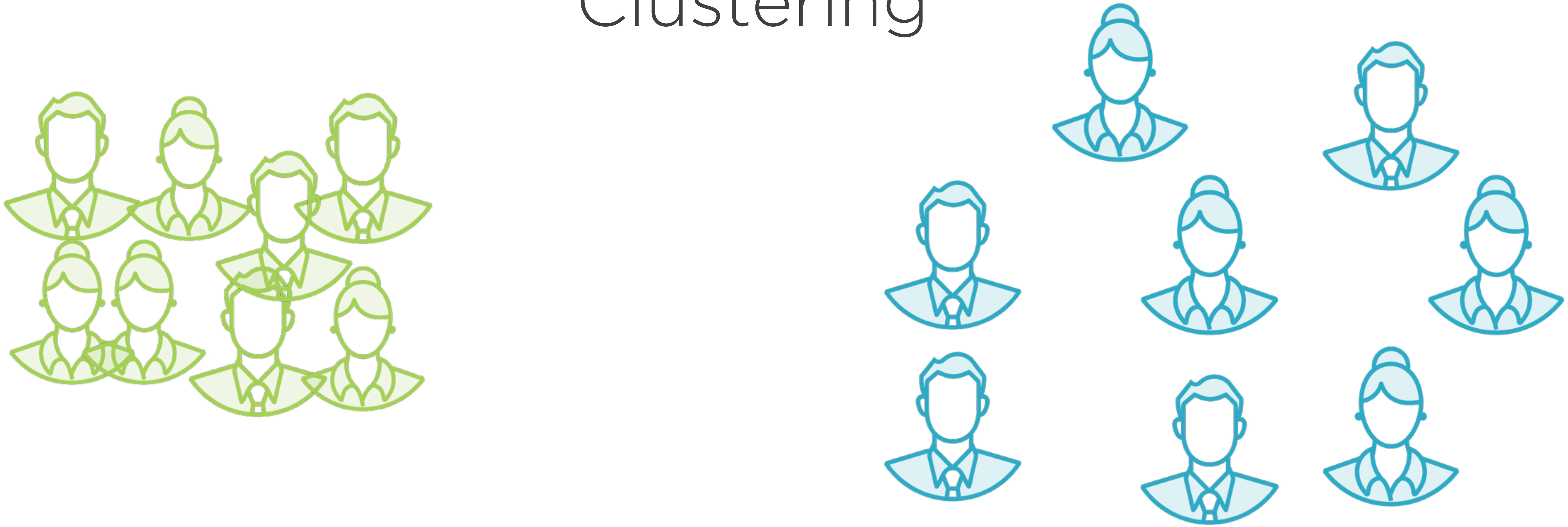
Clustering



Same group = similar

Different group = different

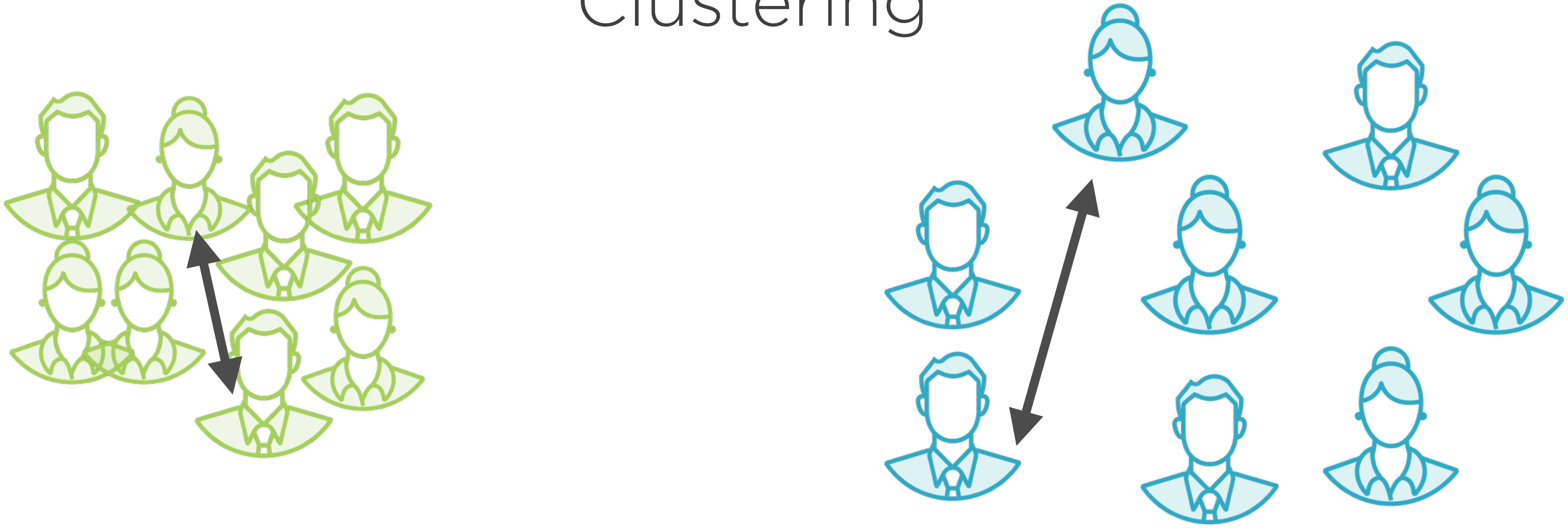
Clustering



Same group = similar

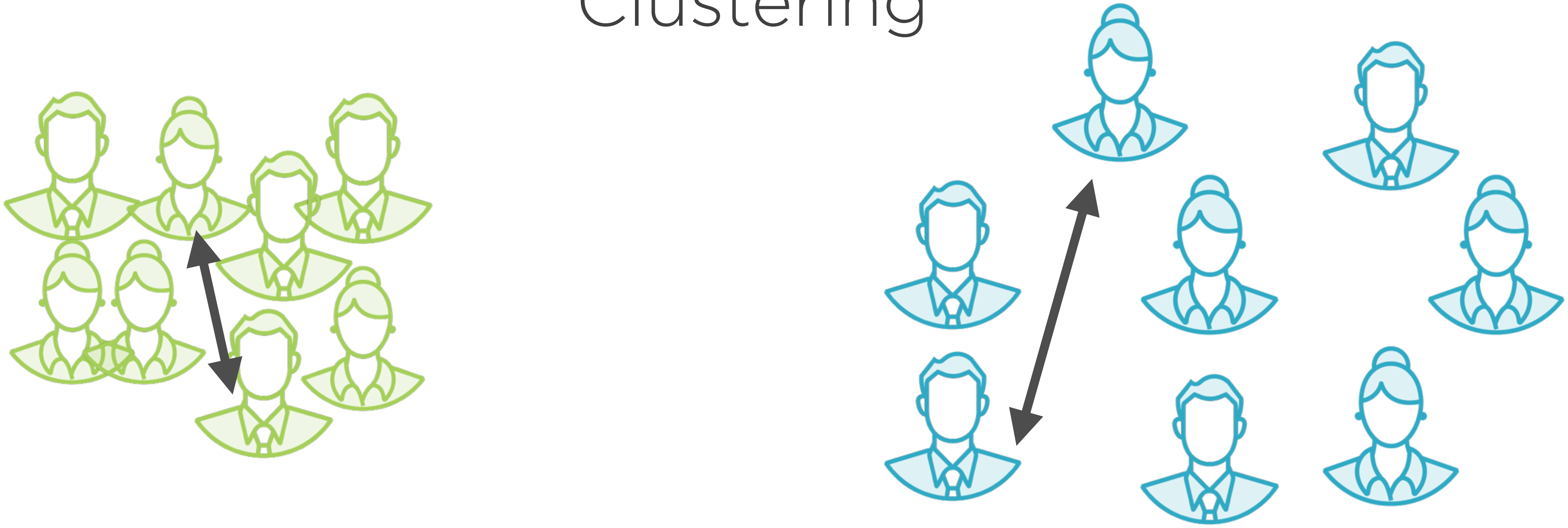
Different group = different

Clustering



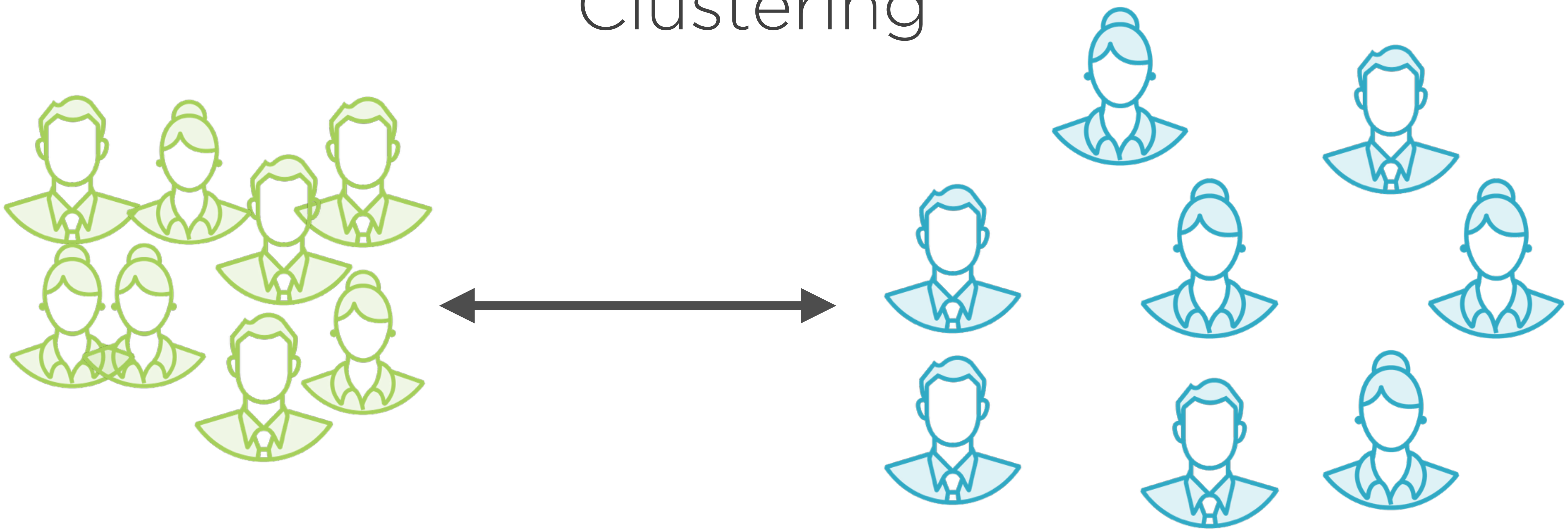
The distance between users in a cluster indicates how similar they are

Clustering

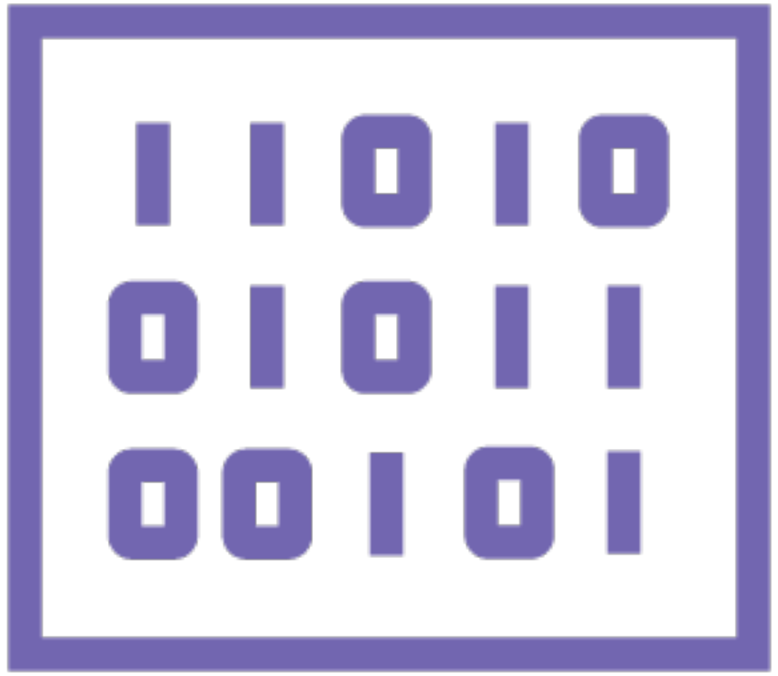


Maximize intra-cluster similarity

Clustering



Minimize inter-cluster similarity



Clustering Objective

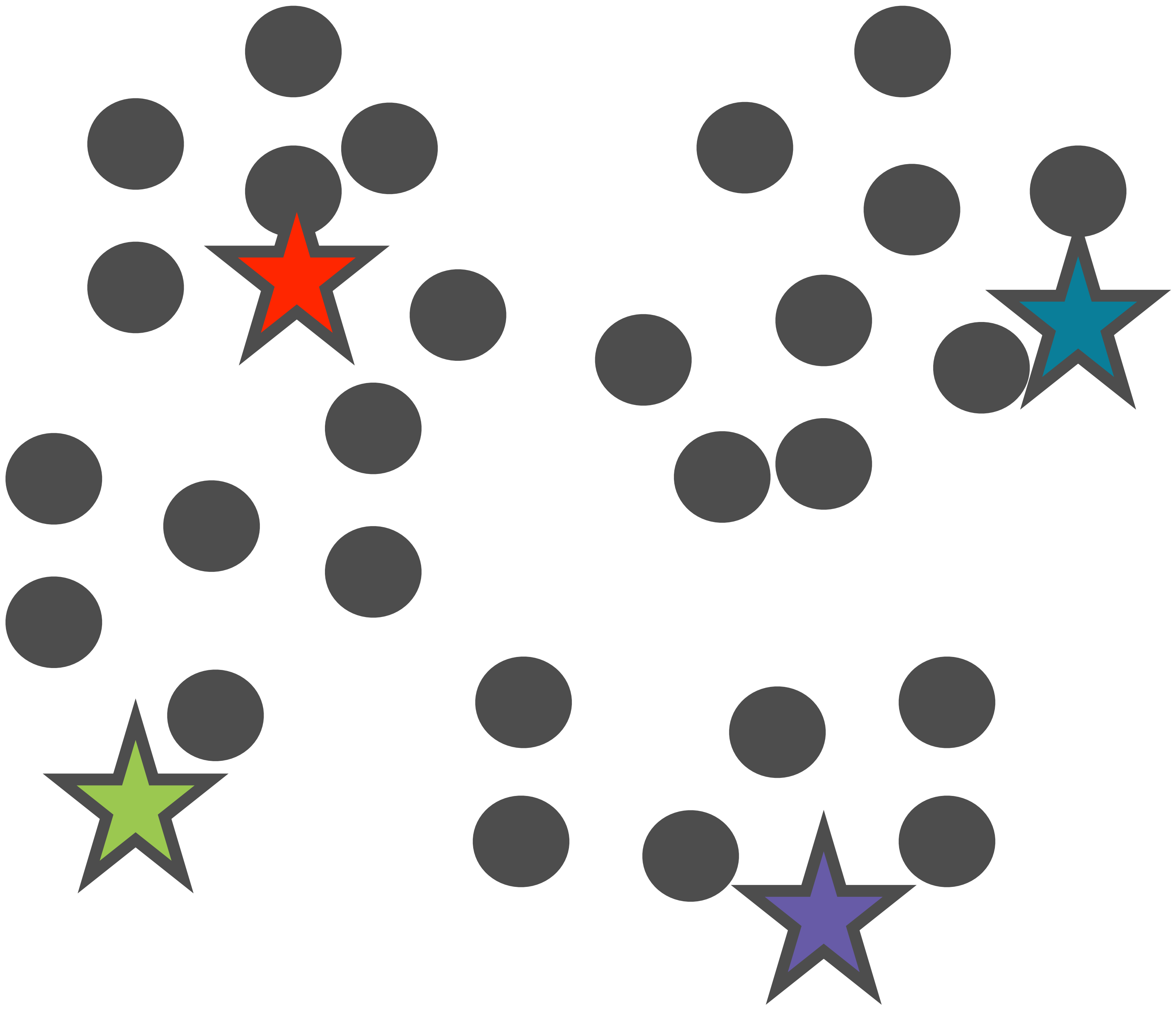
Maximize intra-cluster similarity

Minimize inter-cluster similarity

The **K-Means Clustering** algorithm is a famous Machine Learning algorithm to achieve this

K-Means Clustering

Initialize K centroids i.e. means

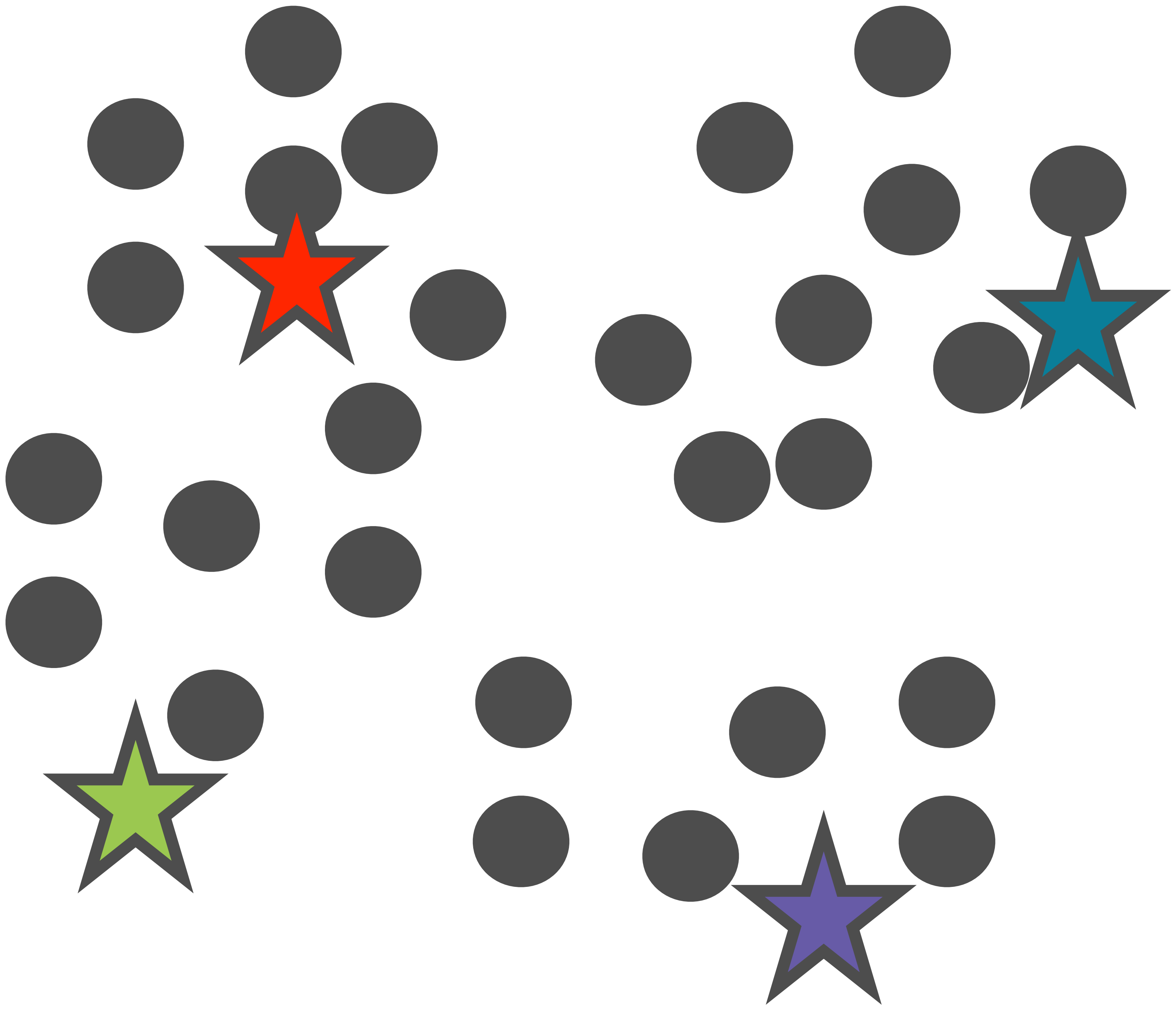


K-Means Clustering

These initial values are called the seeds

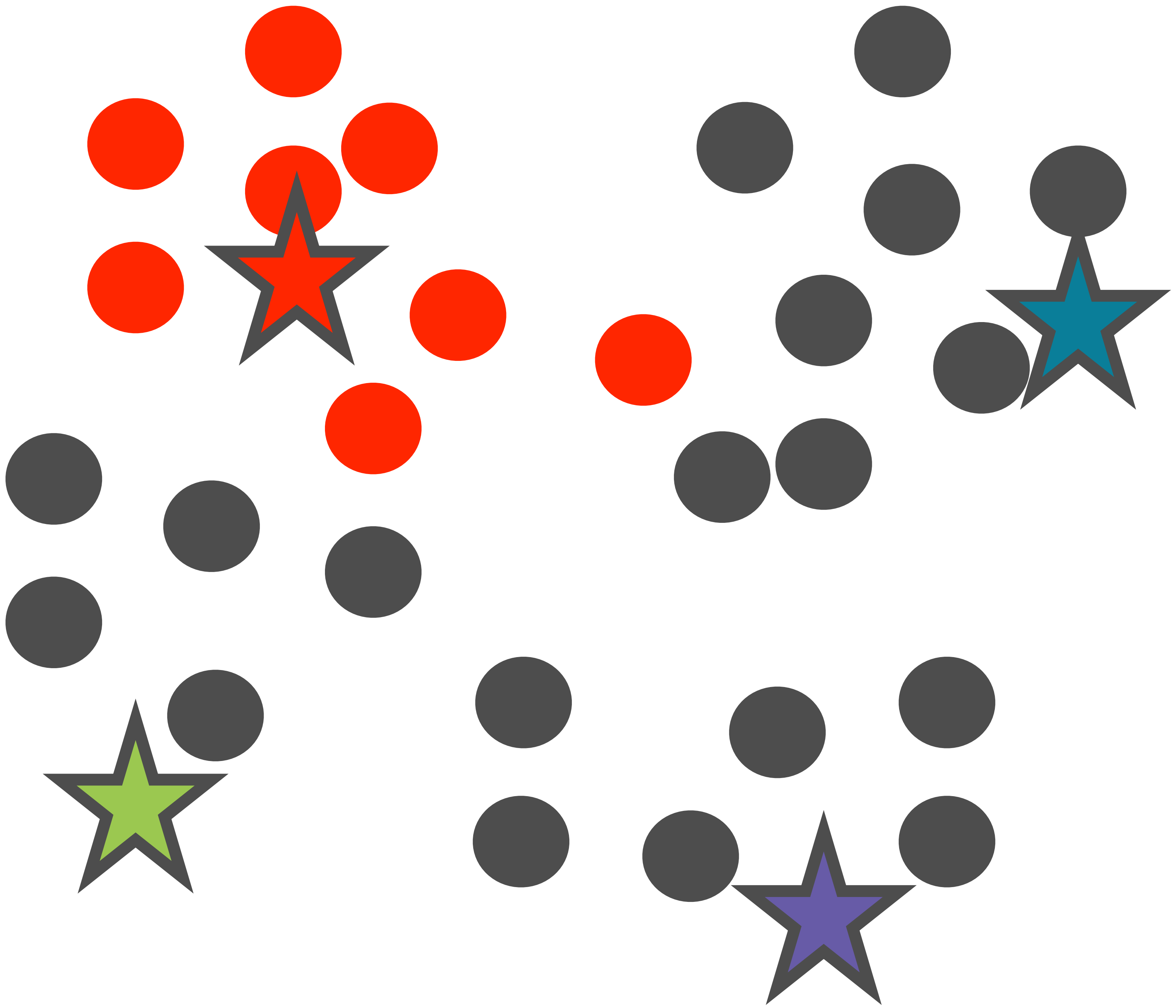
K-Means Clustering

Assign each point to a cluster



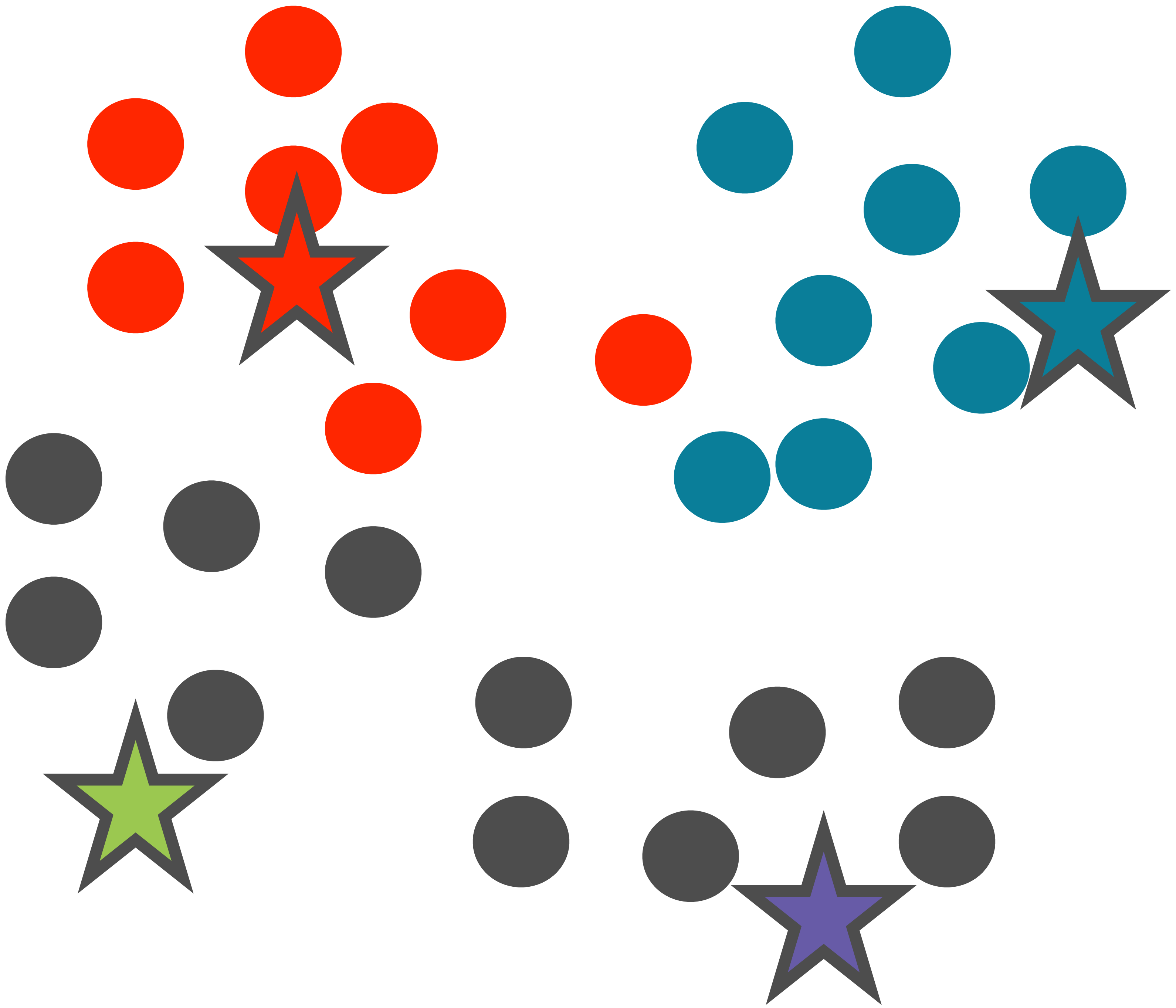
K-Means Clustering

Assign each point to a cluster



K-Means Clustering

Assign each point to a cluster



K-Means Clustering

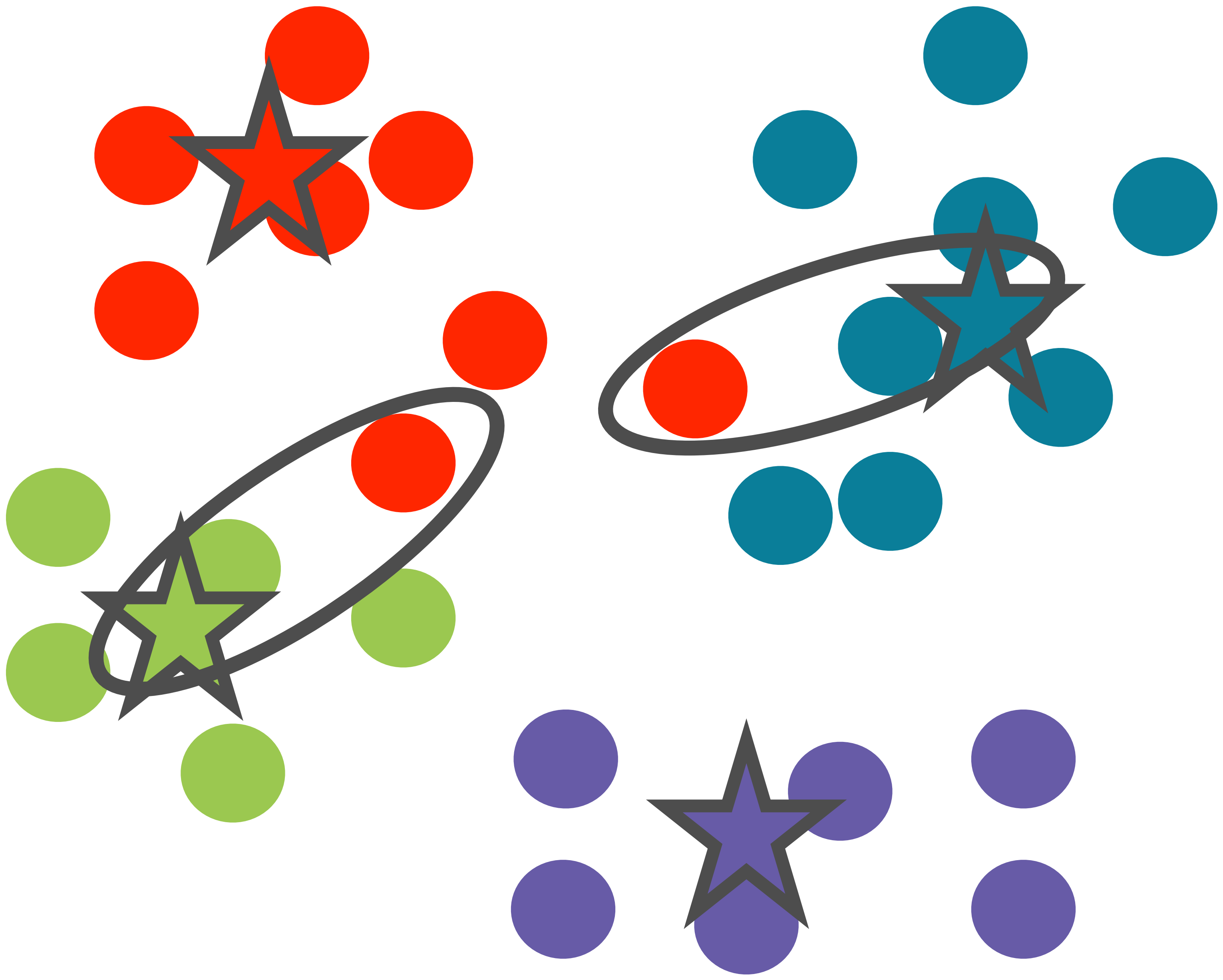
Assign each point to a cluster

K-Means Clustering

Recalculate the mean for each cluster

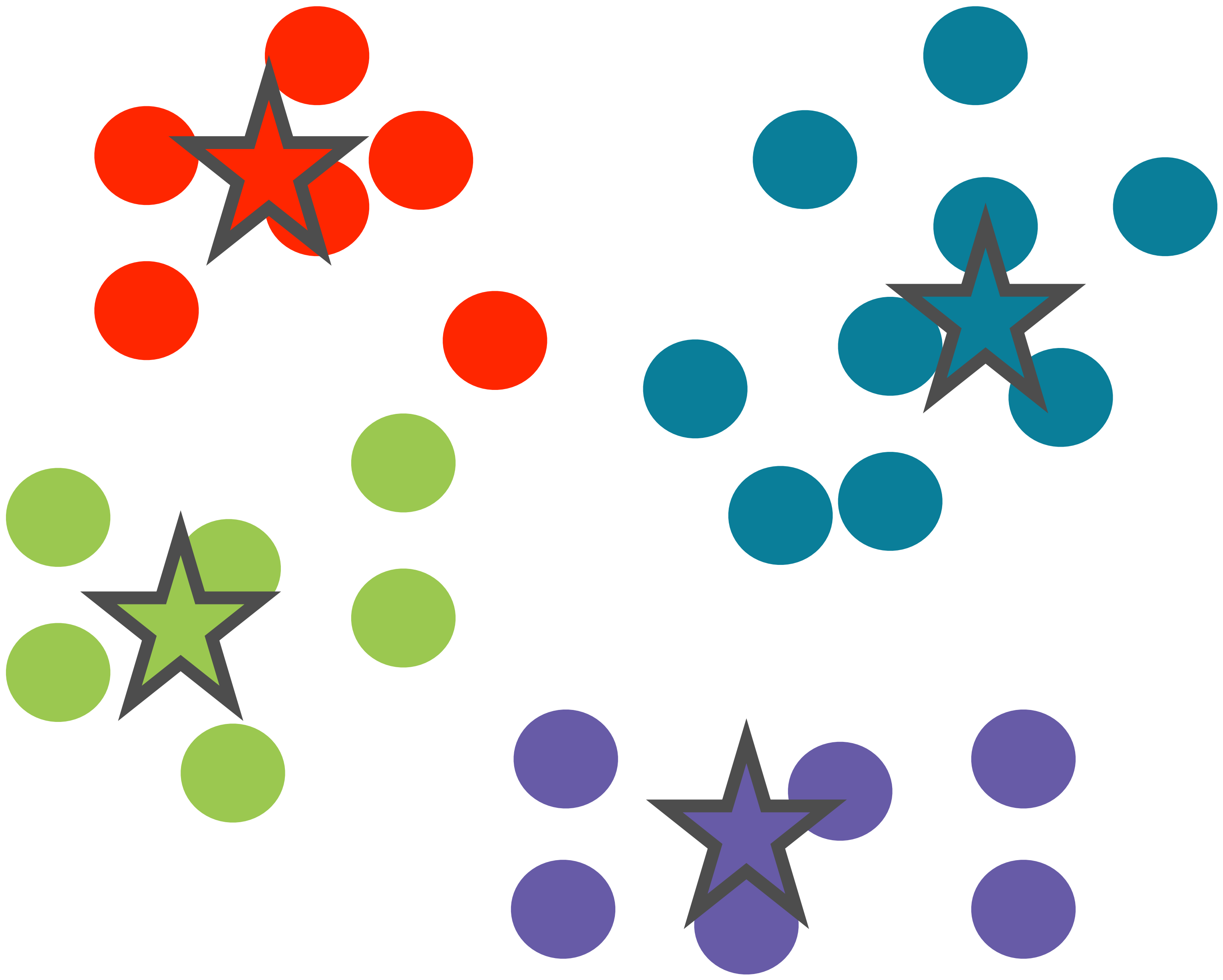
K-Means Clustering

Re-assign the points to clusters

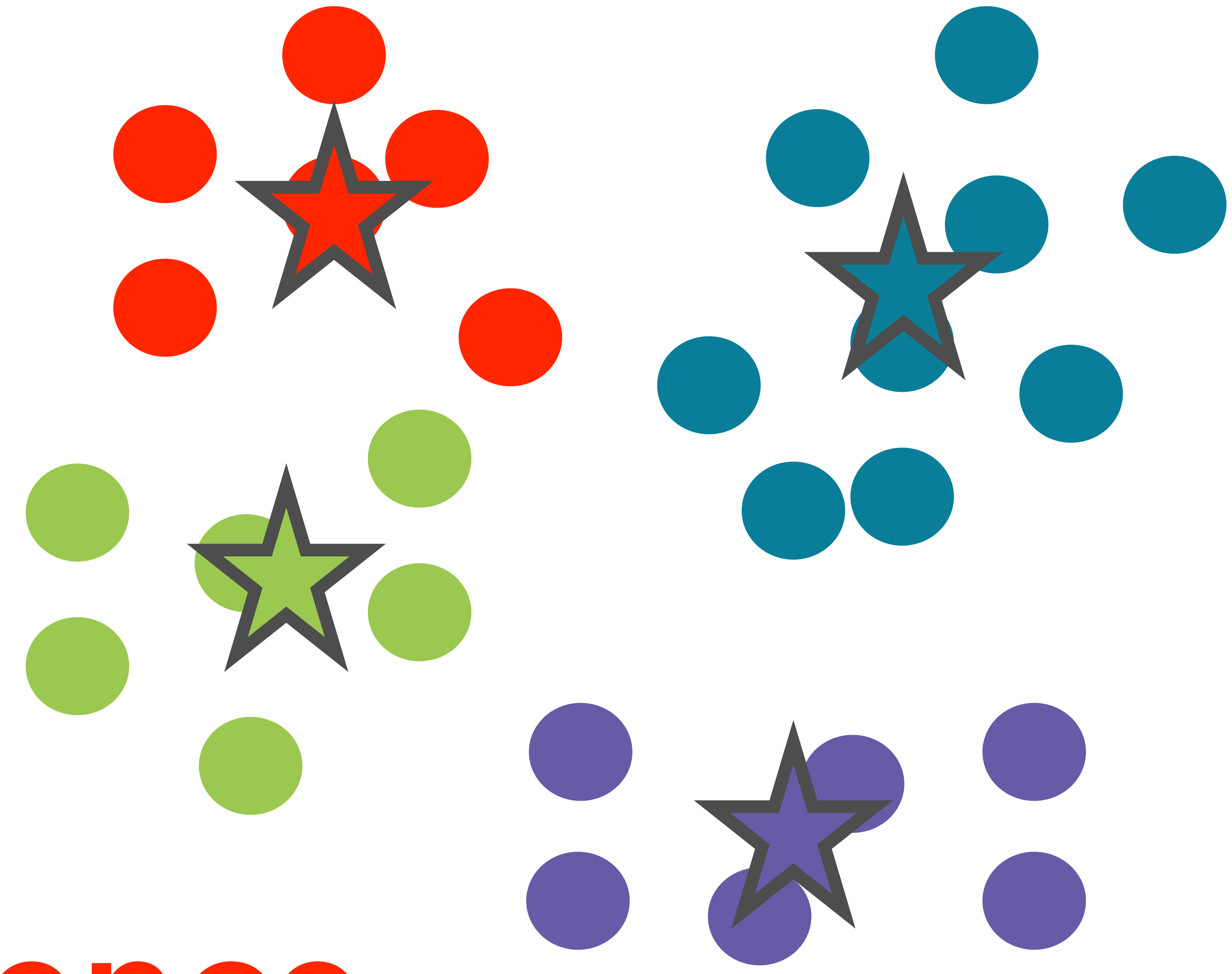


K-Means Clustering

Iterate until points are in their final clusters

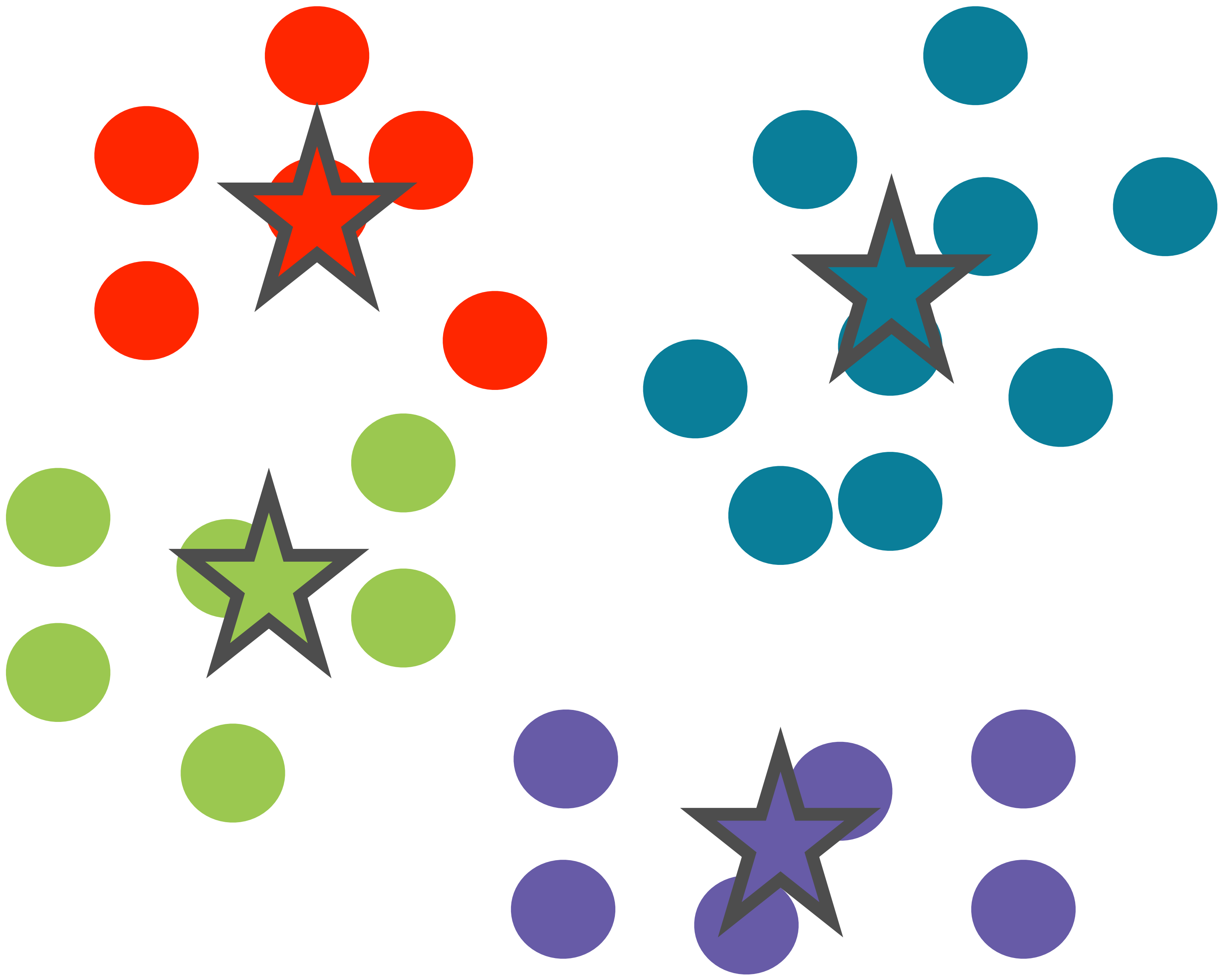


K-Means Clustering



Convergence

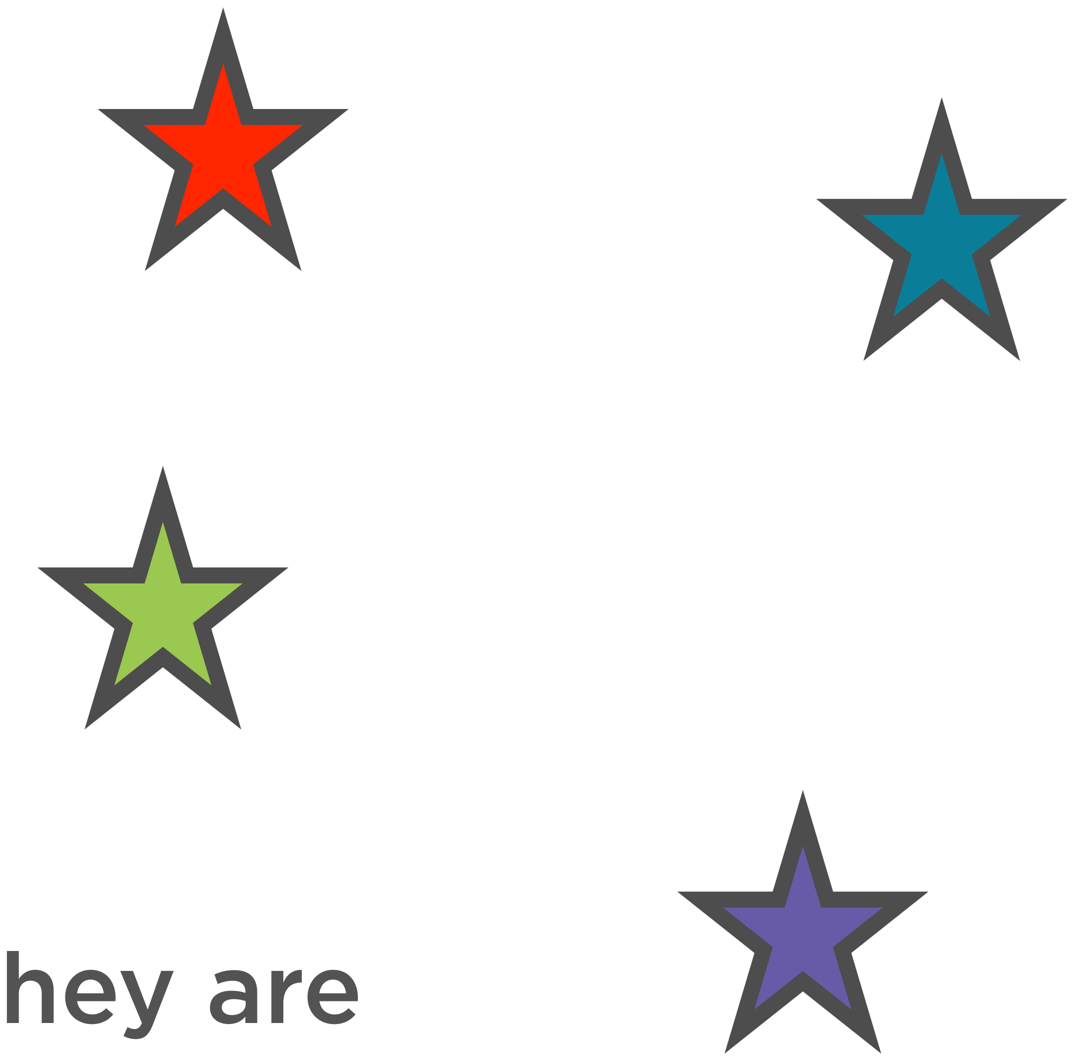
K-Means Clustering



K-Means Clustering

Each cluster has a representative point called a **reference vector**

K-Means Clustering



Because of how they are calculated, these reference vectors are often called centroids

```
Initialise K centroids

Repeat:
  For each data point:
    Assign to "nearest" cluster


  For each centroid:
    Update coordinates

Have centroids converged?
   Yes: Stop, we're done
    No: Keep iterating
```

- Pick an initial solution (algorithms exist to pick well)
- Iterate until convergence
  - Update assignments of points to clusters
  - Update coordinates of reference vectors
- Keep iterating until we converge

```
Initialise K centroids


Repeat:
  For each data point:
    Assign to "nearest" cluster


  For each centroid:
    Update coordinates

Have centroids converged?
   Yes: Stop, we're done
    No: Keep iterating
```

- Hyperparameters
  - Number of clusters
  - Seeds (Initial values of centroids)

```
Initialise K centroids

Repeat:
  For each data point:
    Assign to "nearest" cluster

  For each centroid:
    Update coordinates

Have centroids converged?
   Yes: Stop, we're done
    No: Keep iterating
```

- **Design choice #1:**
  - **Distance measure between point, cluster**
  - **Euclidean distance often used**

```
Initialise K centroids

Repeat:
  For each data point:
    Assign to "nearest" cluster

  For each centroid:
    Update coordinates

Have centroids converged?
   Yes: Stop, we're done
    No: Keep iterating
```

- **Design choice #2:**
  - **Calculating cluster center from points in cluster**
  - **Centroid (simple average) often used**

Hyperparameter Tuning in K-Means Clustering

Hyperparameters

Model configuration properties that define a model, and remain constant during the training of the model

Understanding Hyperparameters

Model Inputs

Model Parameters

Model Hyperparameters

Understanding Hyperparameters

Model Inputs

Input data points, training dataset

Model Parameters

Model Hyperparameters

Understanding Hyperparameters

Model Inputs

Input data points, training dataset

Model Parameters

Reference vectors, i.e. centroids of each cluster

Model Hyperparameters

Understanding Hyperparameters

Model Inputs

Input data points, training dataset

Model Parameters

Reference vectors, i.e. centroids of each cluster

Model Hyperparameters

Number of clusters, initial values, distance measure

Hyperparameters in K-Means Clustering

Number of clusters

Seeds - initial values

Distance measures

Number of Clusters

K is the most important hyperparameter

Sometimes obvious e.g. 10 in MNIST digit classification

Otherwise, apply standard method to find the "best" value of K

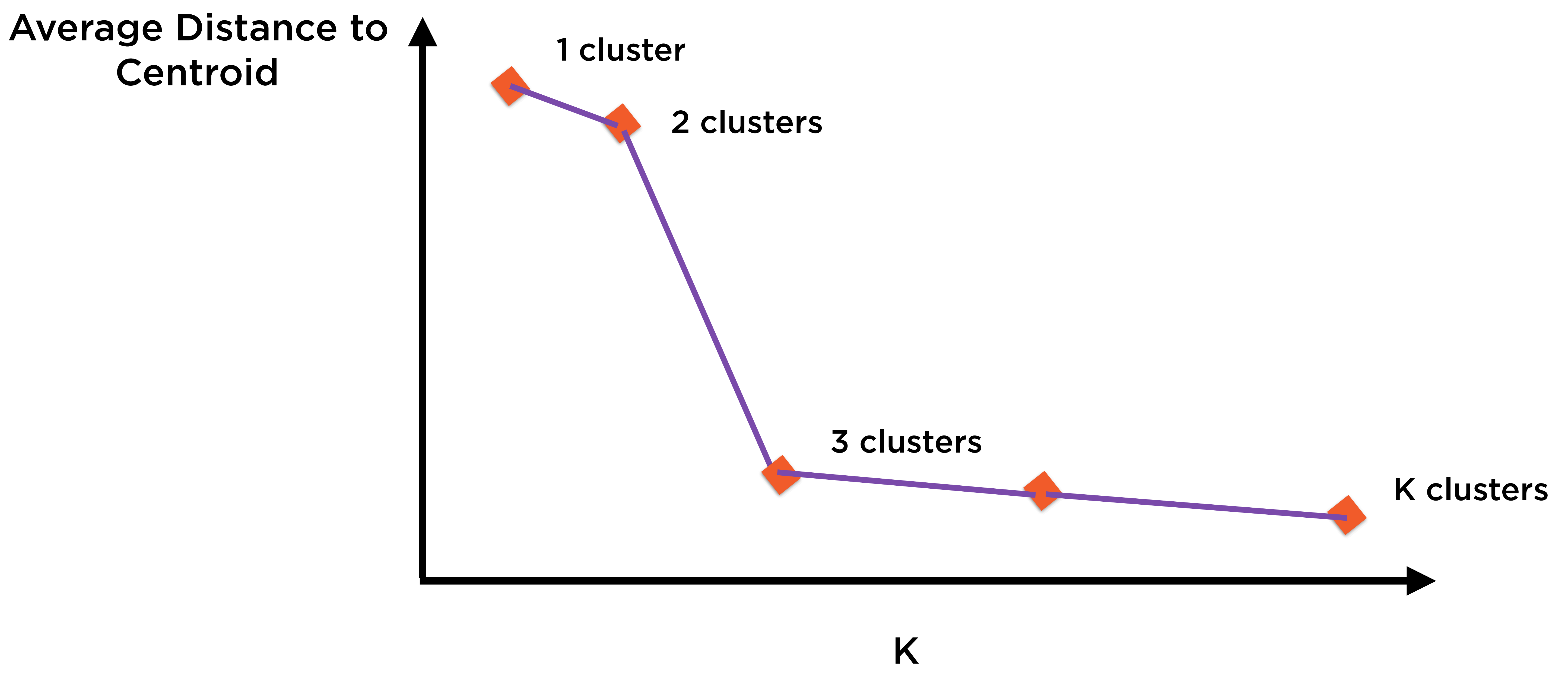
Elbow Method

Pick range of candidate values of K (e.g. 1 to 10)

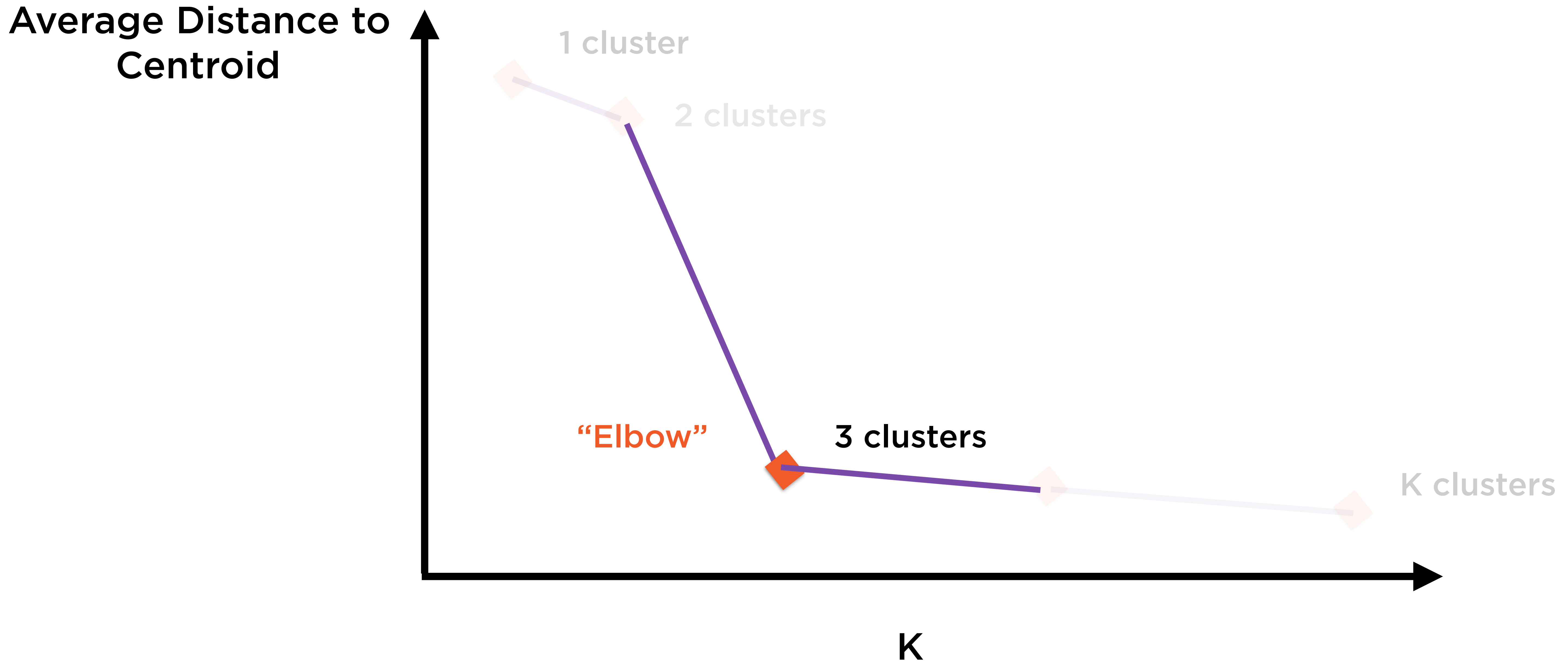
Calculate average distance from centroid for each value

Plot and find “elbow”

Elbow Method



Elbow Method



Silhouette Method

Pick range of candidate values of K (e.g. 1 to 10)

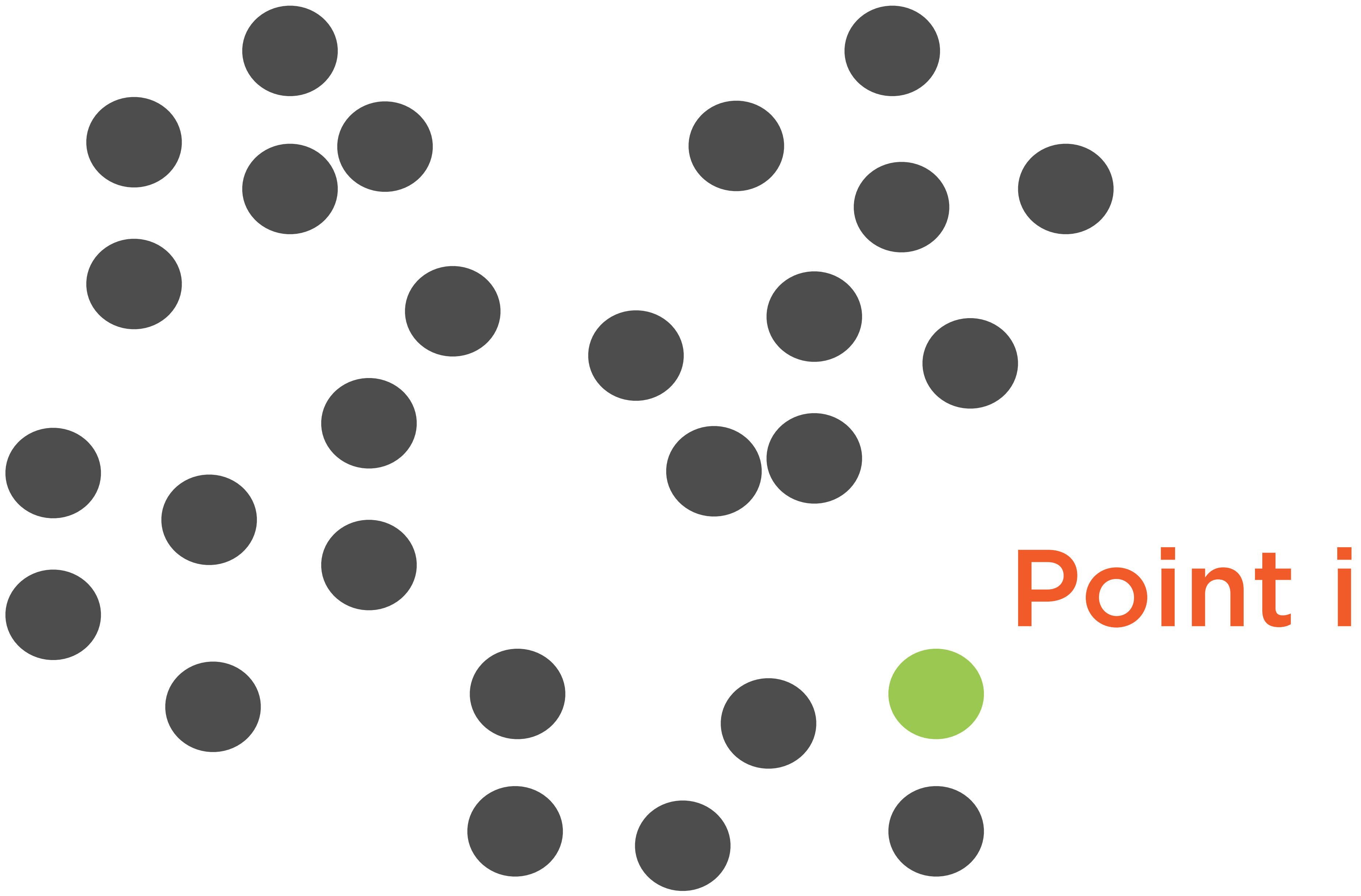
Plot silhouettes for each value of K

Ideal value of silhouette = 1

Worst possible value of silhouette = -1

Silhouette Coefficient

For any point i, calculate silhouette coefficient



Silhouette Coefficient

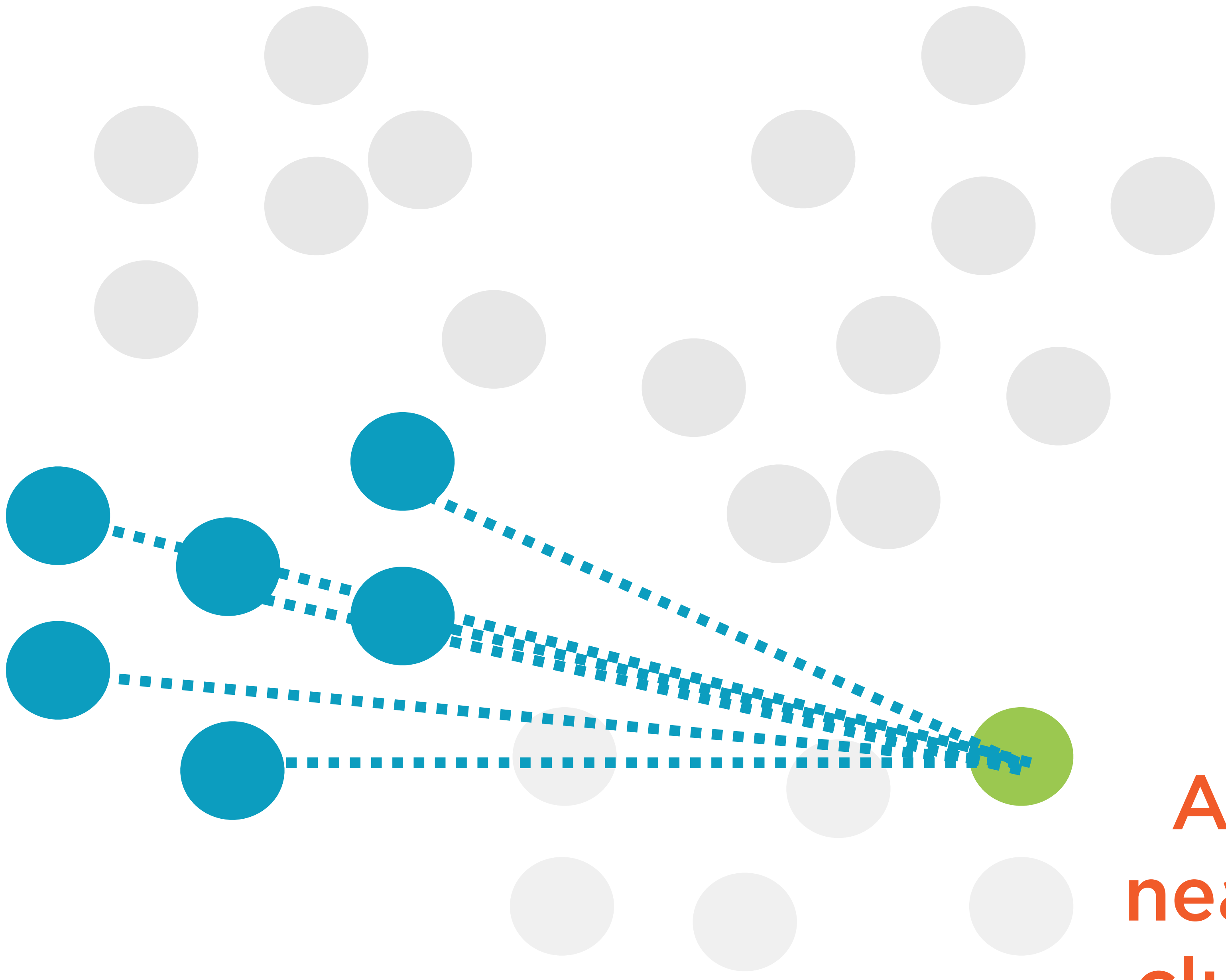
For any point i, calculate silhouette coefficient

Silhouette Coefficient

Find a(i) = average distance of i to other points in same cluster

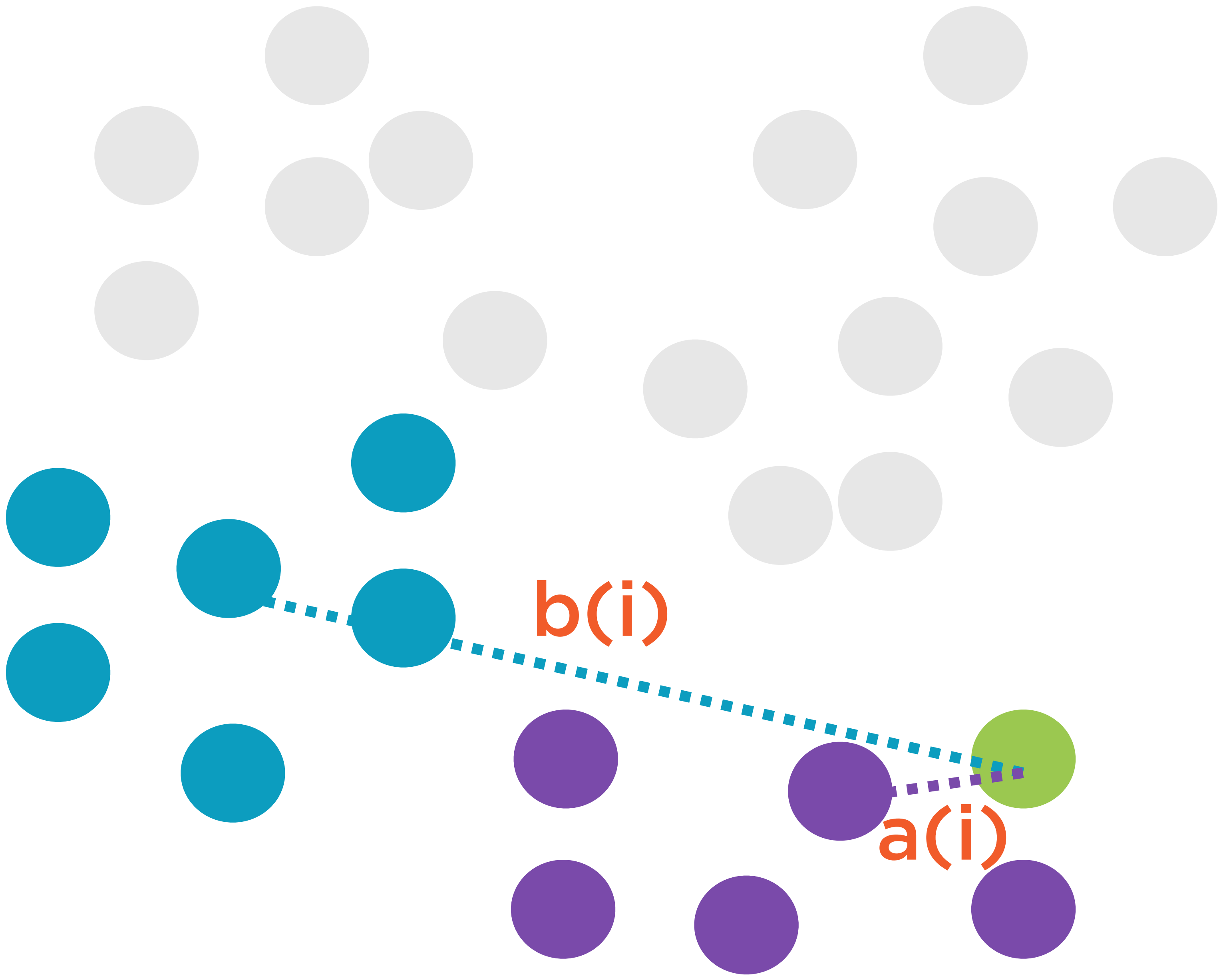
Silhouette Coefficient

Find b(i) = average distance to nearest other cluster



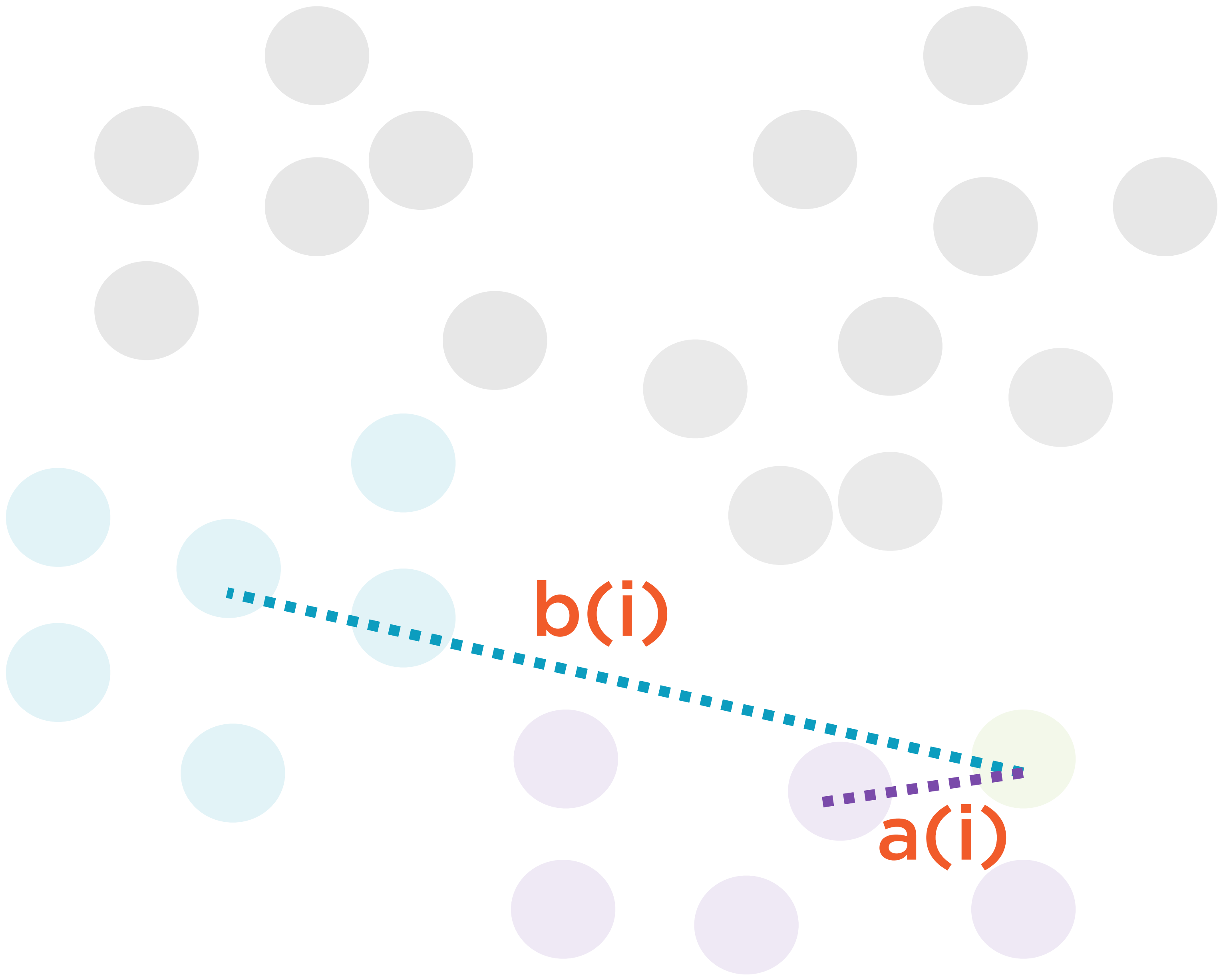
Silhouette Coefficient

Ideally, a(i) << b(i)



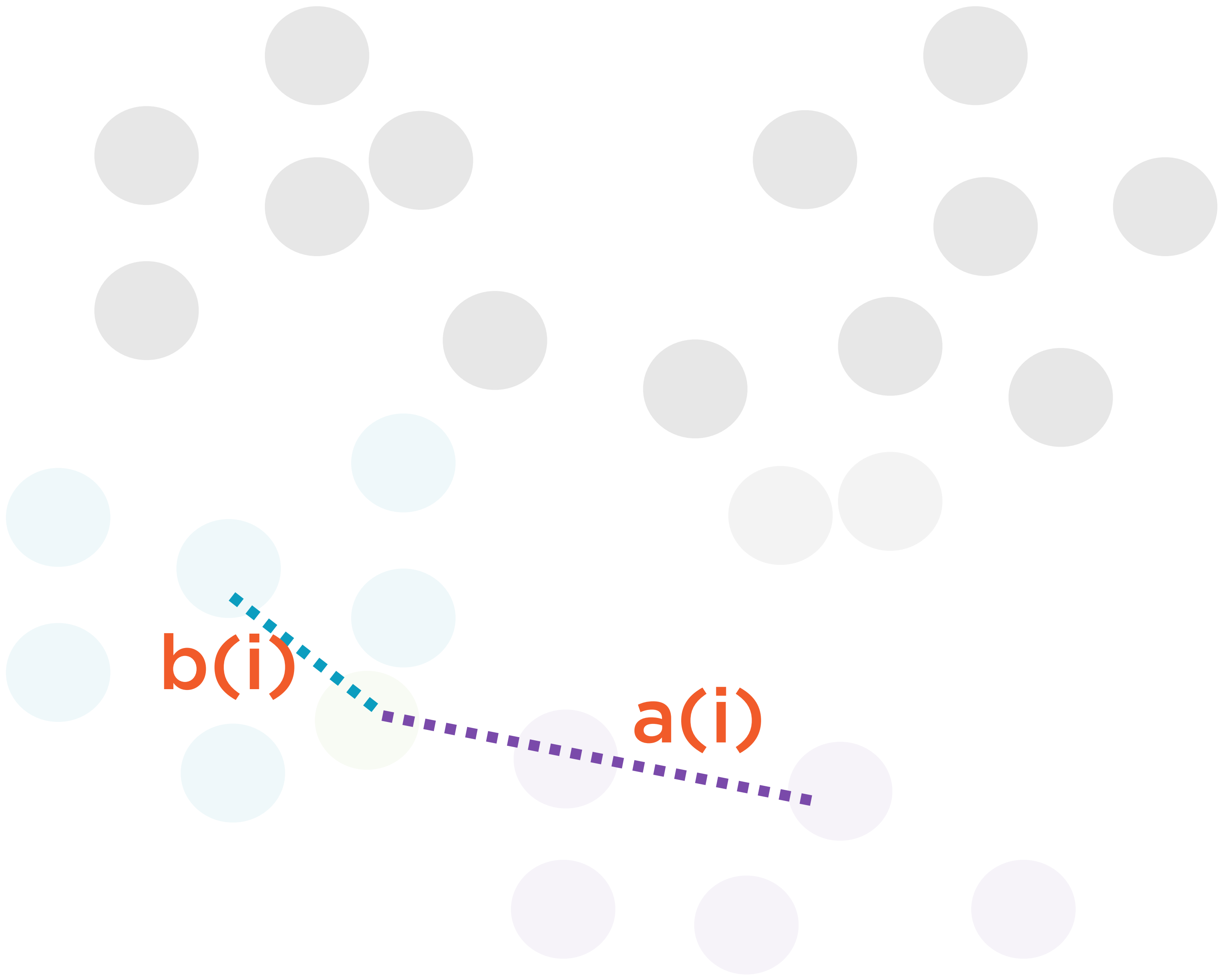
Silhouette Coefficient

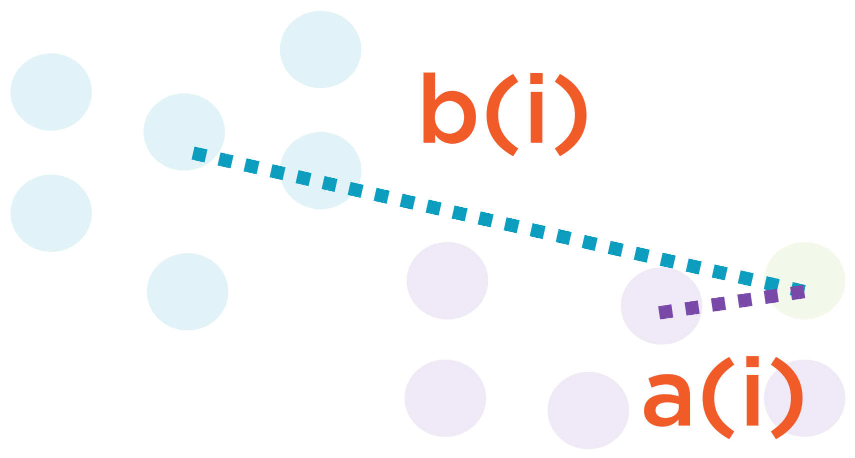
Ideally, a(i) << b(i)



Silhouette Coefficient

If a(i) > b(i), i is likely misclassified





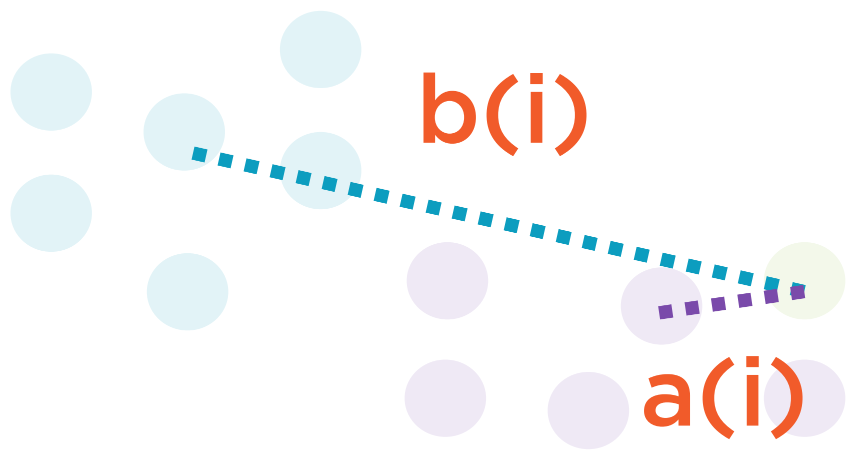
Silhouette Coefficient

For any point i

$$s(i) = \frac{b(i) - a(i)}{\text{Larger of } b(i) \text{ and } a(i)}$$

a(i) = Average distance inside cluster

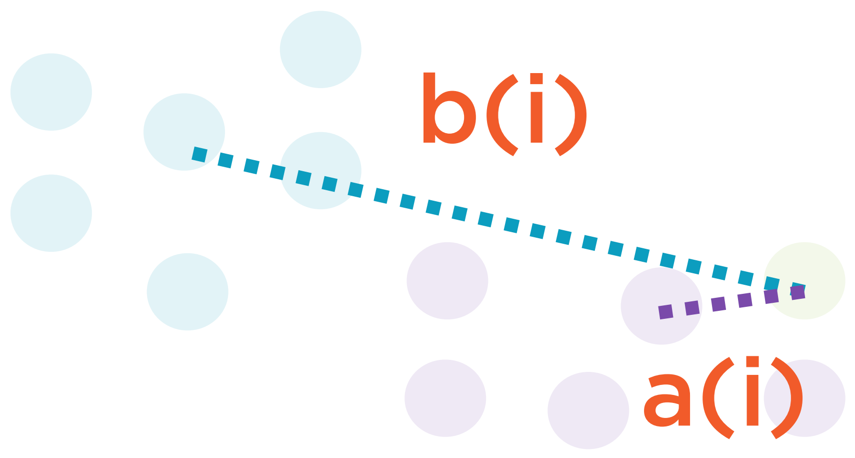
b(i) = Average distance to nearest other cluster



Ideally s(i) = 1

Ideally, a(i) = 0, b(i) = Infinity

$$s(i) = \frac{b(i) - a(i)}{\text{Larger of } b(i) \text{ and } a(i)} = 1$$



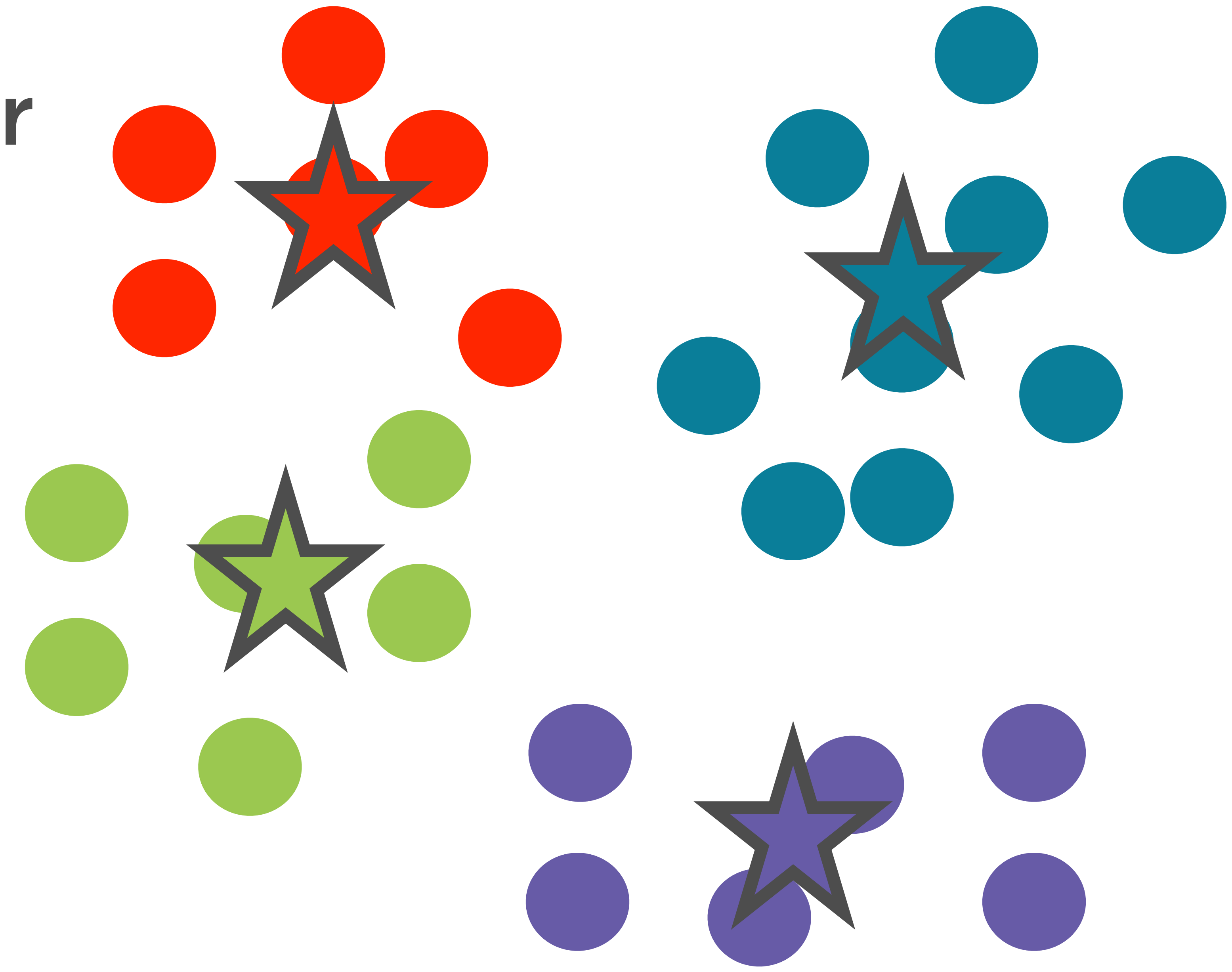
Worst-case s(i) = -1

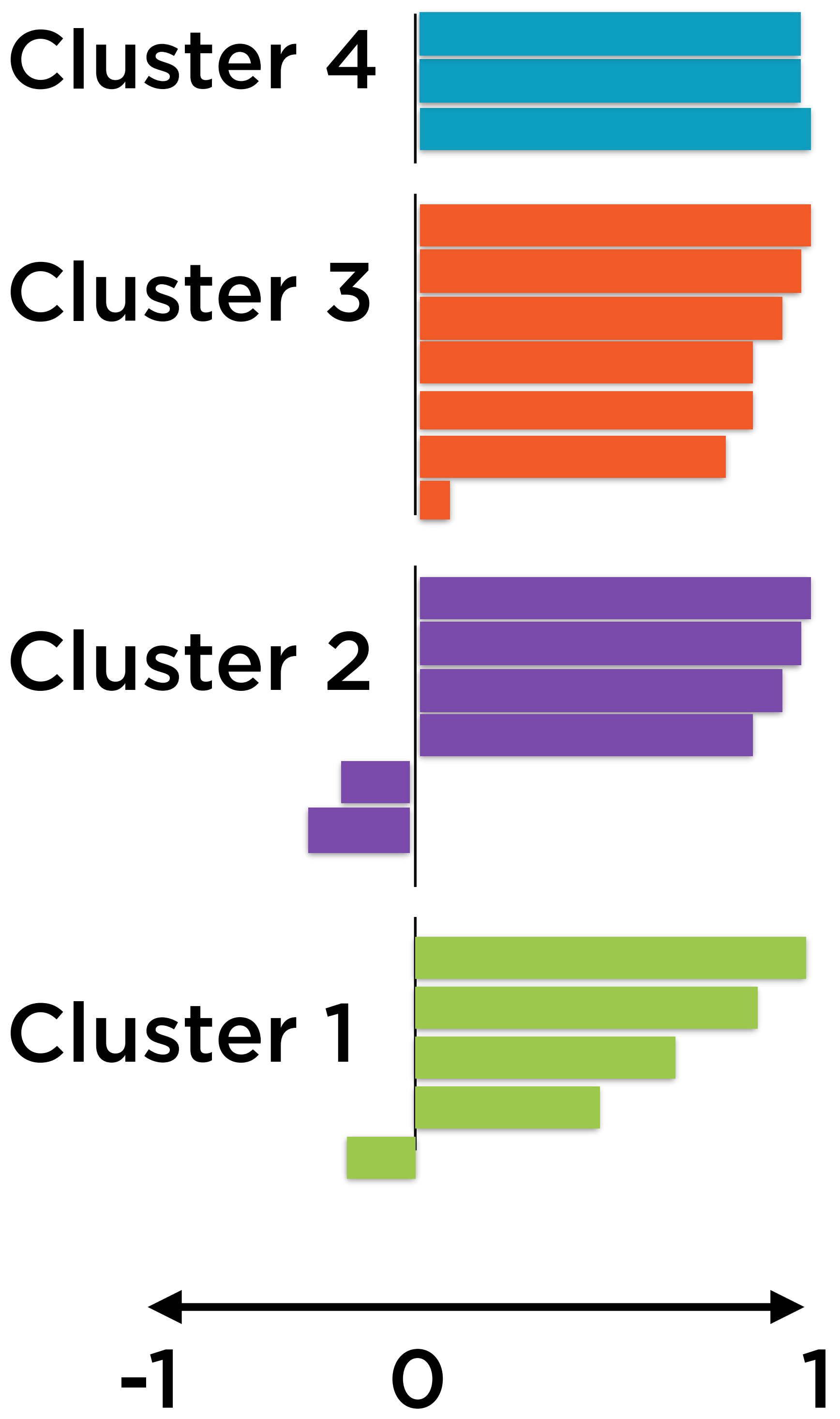
Worst case, a(i) = Infinity, b(i) = 0

$$s(i) = \frac{b(i) - a(i)}{\text{Larger of } b(i) \text{ and } a(i)} = -1$$

Silhouette Plot

Calculate s(i) for each point

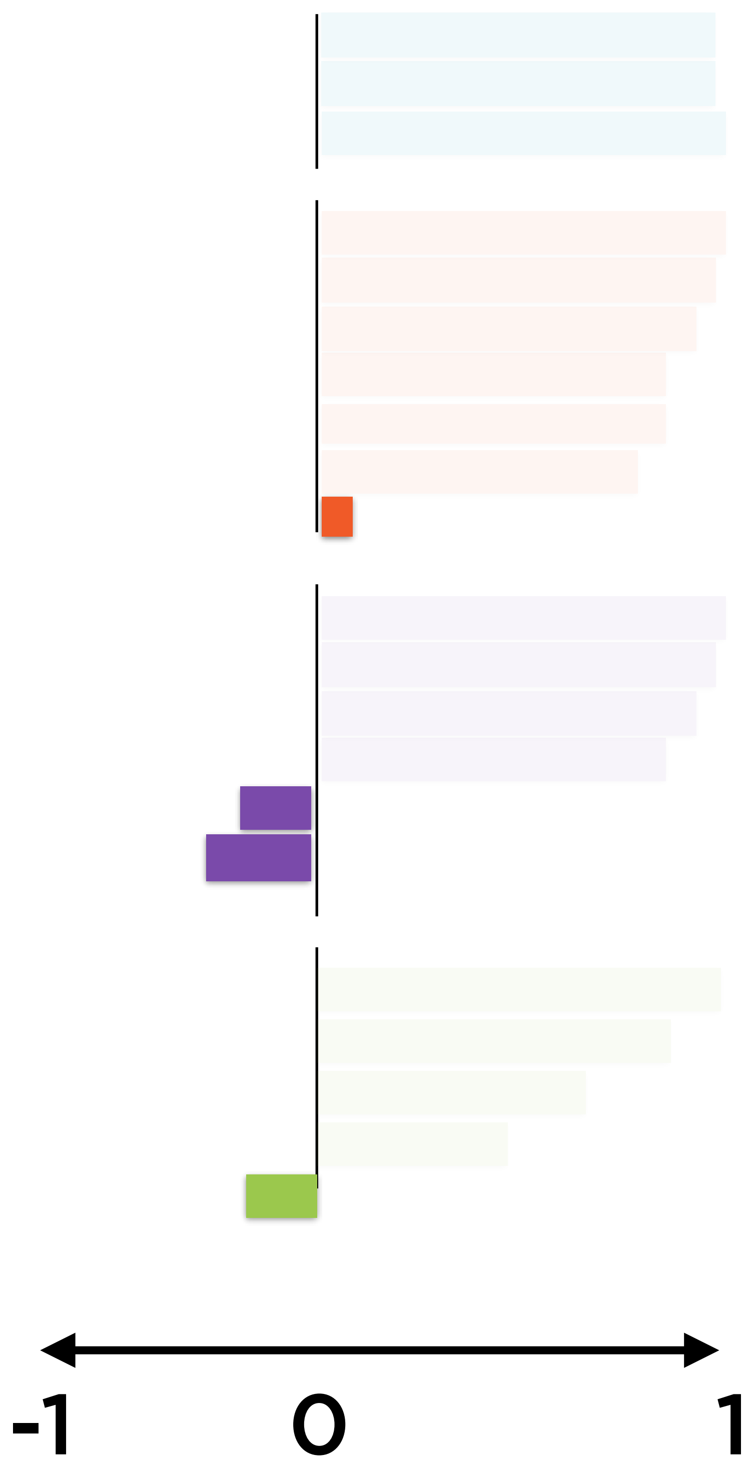




Silhouette Plot

Calculate s(i) for each point

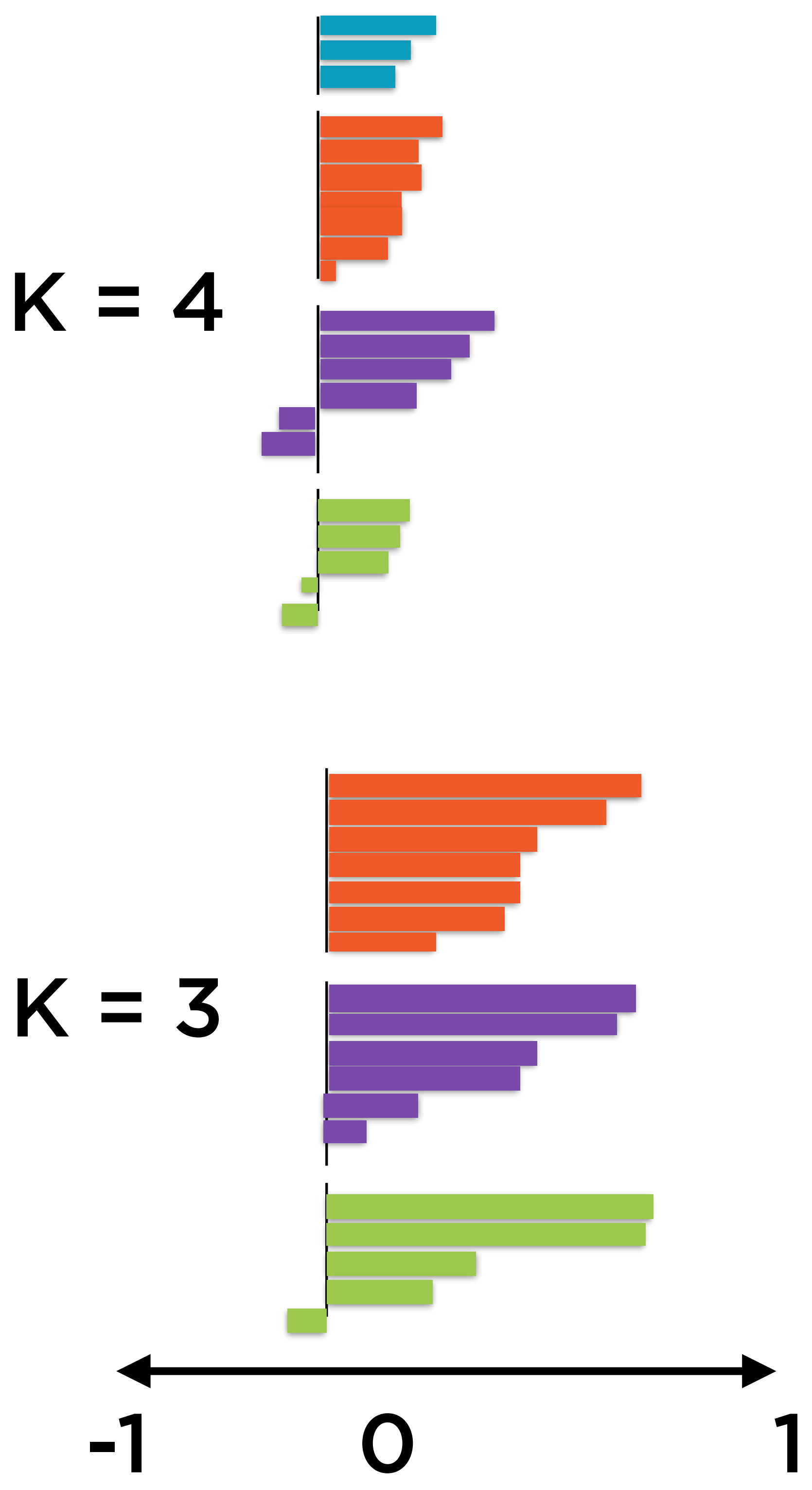
Plot value of s(i) to identify outliers



Outliers

Ideally, s(i) = 1

So, s(i) < 0 indicates outliers



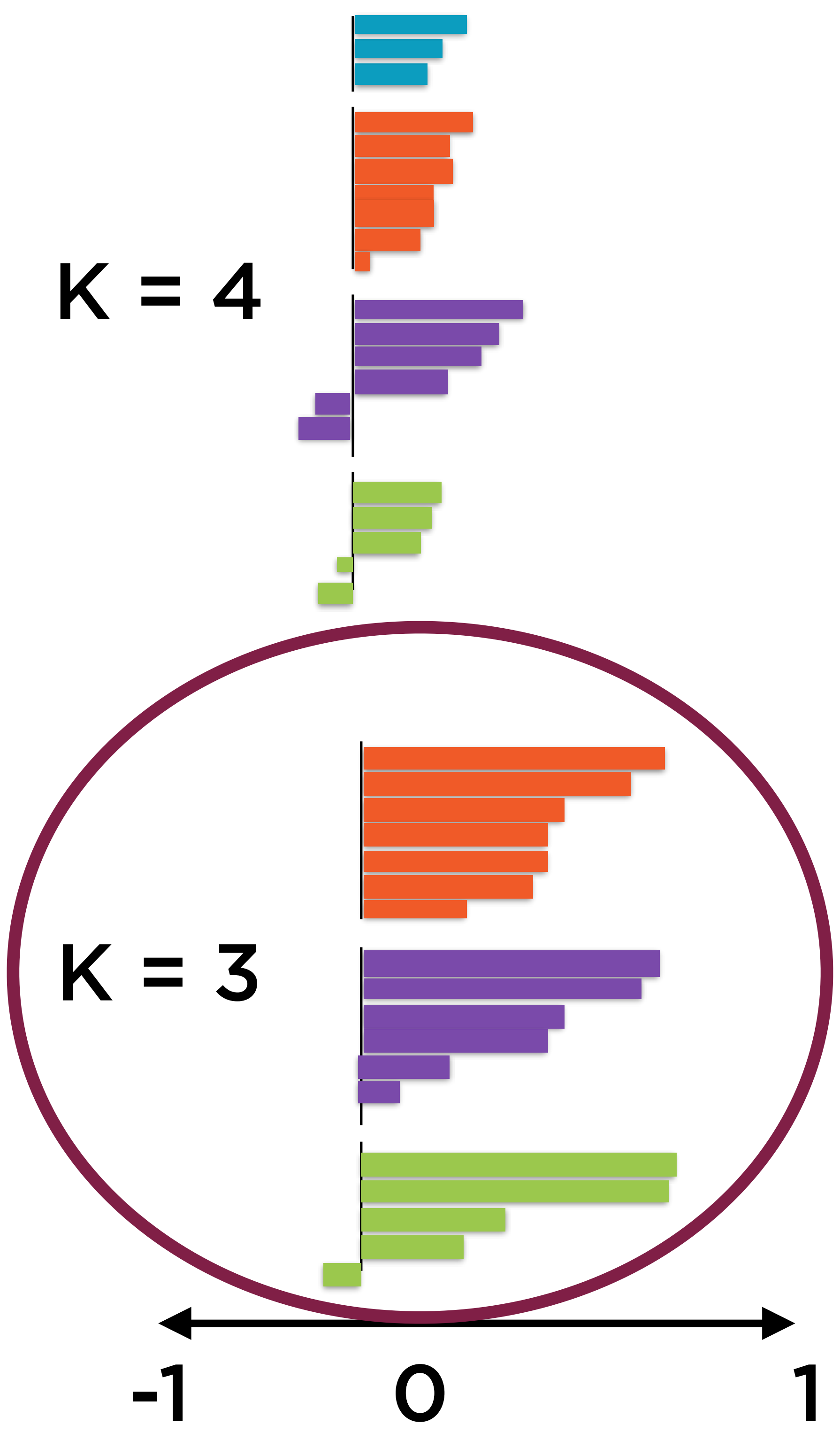
“Best” K

Extend the same idea

Replicate plot for different values of K

Pick K where average silhouette is closest to 1

“Best” K



Here K = 3 is noticeably better than K = 4

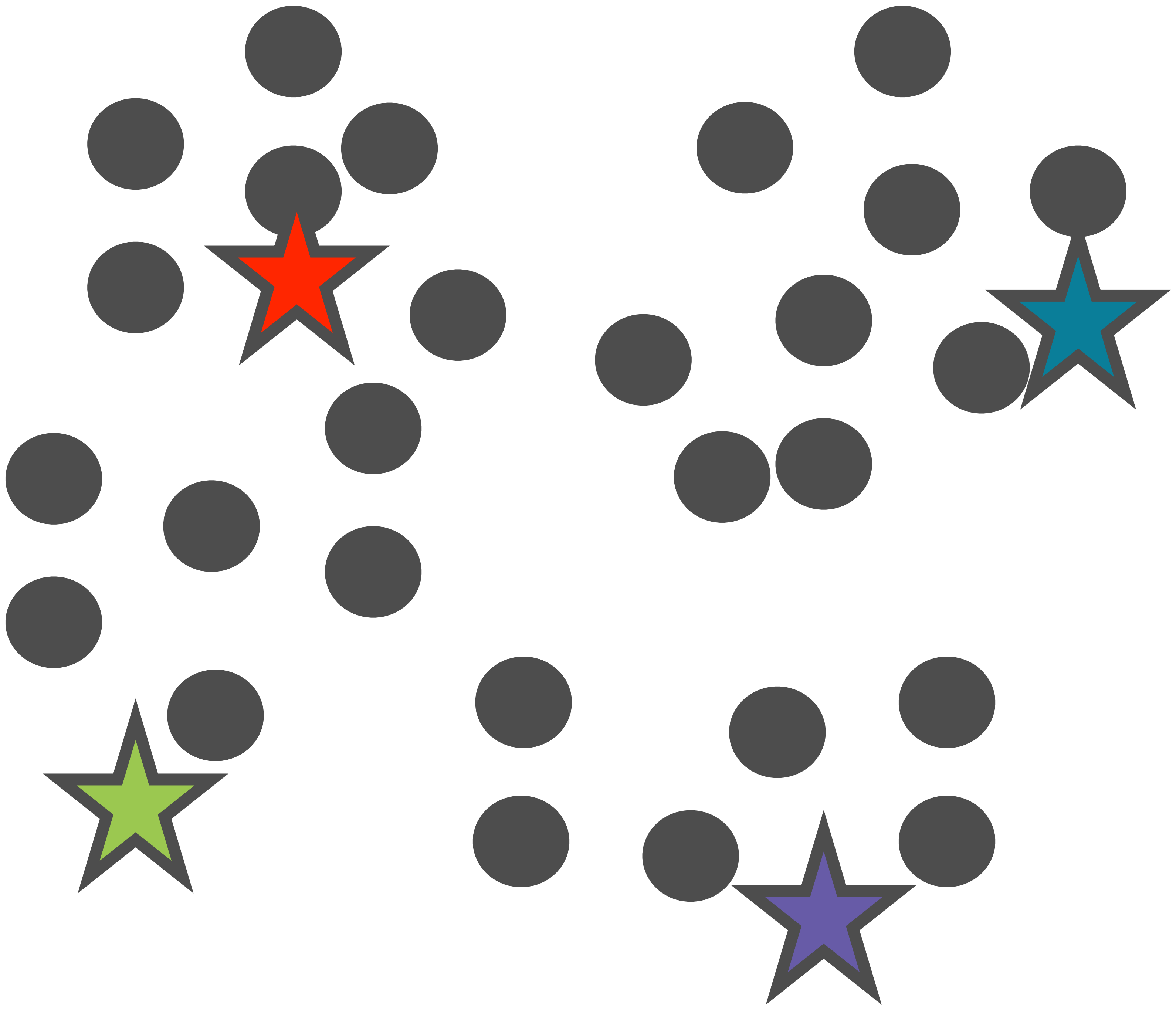
K = 3 has noticeably larger positive values

Seeds

Final reference vector values sensitive to initial values

Random initialization might not work - examine data carefully

Seeds



Seeds

Final reference vector values sensitive to initial values

Random initialization might not work - examine data carefully

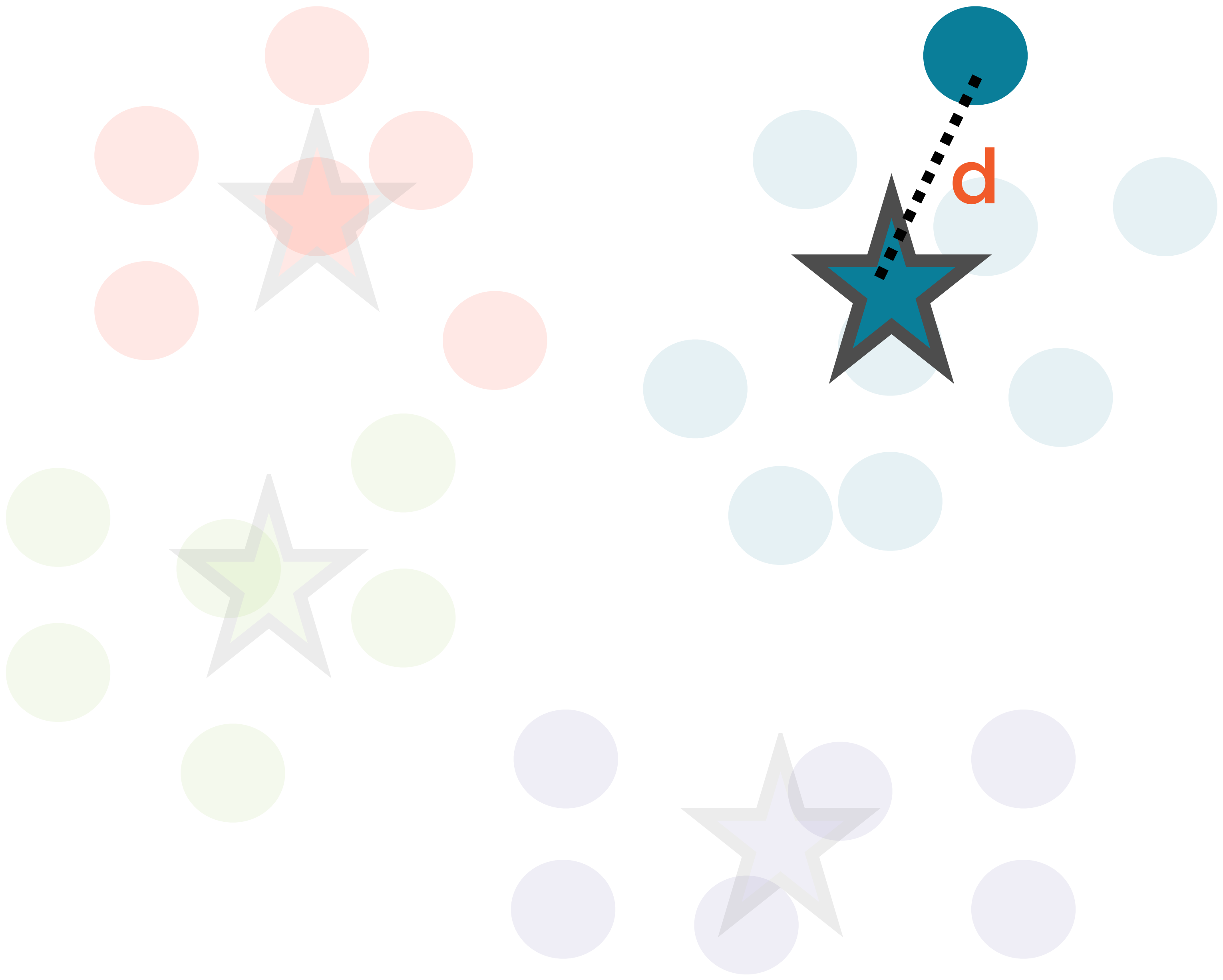
- Can perform PCA of data
- Divide range of normalized PCs into K
- Take average of each

Distance Measures

Can choose multiple distance measures:

Distance Measures

Distance from each point to the center



Distance Measures

Can choose multiple distance measures:

- Euclidean distance - centroid might not be actual data point
- Mahalanobis distance - normalize each dimension to have equal variance
- Cosine distance - cosine of angle between point and centroid

Demo

Implement k-means clustering using spark.ml

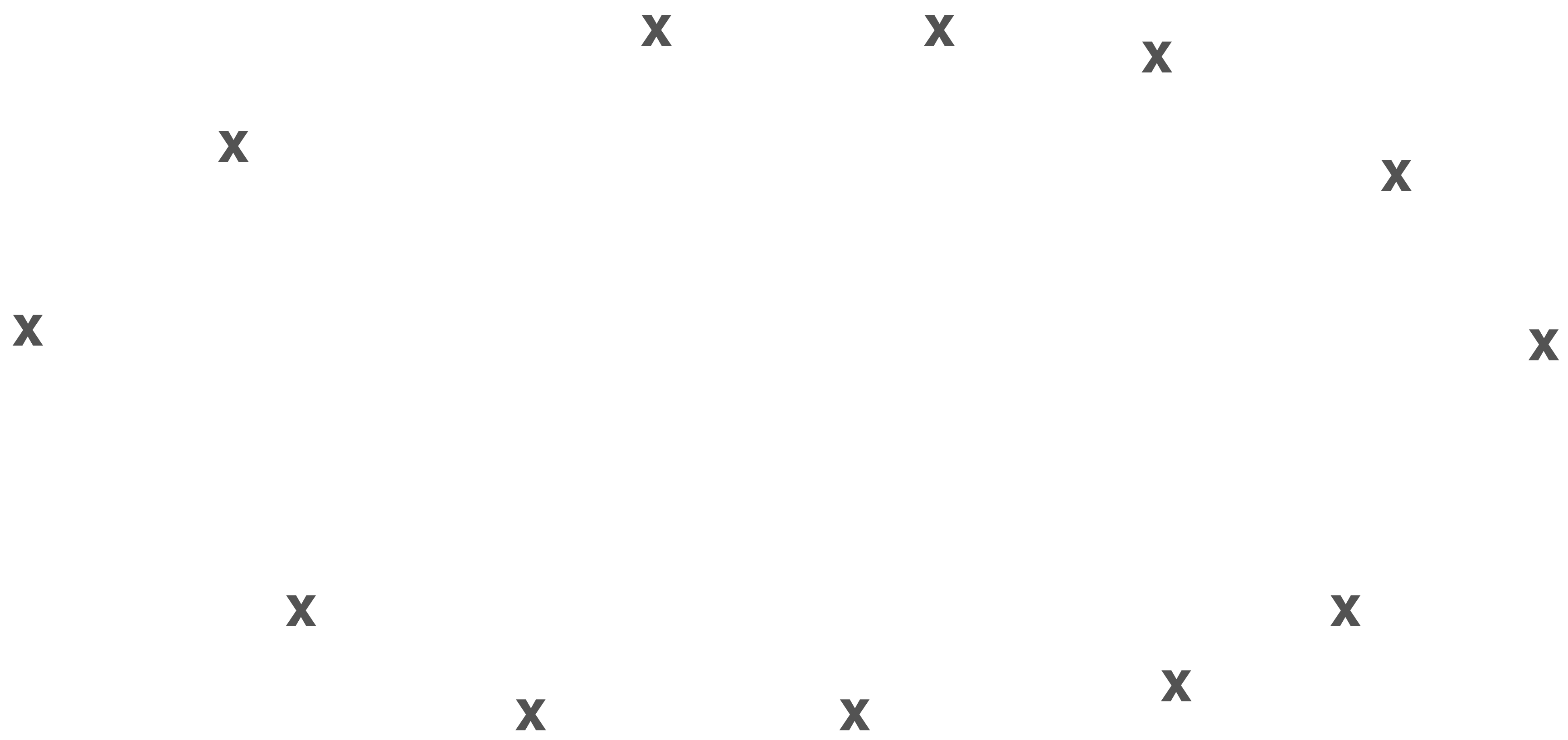
Use the silhouette method to evaluate the number of clusters chosen

Principal Components Analysis

Principal Components Analysis

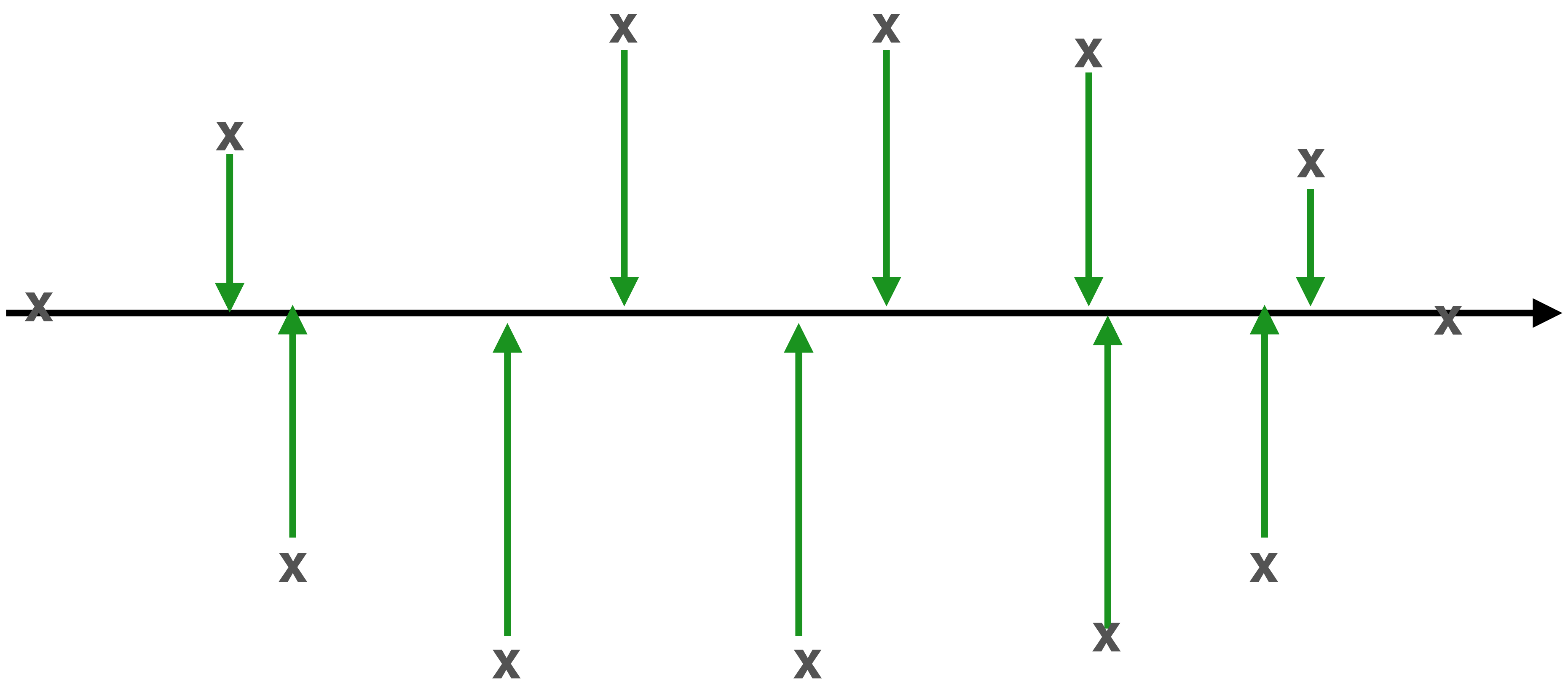
A technique to re-express complex data in terms of a few, well-chosen vectors (Principal Components) that most efficiently capture the variation in that data

Intuition Behind PCA



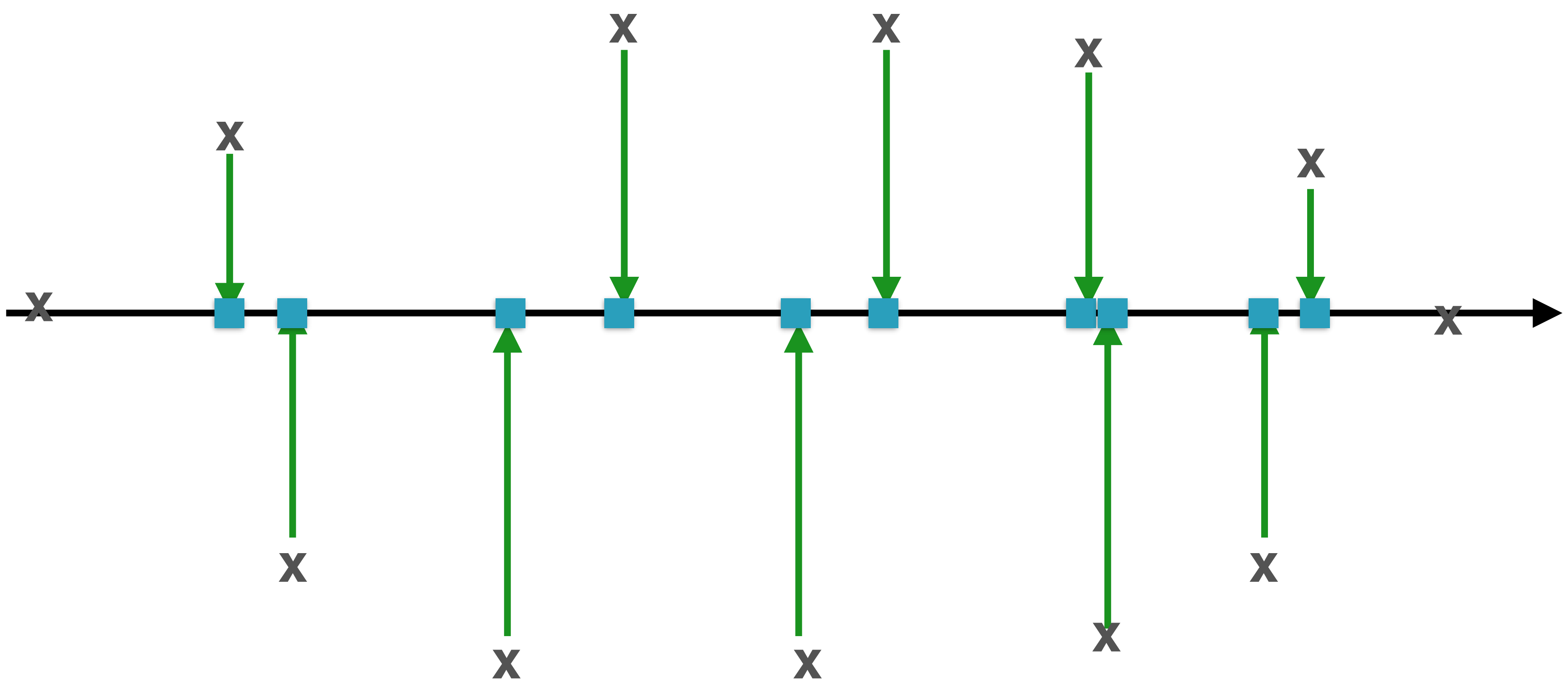
Objective: Find the “best” directions to represent this data

Intuition Behind PCA



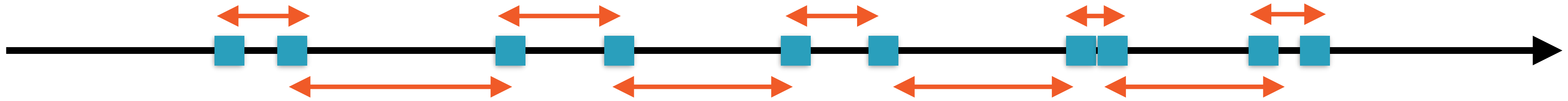
Start by "projecting" the data onto a line in some direction

Intuition Behind PCA



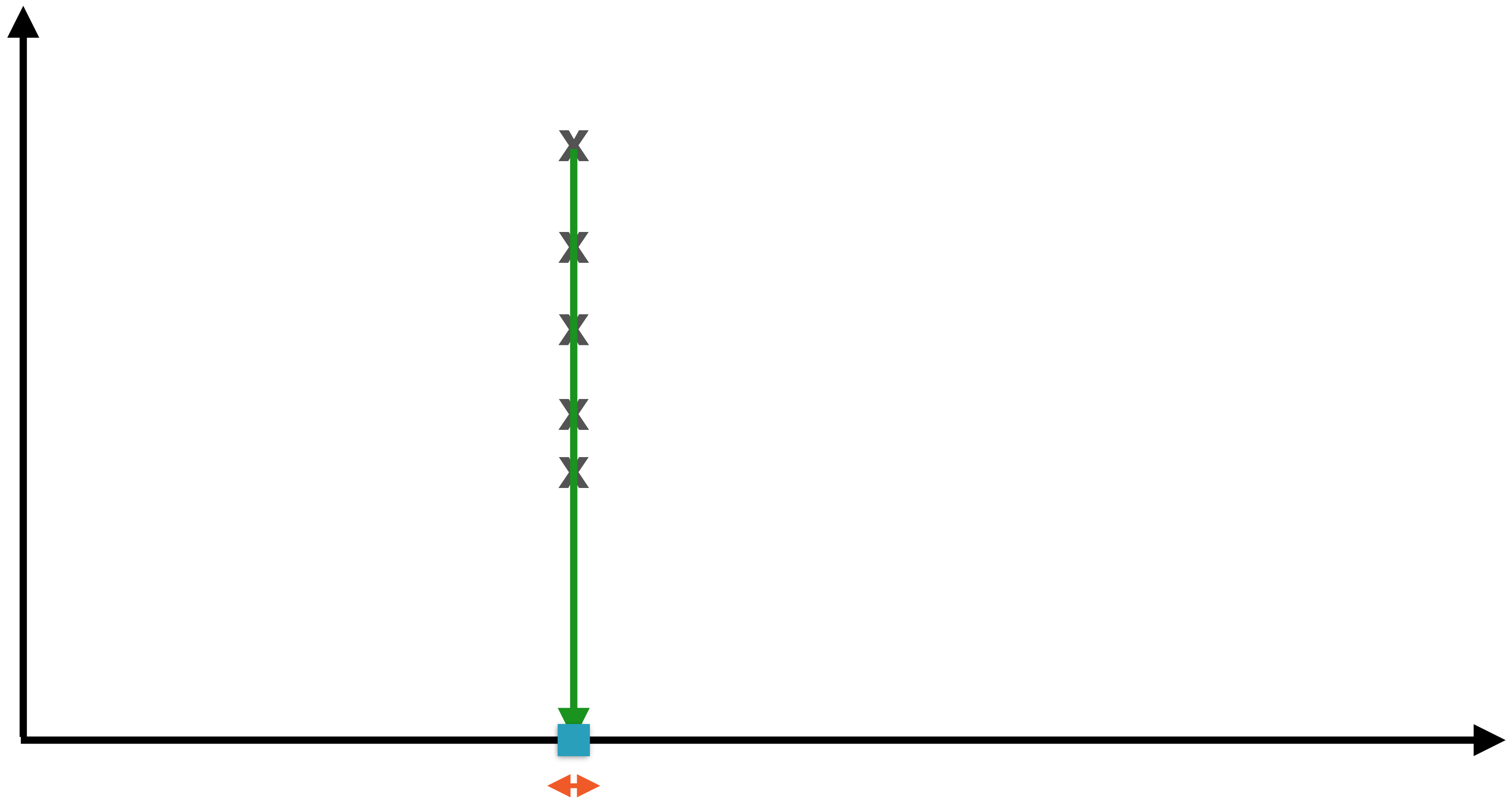
Start by "projecting" the data onto a line in some direction

Intuition Behind PCA



The greater the distances between these projections, the “better” the direction

Bad Projection

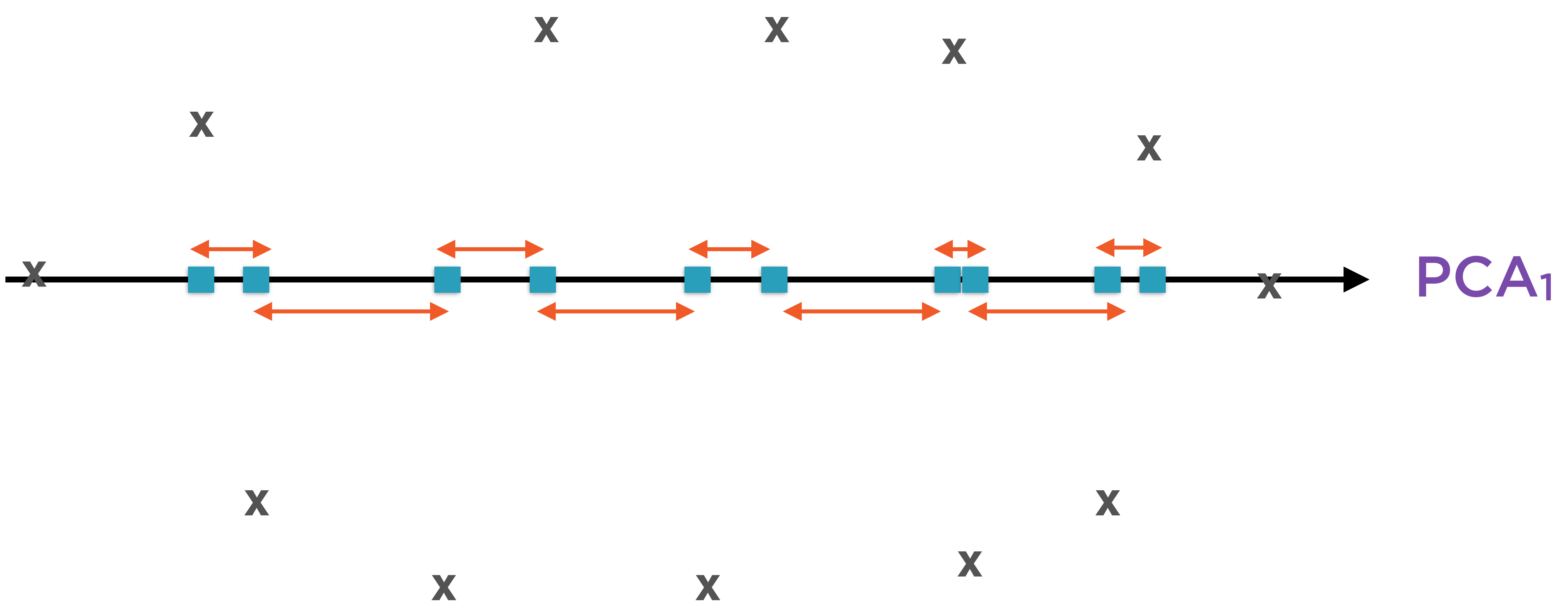


A projection where the distances are minimised is a bad one - information is lost

Good Projection

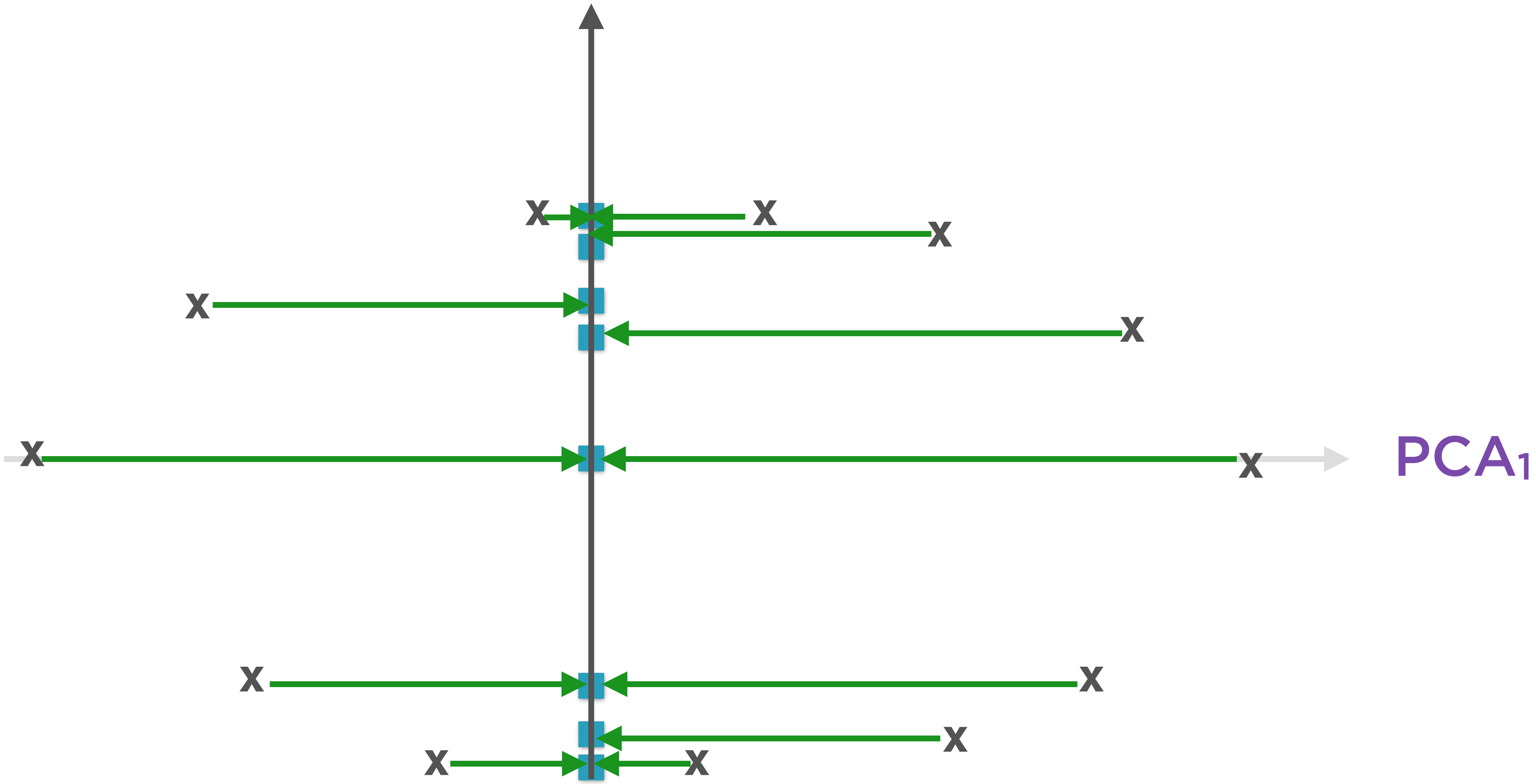
A projection where the distances are maximised is a good one - information is preserved

Intuition Behind PCA



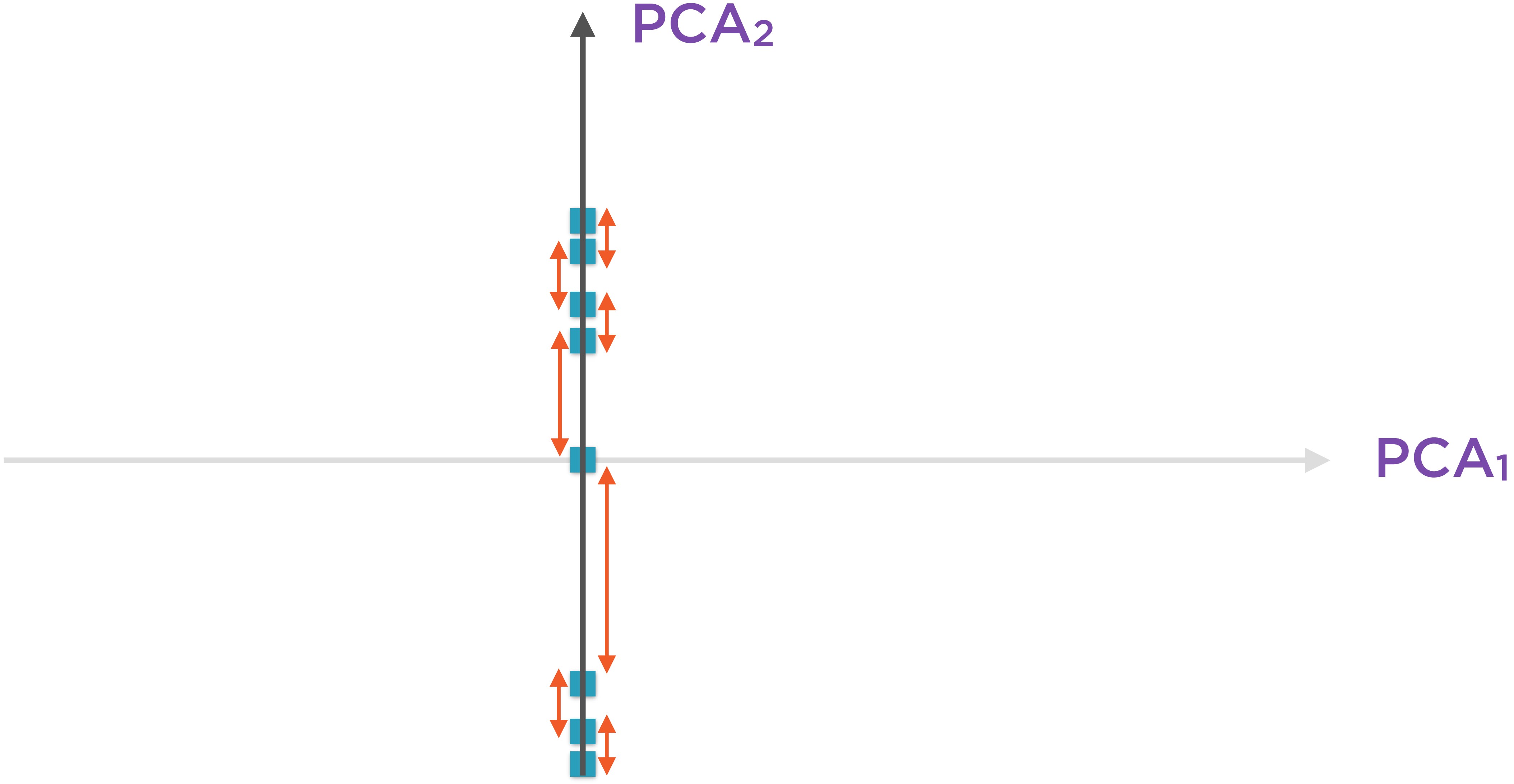
The direction along which this variance is maximised is the first principal component of the original data

Intuition Behind PCA



Find the next best direction, the **second principal component**, which must be at right angles to the first

Intuition Behind PCA

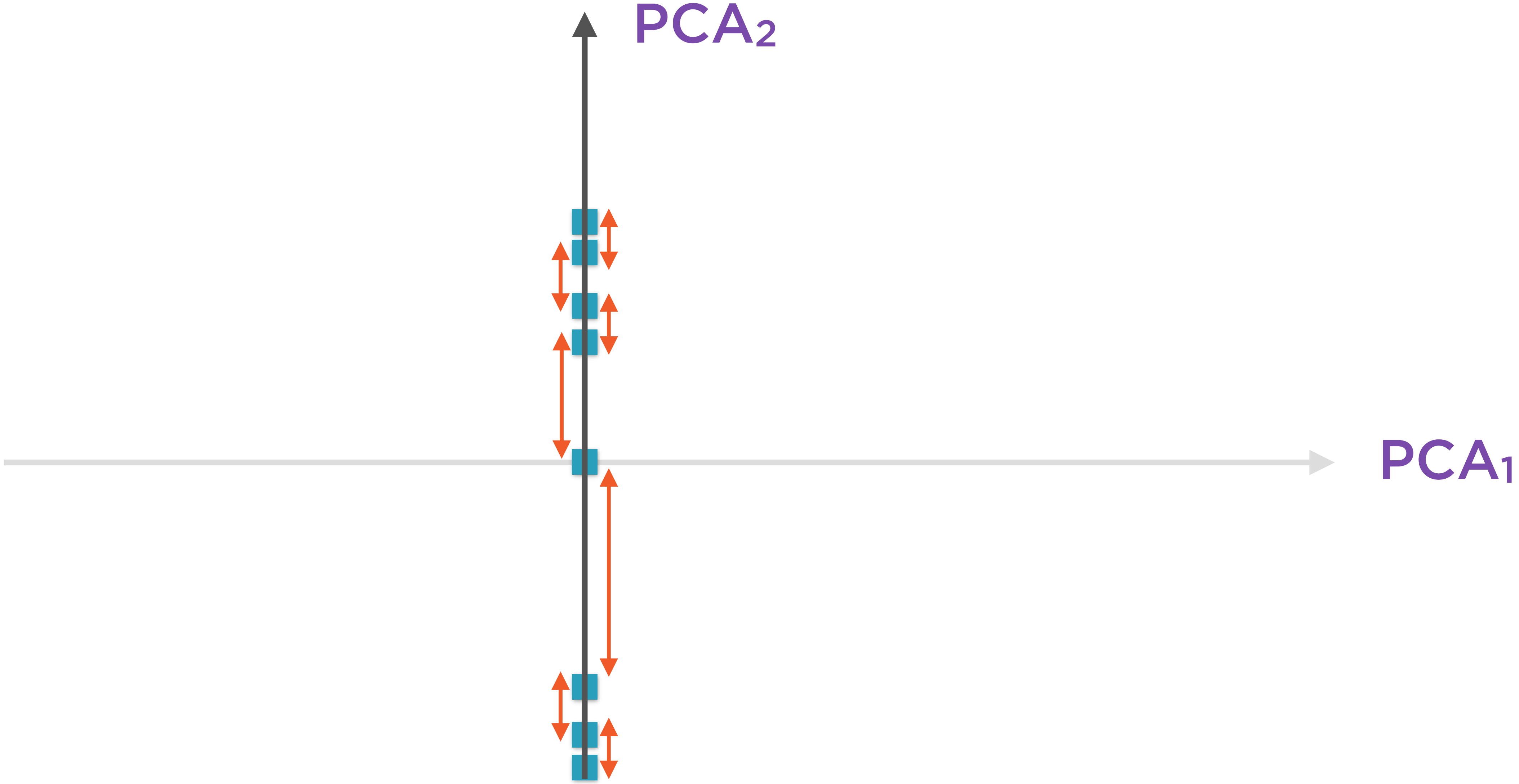


Find the next best direction, the **second principal component**, which must be at right angles to the first

Principal Components at Right Angles

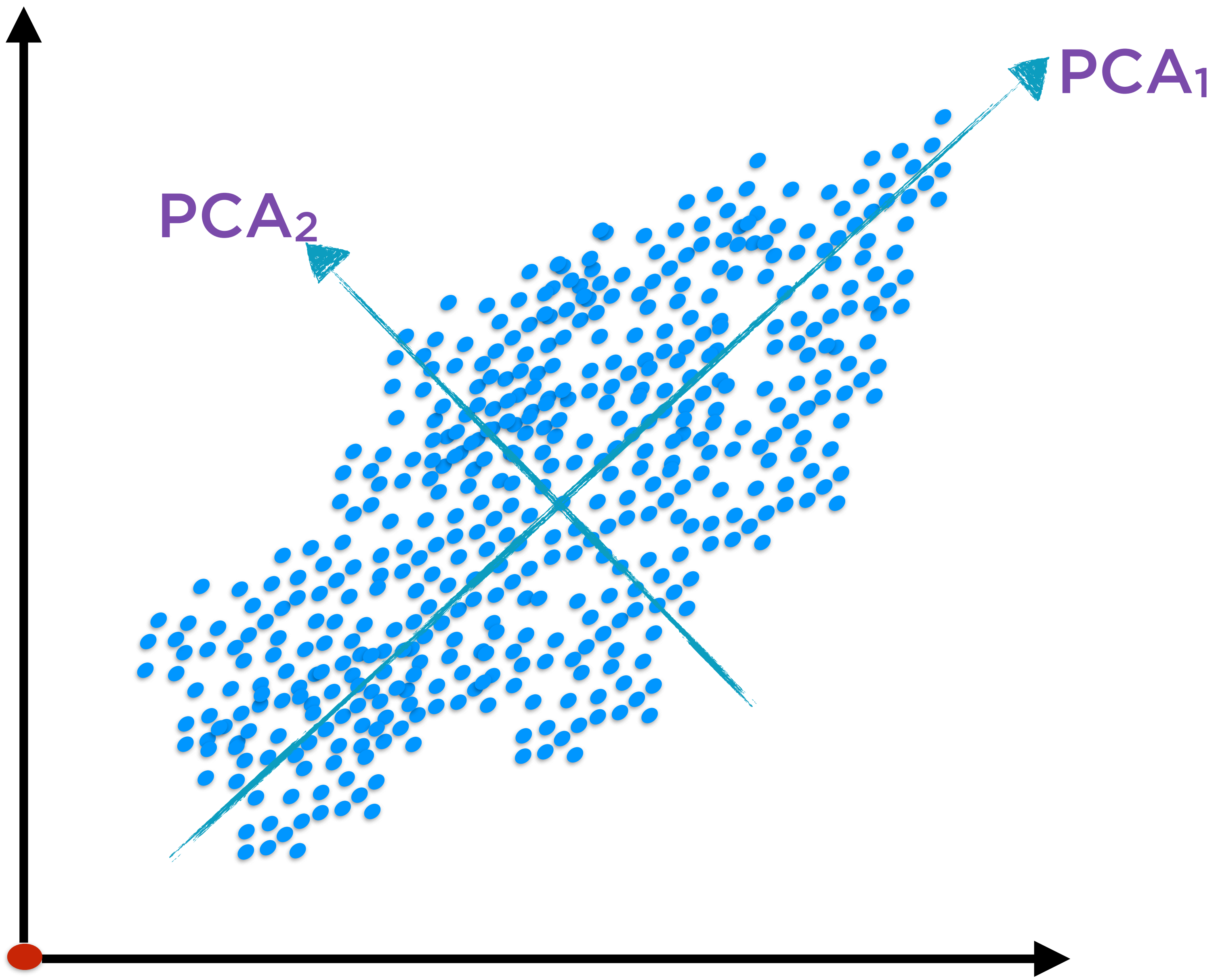
Directions at right angles help express the most variation with the smallest number of directions

Intuition Behind PCA



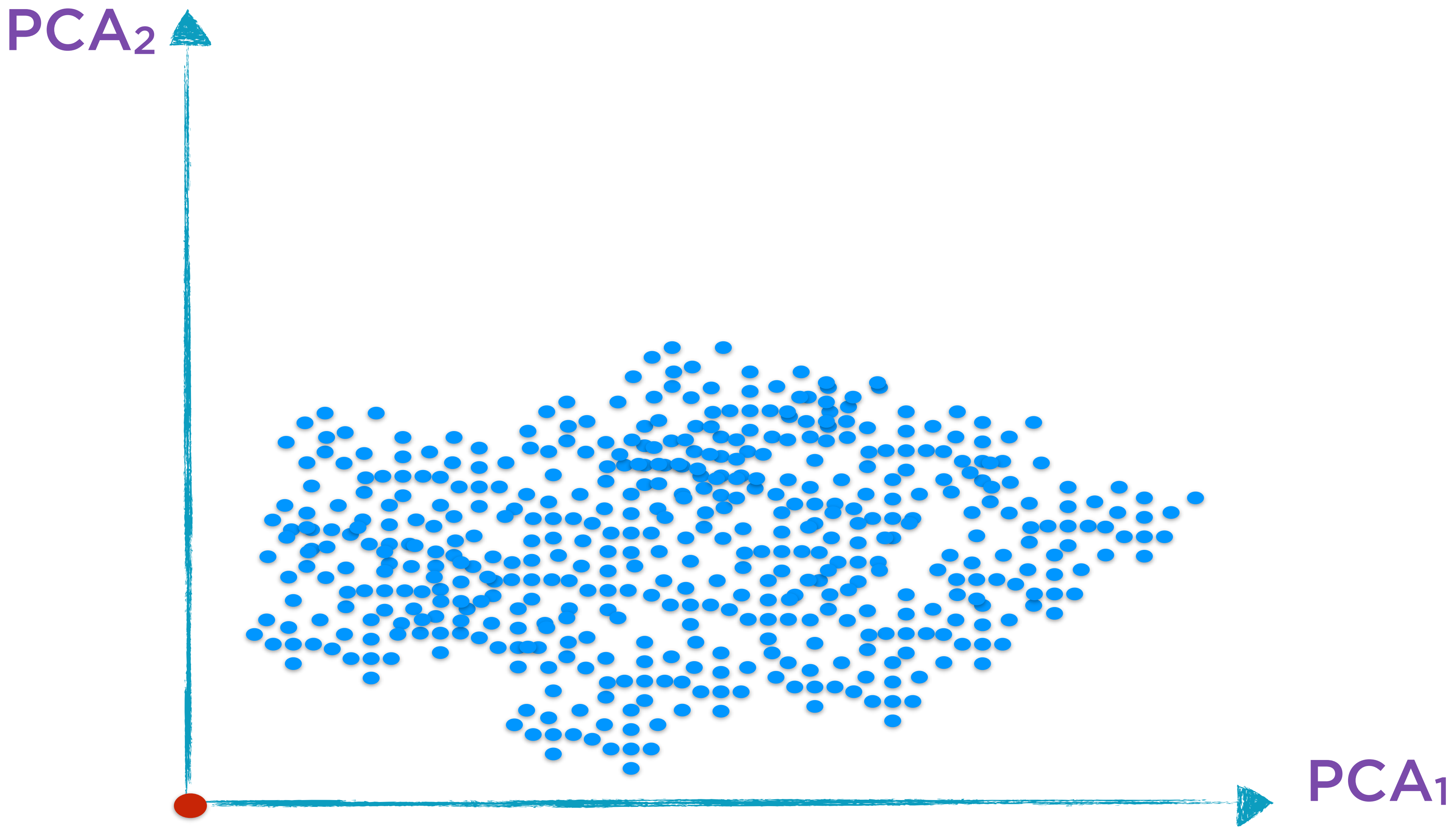
The variances are clearly smaller along this second principal component than along the first

Intuition Behind PCA



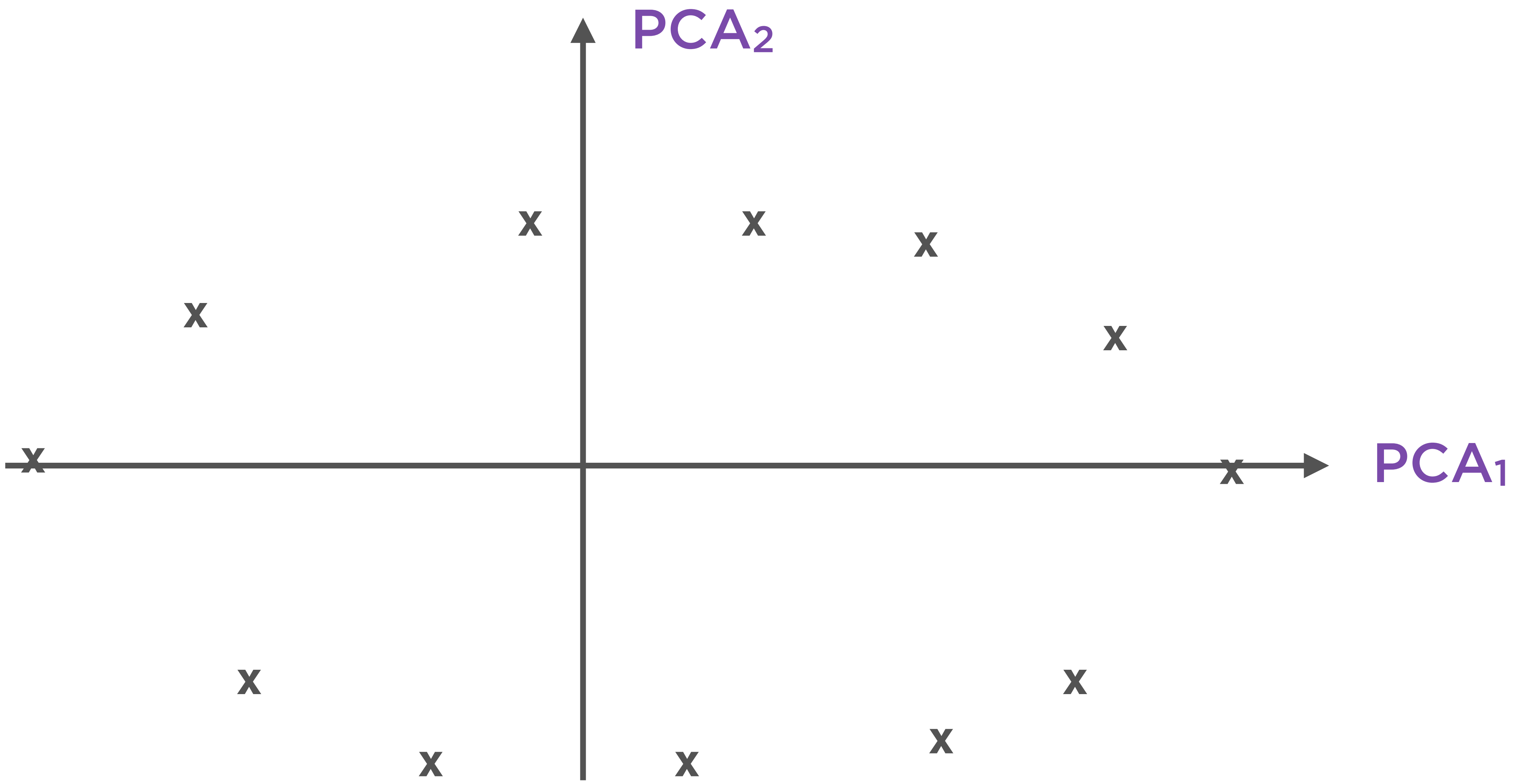
In general, there are as many principal components as there are dimensions in the original data

Intuition Behind PCA



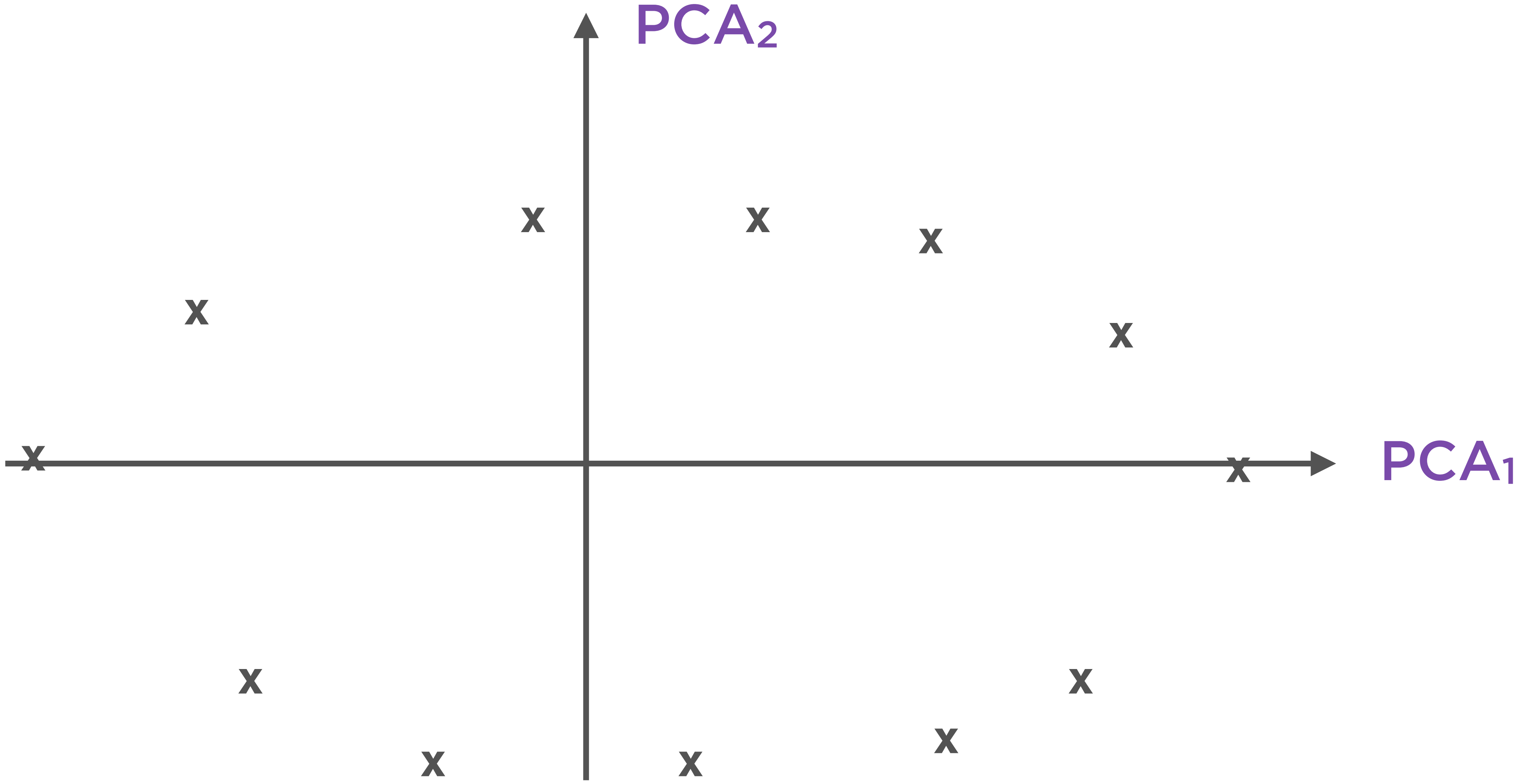
Re-orient the data along these new axes

Dimensionality Reduction



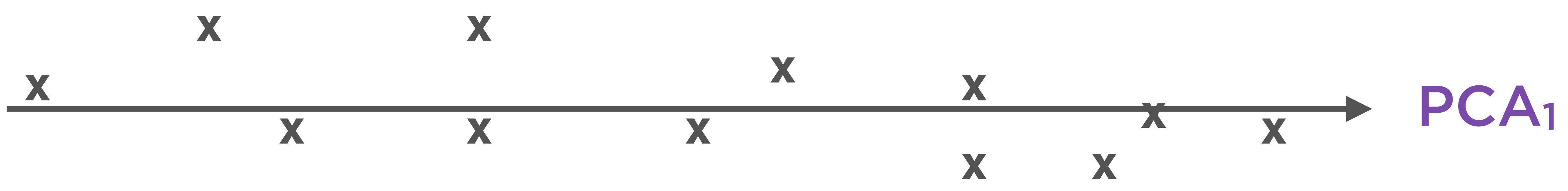
If the **variance** along the second principal component is small enough, we can just **ignore** it and use just 1 dimension to represent the data

Dimensionality Reduction



Variation along 2 dimensions: 2 principal components required

Dimensionality Reduction



Variation along 1 dimension: 1 principal component is sufficient

Principal Components Analysis

A technique to re-express complex data in terms of a few, well-chosen vectors (Principal Components) that most efficiently capture the variation in that data

Data of high dimensionality, each point represented as $(x_1, x_2 \ldots x_N)$

Principal Components Analysis

A technique to re-express complex data in terms of a few, well-chosen vectors (Principal Components) that most efficiently capture the variation in that data

Principal Components Analysis

A technique to re-express complex data in terms of a few, well-chosen vectors (Principal Components) that most efficiently capture the variation in that data

These define a smaller number of new dimensions, e.g. just two (F_1, F_2)

Express each original point
$(x_1, x_2 \ldots x_N)$ as just (f_1, f_2)

Principal Components Analysis

A technique to re-express complex data in terms of a few, well-chosen vectors (Principal Components) that most efficiently capture the variation in that data

Principal Components Analysis

A technique to re-express complex data in terms of a few, well-chosen vectors (Principal Components) that most efficiently capture the variation in that data

Very little information from the original data is lost

Principal Components Analysis

A technique to re-express complex data in terms of a few, well-chosen vectors (Principal Components) that most **efficiently** capture the variation in that data

Principal Components are a very efficient representation of the original data

Principal Components Analysis

$$\begin{bmatrix} x_{11} & & x_{1k} \\ x_{21} & & x_{2k} \\ x_{31} & \dots & x_{3k} \\ \dots & & \dots \\ x_{n1} & & x_{nk} \end{bmatrix} n$$

k

Original Data

PCA

$$\begin{bmatrix} f_{11} & & f_{1k} \\ f_{21} & & f_{2k} \\ f_{31} & \dots & f_{3k} \\ \dots & & \dots \\ f_{n1} & & f_{nk} \end{bmatrix} n$$

k

Principal Components

Same number of columns

Dimensionality Reduction

$$\begin{bmatrix} f_{11} & & f_{1k} \\ f_{21} & & f_{2k} \\ f_{31} & \cdots & f_{3k} \\ \cdots & & \cdots \\ f_{n1} & & f_{nk} \end{bmatrix} \xrightarrow{\text{Choose first 2 principal components}} \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \\ f_{31} & f_{32} \\ \cdots & \cdots \\ f_{n1} & f_{n2} \end{bmatrix}$$

Principal Components

Reconstruct Original Data

$$\underbrace{\begin{bmatrix} f_{11} & & f_{1k} \\ f_{21} & & f_{2k} \\ f_{31} & \cdots & f_{3k} \\ \cdots & & \cdots \\ f_{n1} & & f_{nk} \end{bmatrix}}_{k} n \; \times \; \underbrace{\begin{bmatrix} w_{11} & & w_{1k} \\ w_{21} & & w_{2k} \\ w_{31} & \cdots & w_{3k} \\ \cdots & & \cdots \\ w_{kk} & & w_{kk} \end{bmatrix}}_{k} k \; = \; \underbrace{\begin{bmatrix} x_{11} & & x_{1k} \\ x_{21} & & x_{2k} \\ x_{31} & \cdots & x_{3k} \\ \cdots & & \cdots \\ x_{n1} & & x_{nk} \end{bmatrix}}_{k} n$$

Principal Components

Weight Vectors

Original Data

Demo

Implement principal components analysis in spark.ml

Summary

Unsupervised learning is used to find patterns within the data itself

K-means clustering is a popular technique to find logical groupings of data

Elbow and silhouette methods are used to find the best value of hyperparameter k

Dimensionality reduction is used to discover latent factors in underlying data

PCA is very commonly used method for dimensionality reduction