Implementing Clustering and Dimensionality Reduction in scikit-learn

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Clustering is an unsupervised learning technique which helps find patterns in data

Common clustering algorithms are k-means, mean-shift clustering

Dimensionality reduction represents inputs in terms of their most significant features

PCA is a very commonly used technique for latent factor analysis

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

Types of ML Algorithms

Supervised

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

Unsupervised

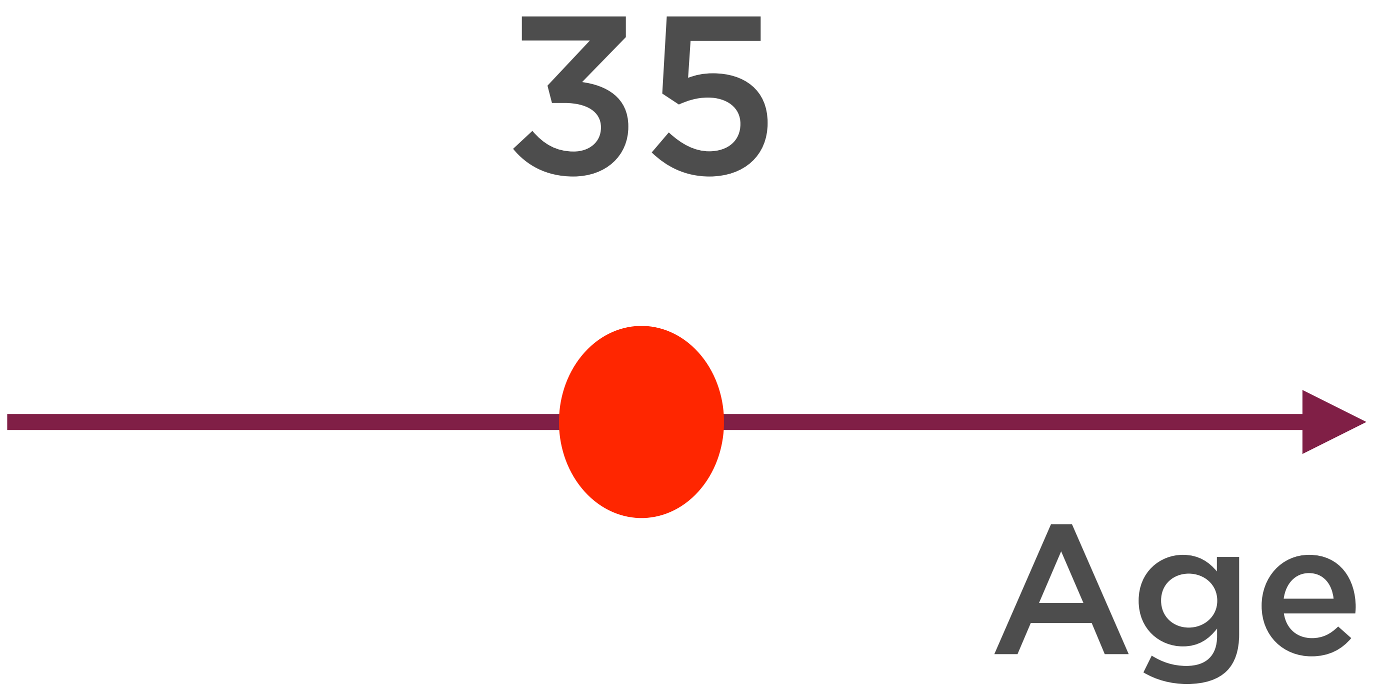
The model has to be set up right to learn structure in the data

Clustering

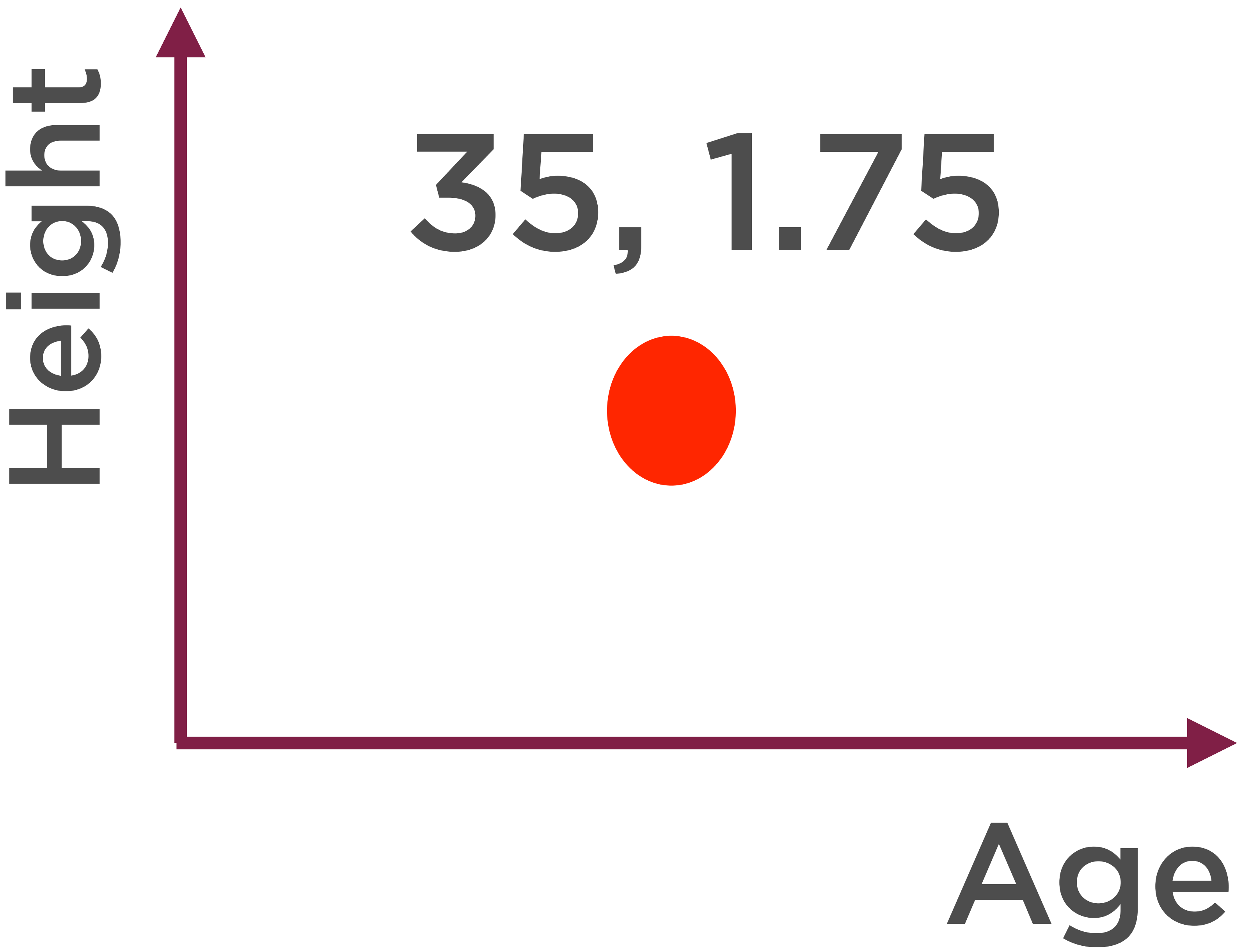
Clustering

Anything can be represented by a set of numbers

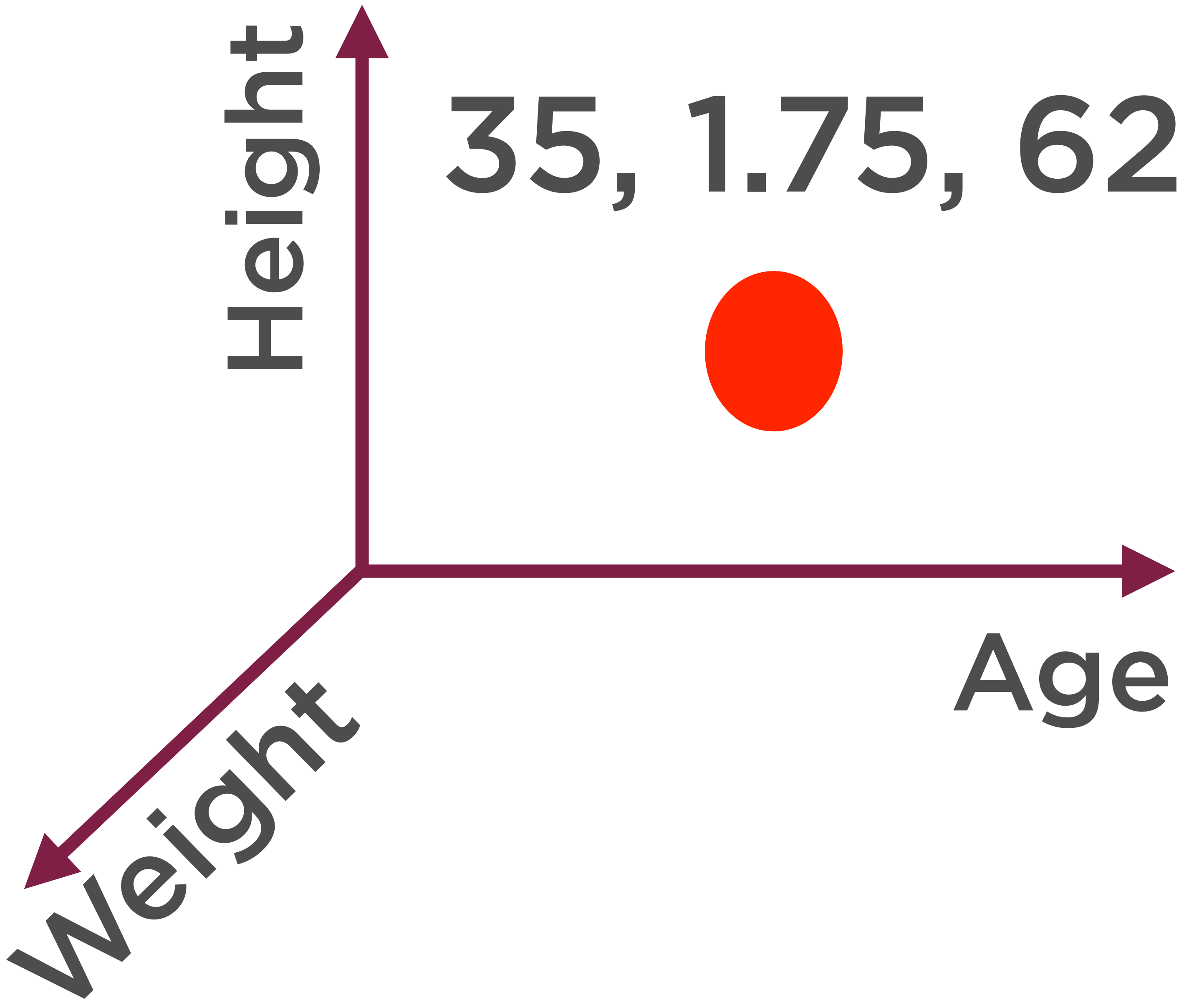
Age, Height, Weight



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

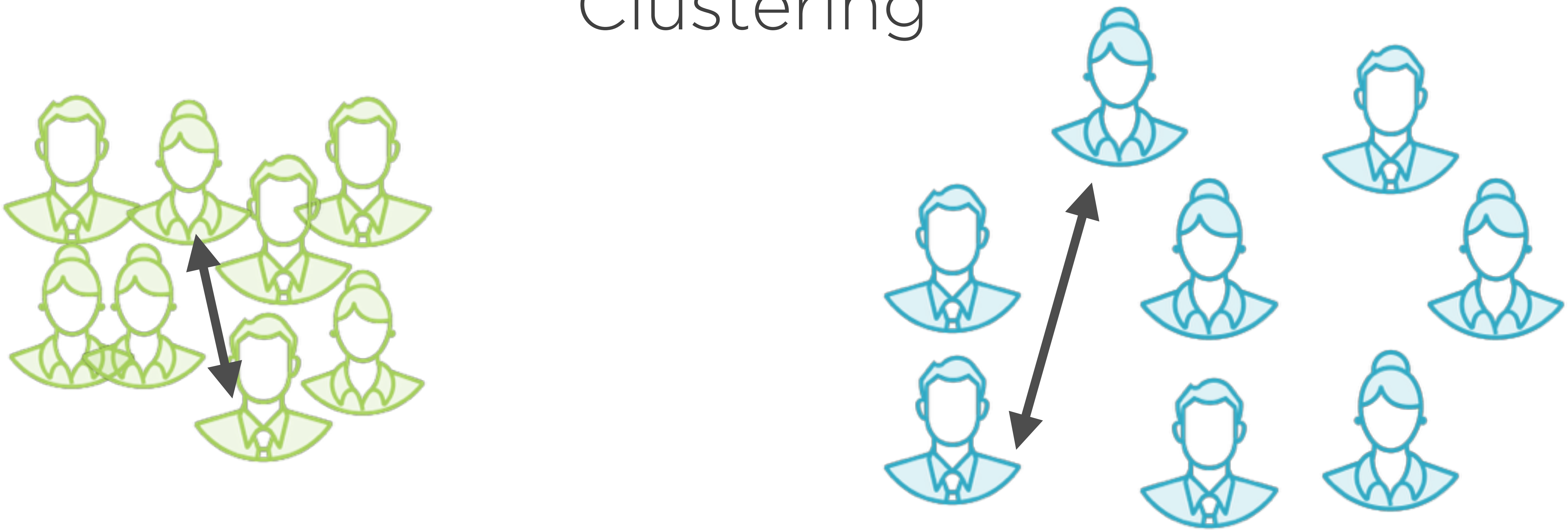
Different group = different

Clustering

Same group = similar

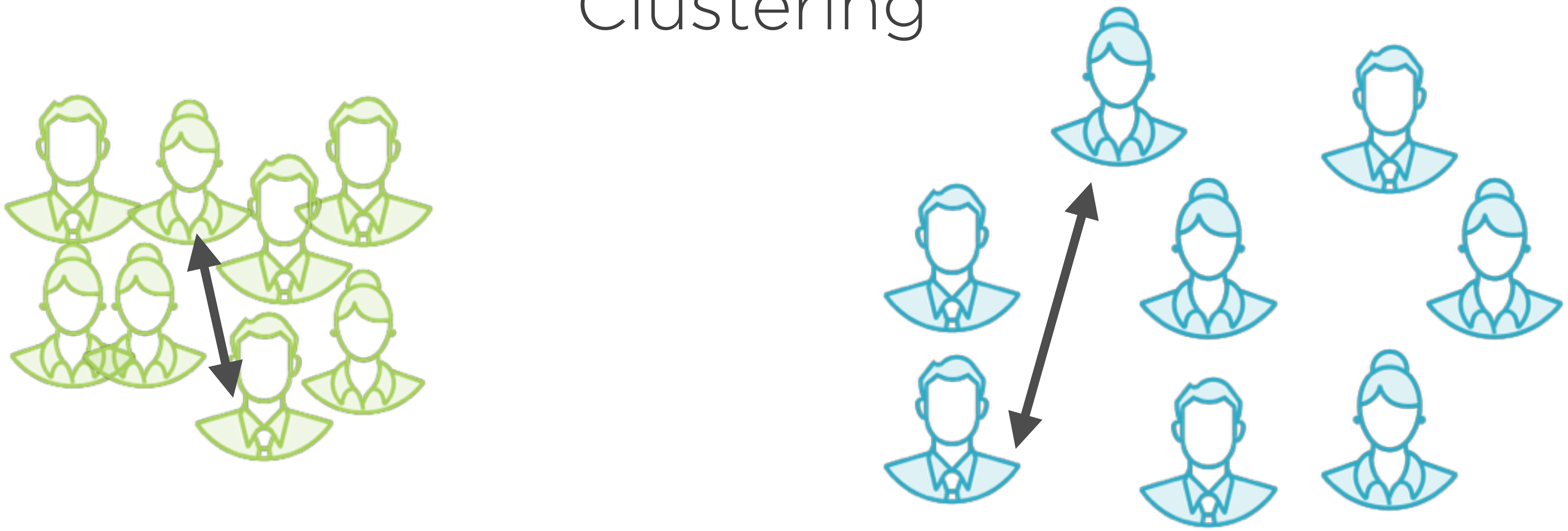
Different group = different

Clustering



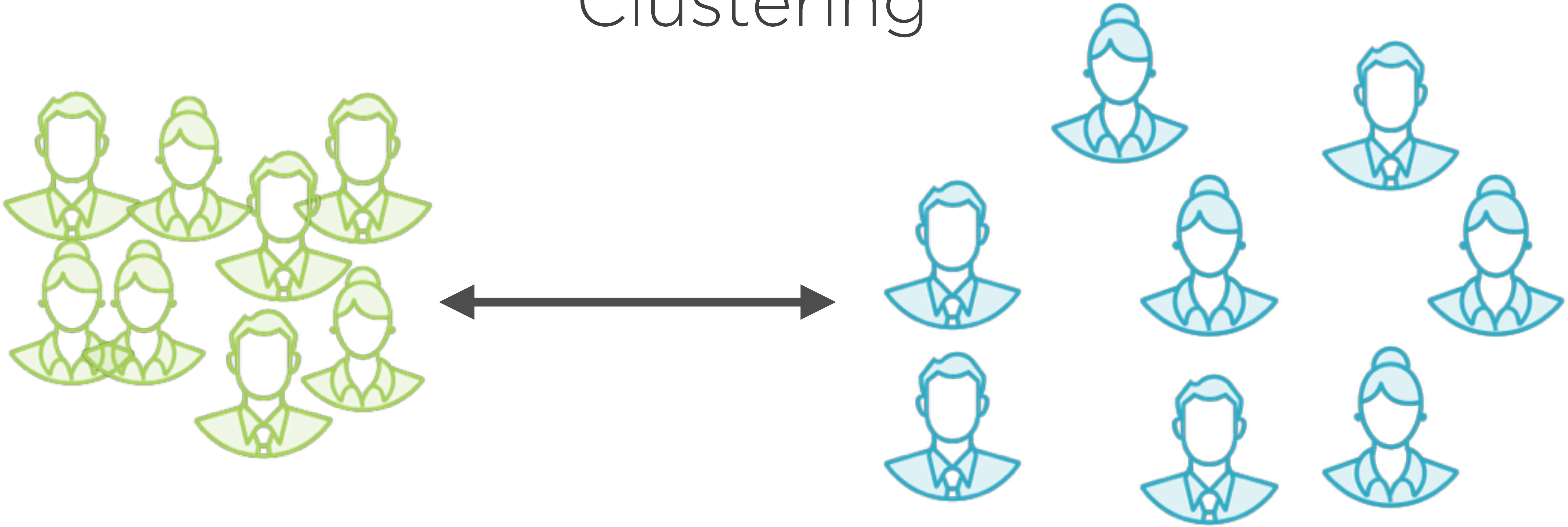
The **distance** between users in a cluster indicates how **similar** they are

Clustering



Maximize intra-cluster similarity

Clustering



Minimize inter-cluster similarity

Clustering Objective

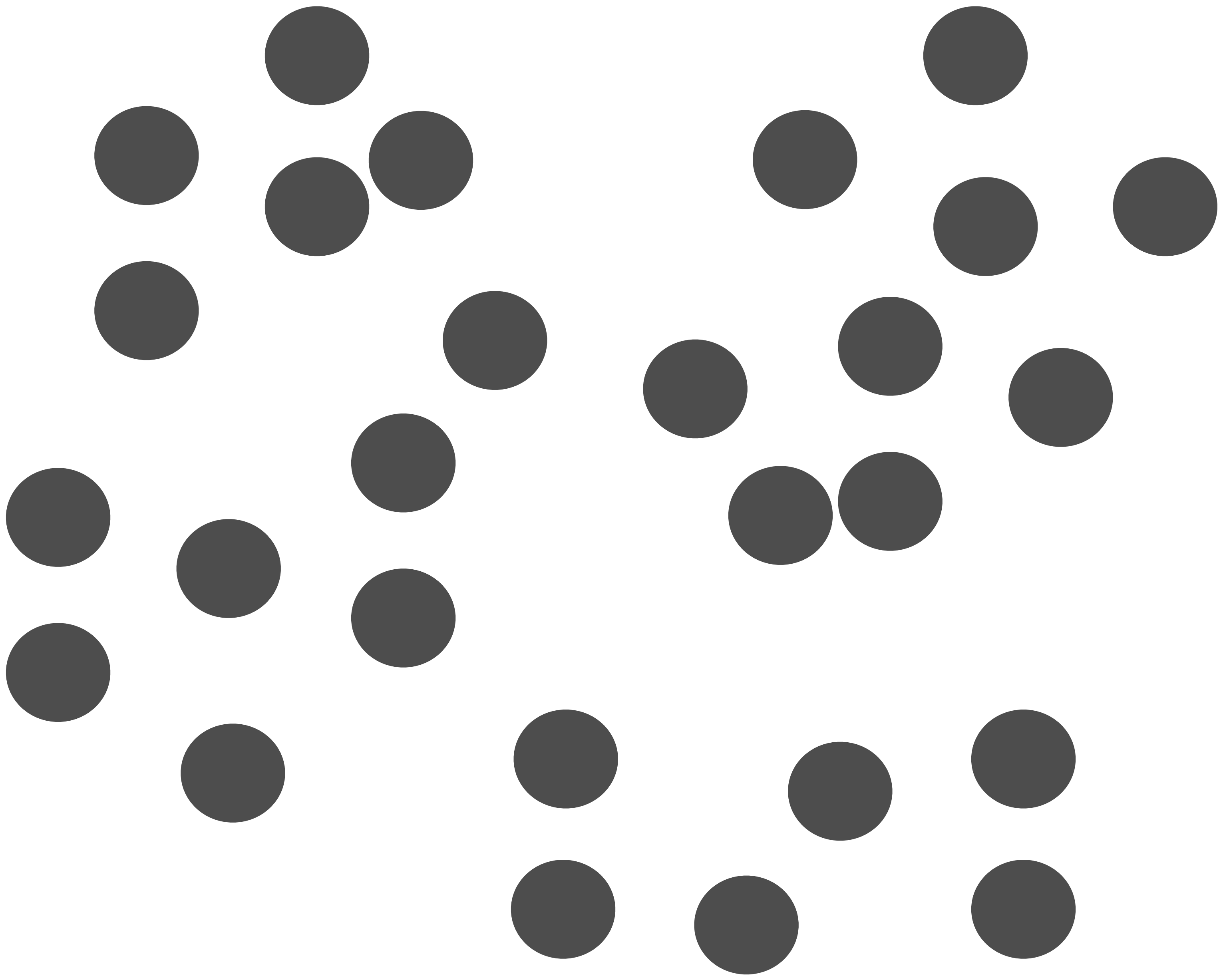
Maximize intra-cluster similarity

Minimize inter-cluster similarity

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

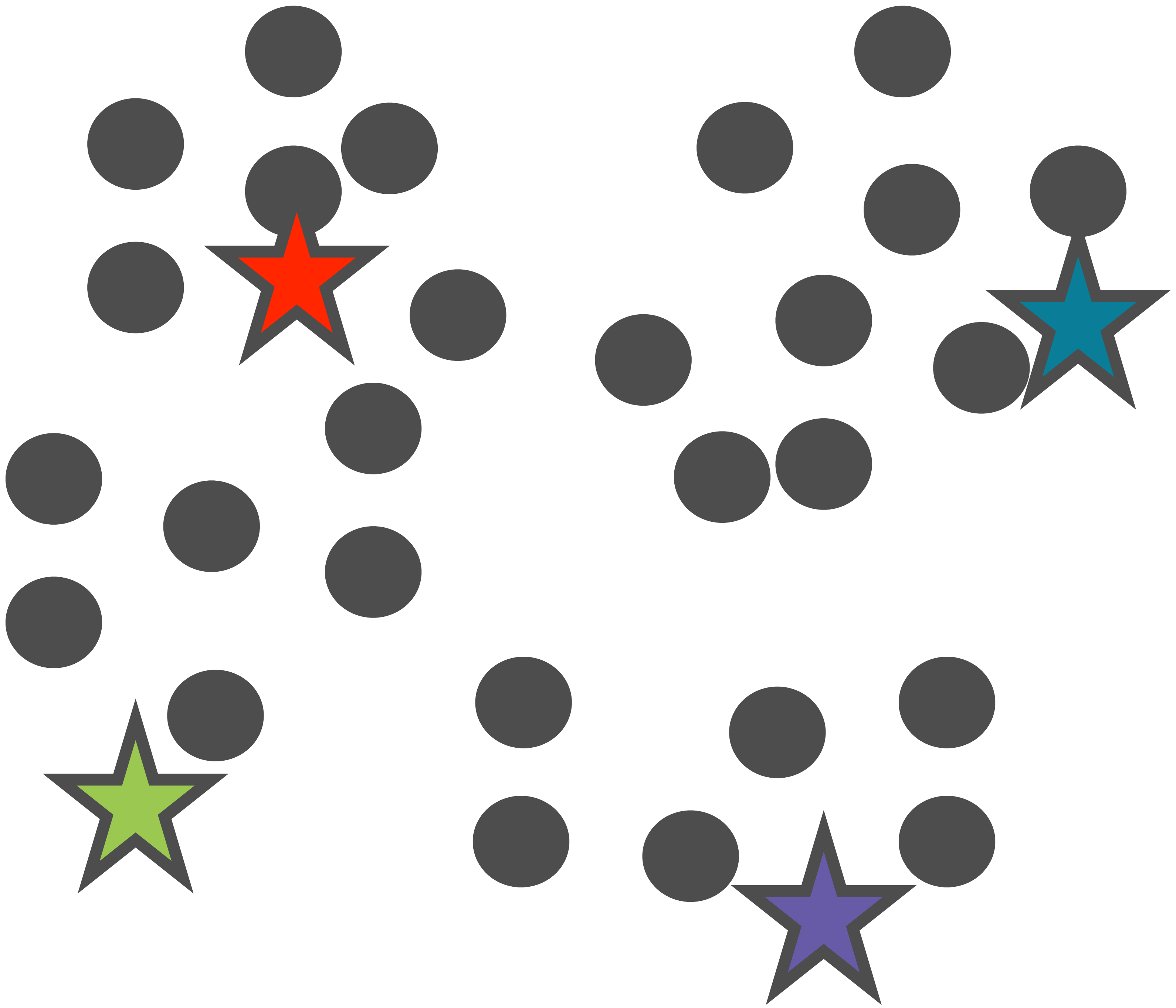
K-Means Clustering

Initialize K centroids i.e. means



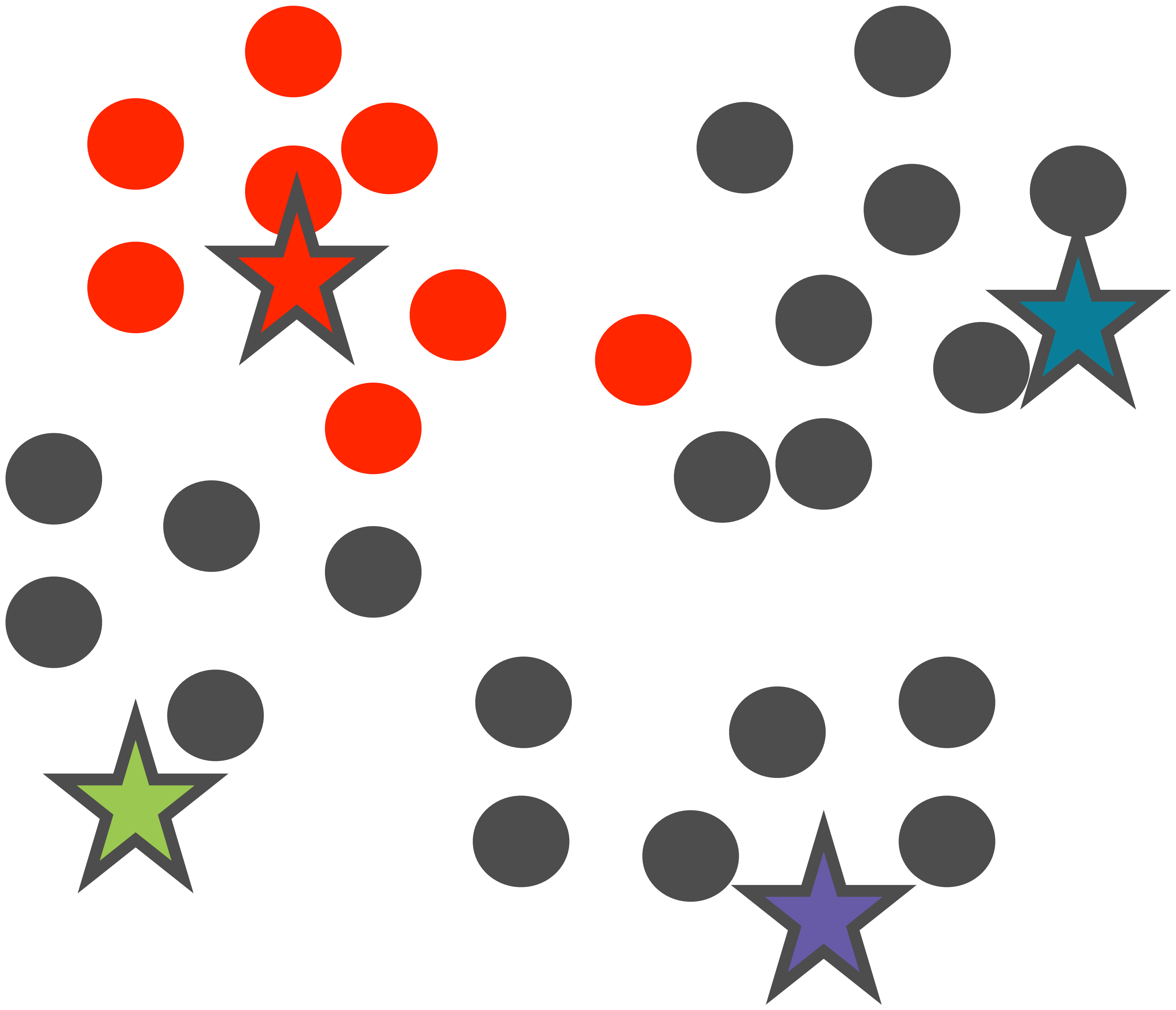
K-Means Clustering

Assign each point to a cluster



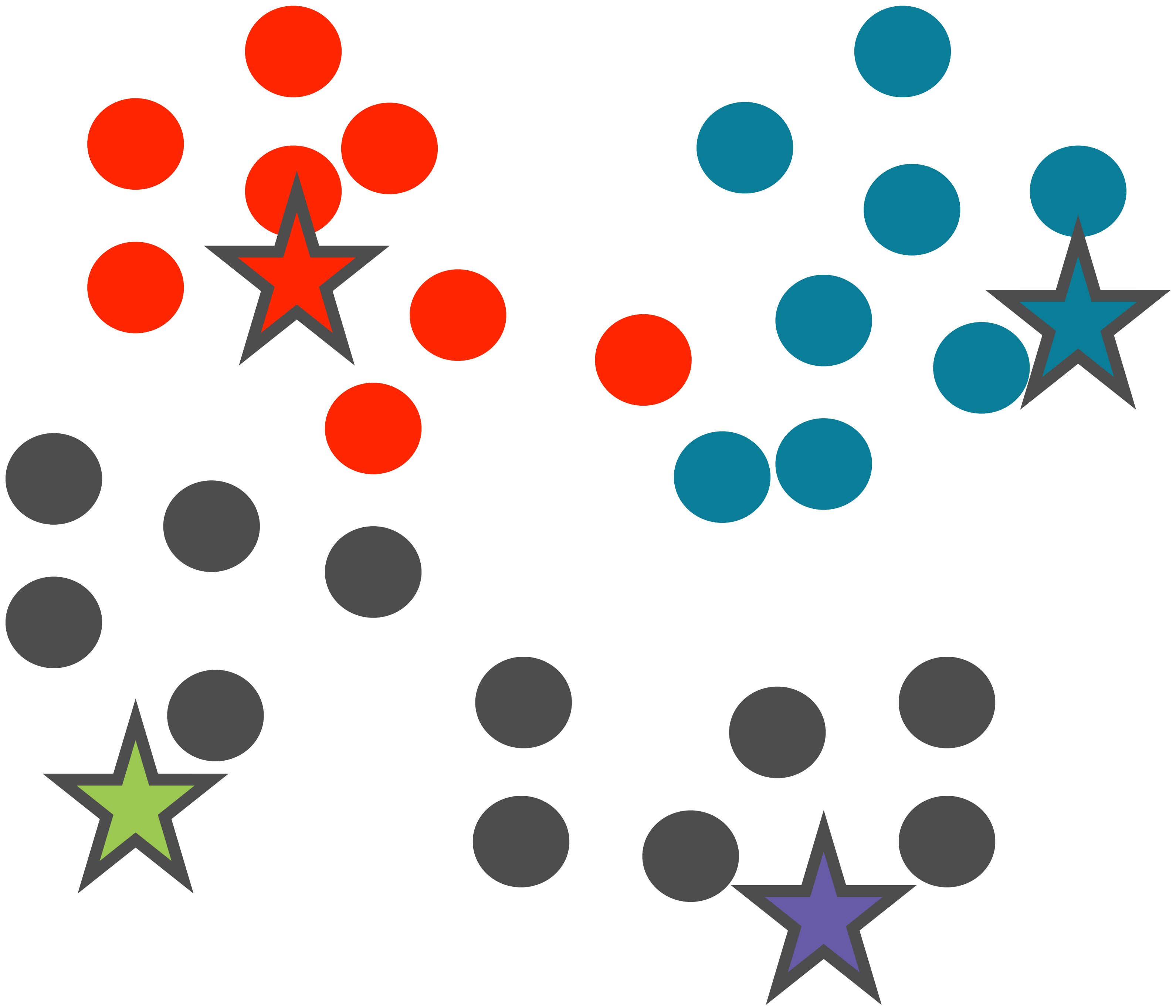
K-Means Clustering

Assign each point to a cluster



K-Means Clustering

Assign each point to a cluster



K-Means Clustering

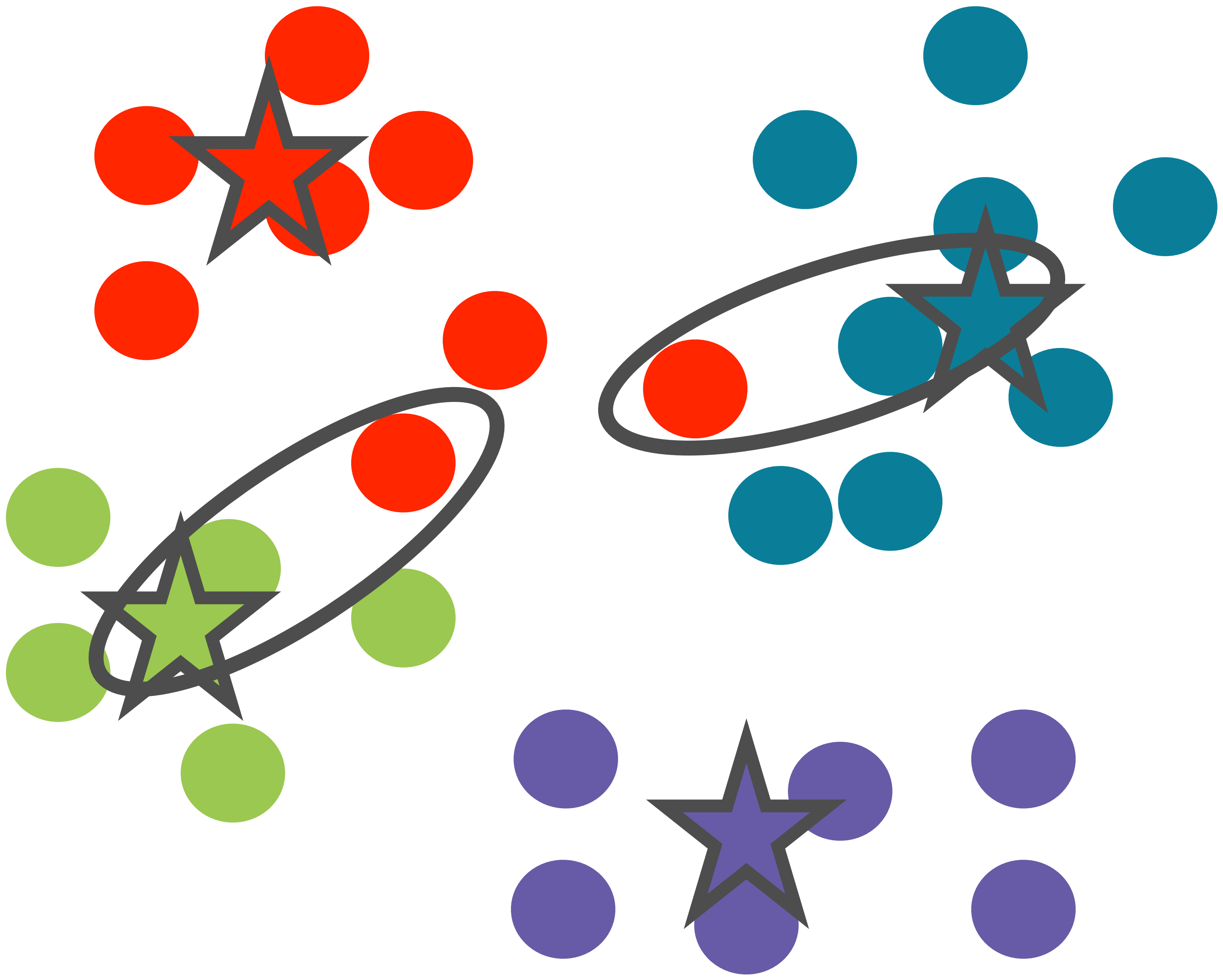
Assign each point to a cluster

K-Means Clustering

Recalculate the mean for each cluster

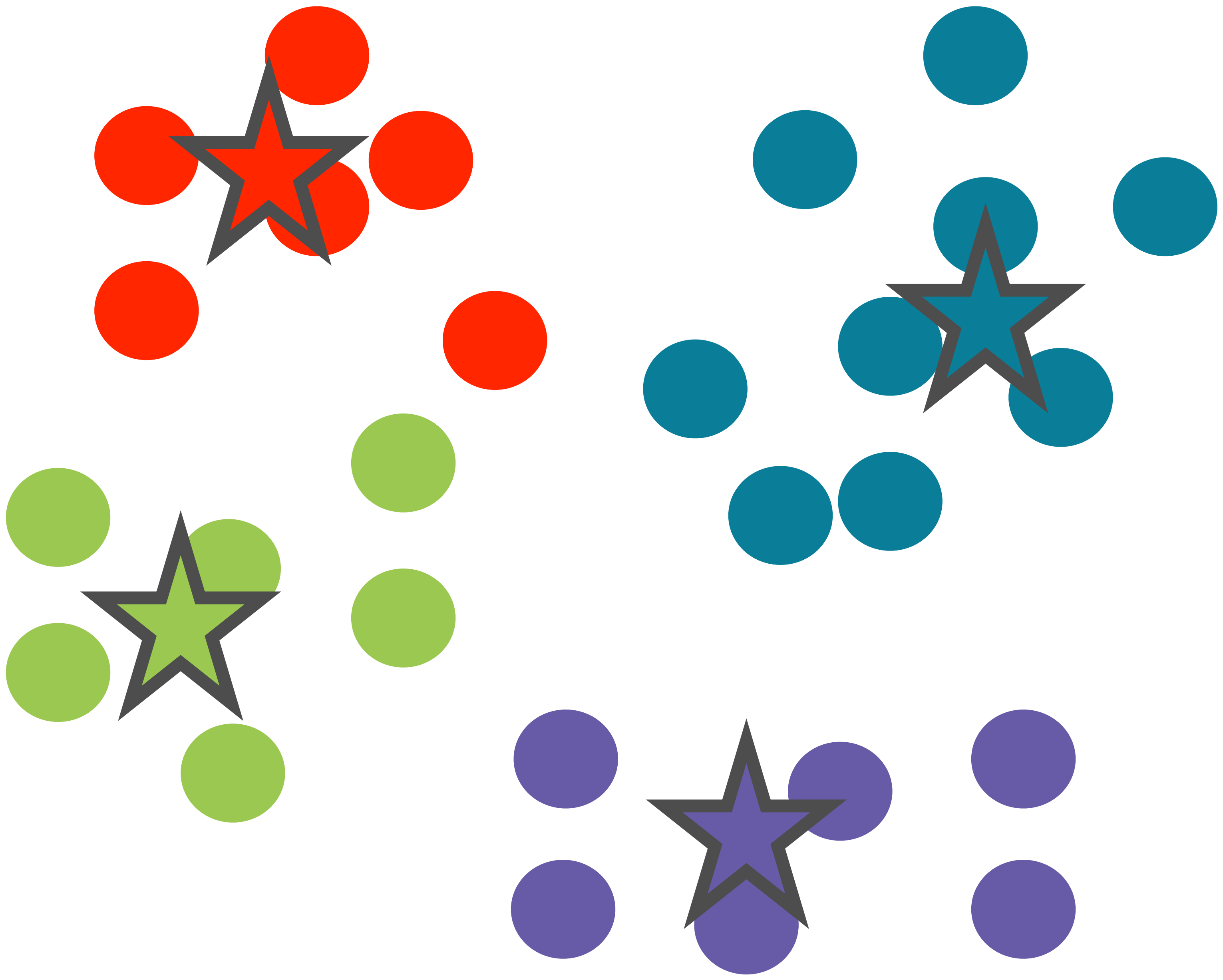
K-Means Clustering

Re-assign the points to clusters

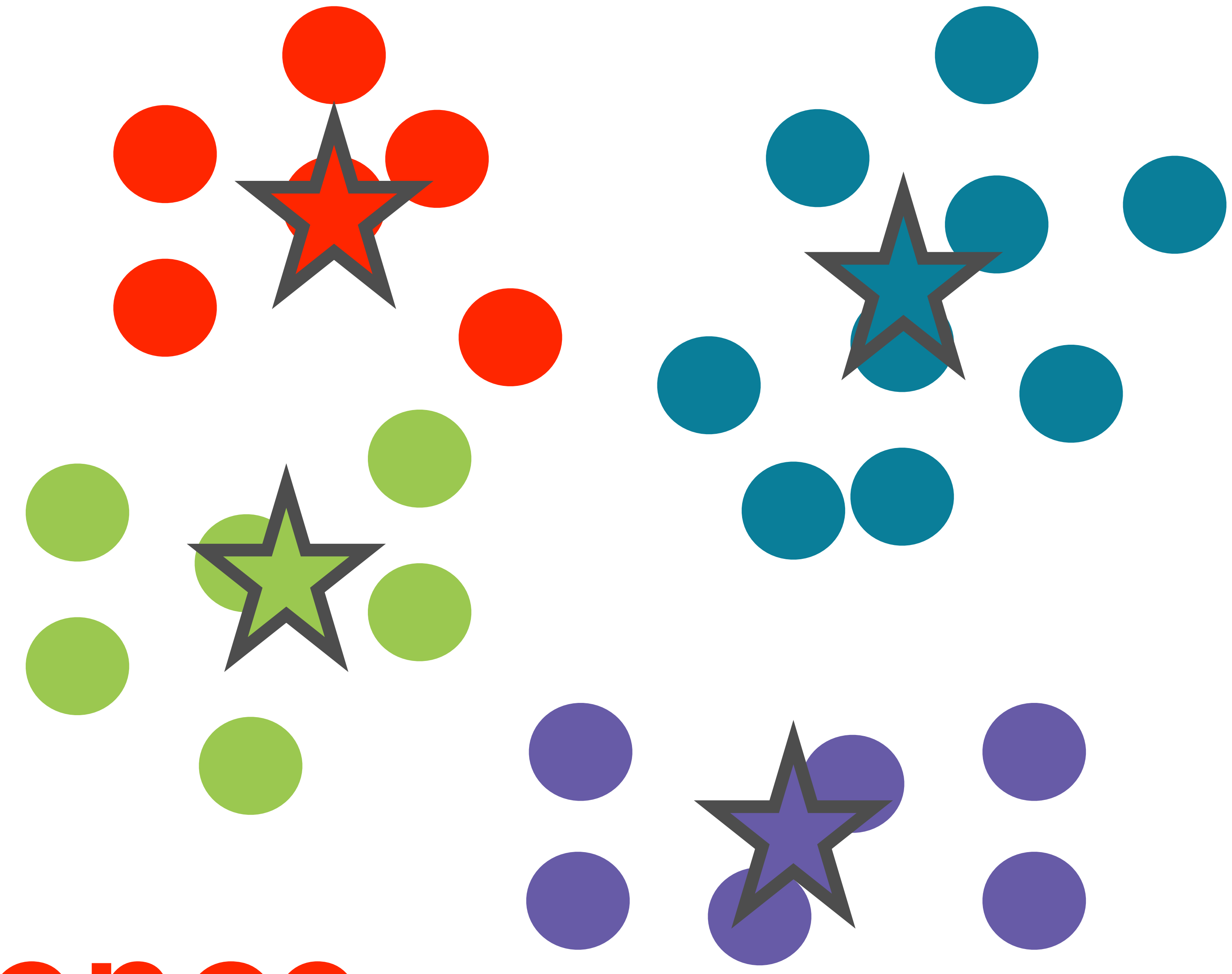


K-Means Clustering

Iterate until points are in their final clusters

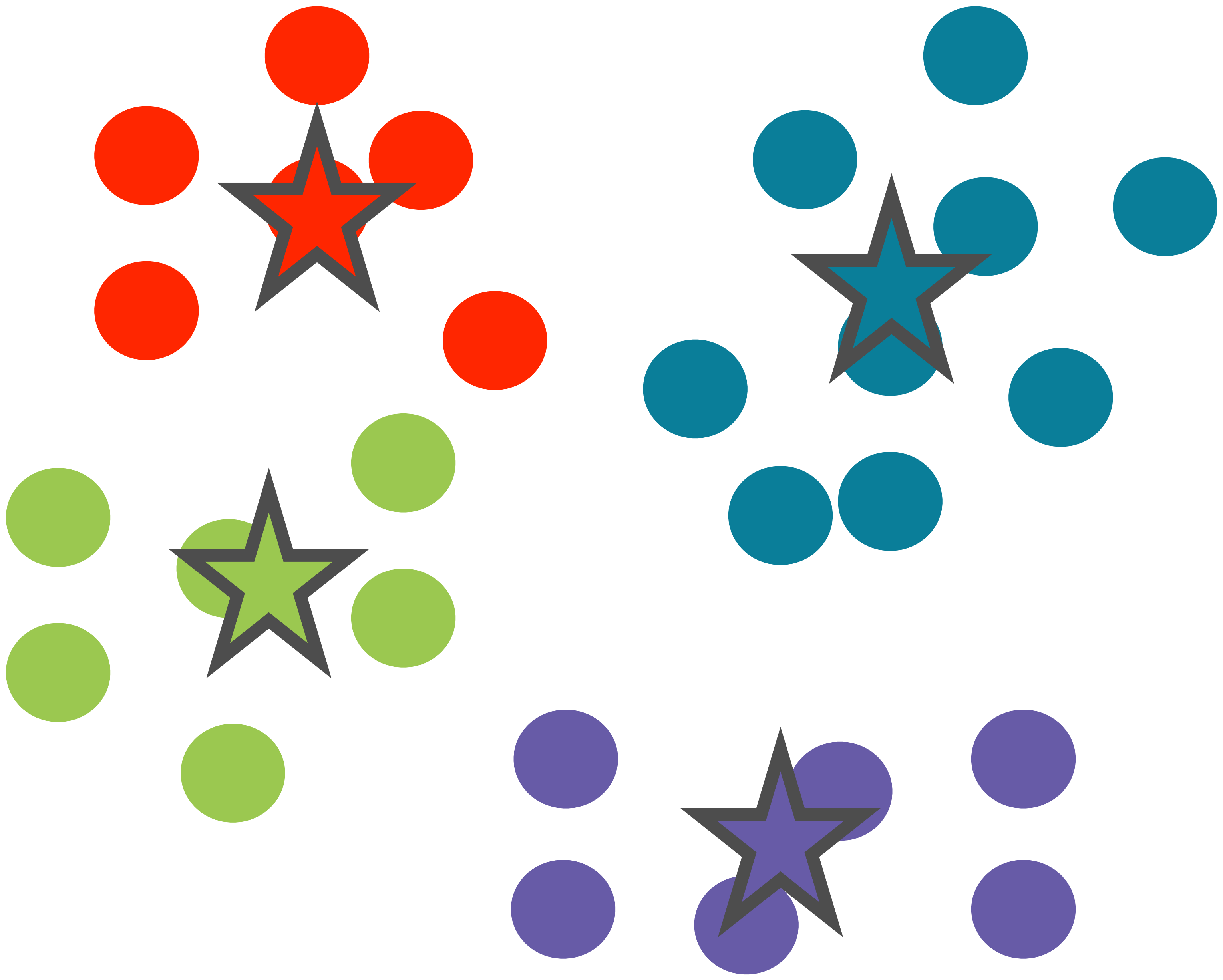


K-Means Clustering



Convergence

K-Means Clustering



K-Means Clustering

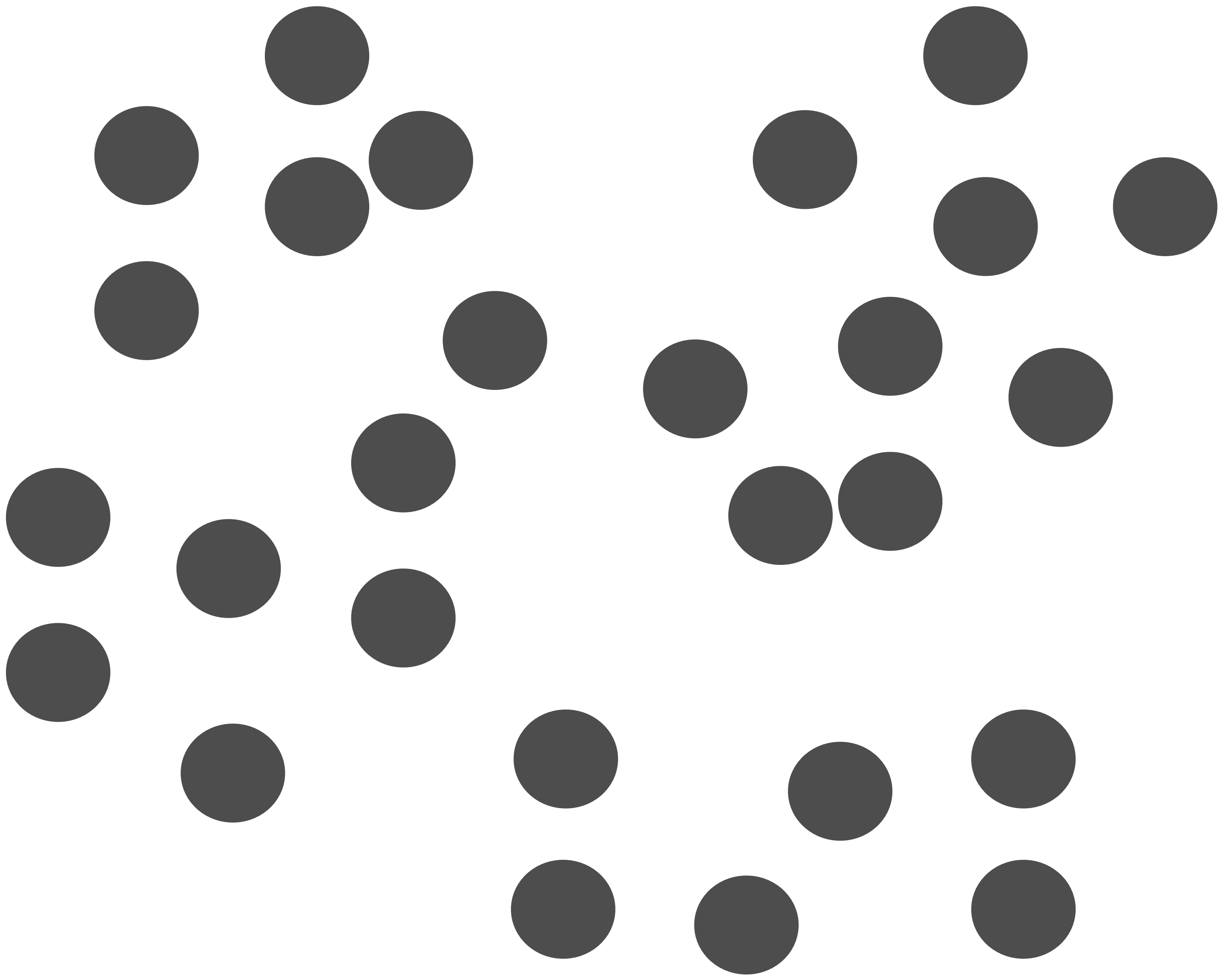
Each cluster has a representative point called a reference vector

K-Means Clustering

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

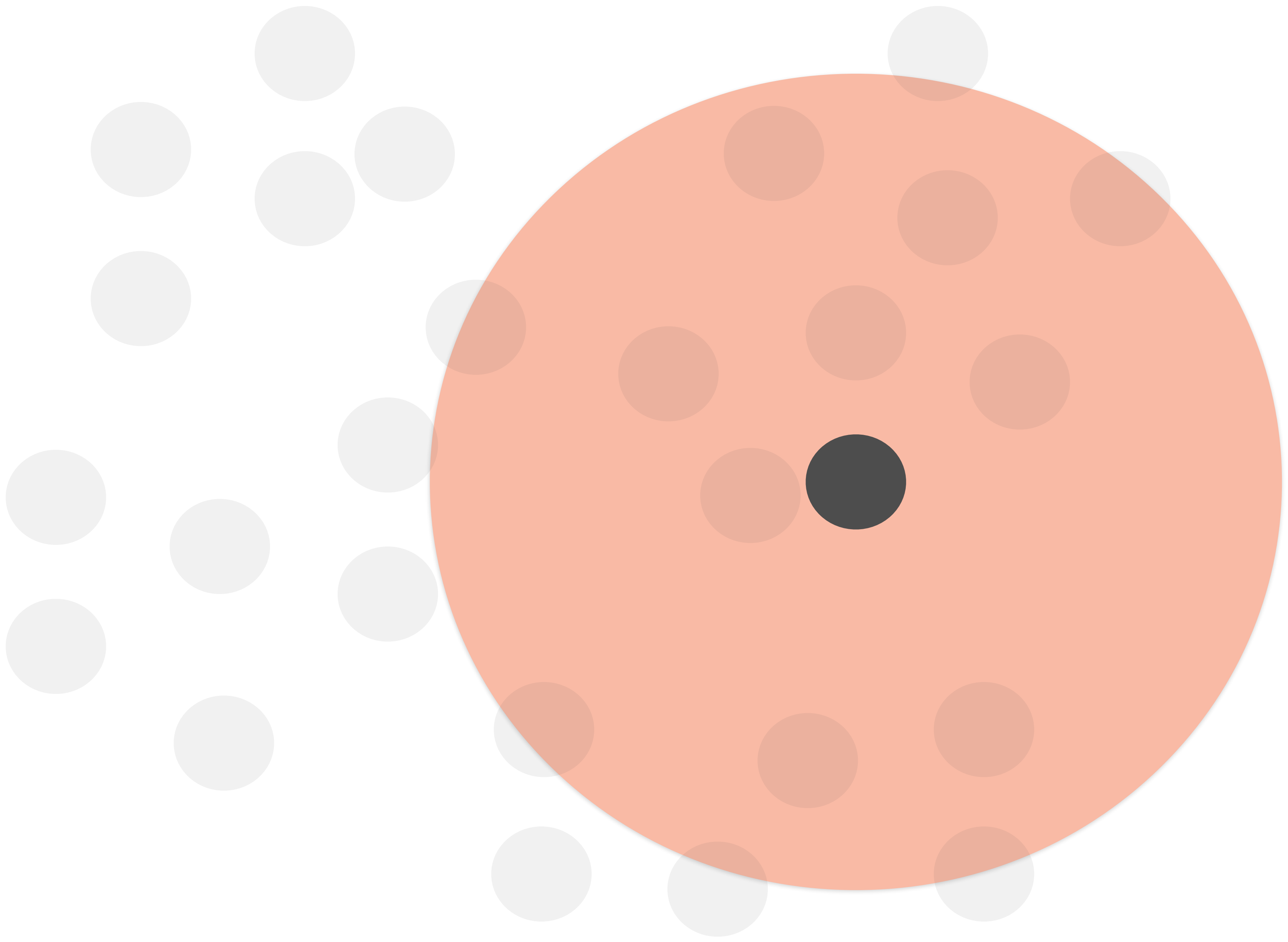
Mean Shift Clustering

Start with a set of points in space



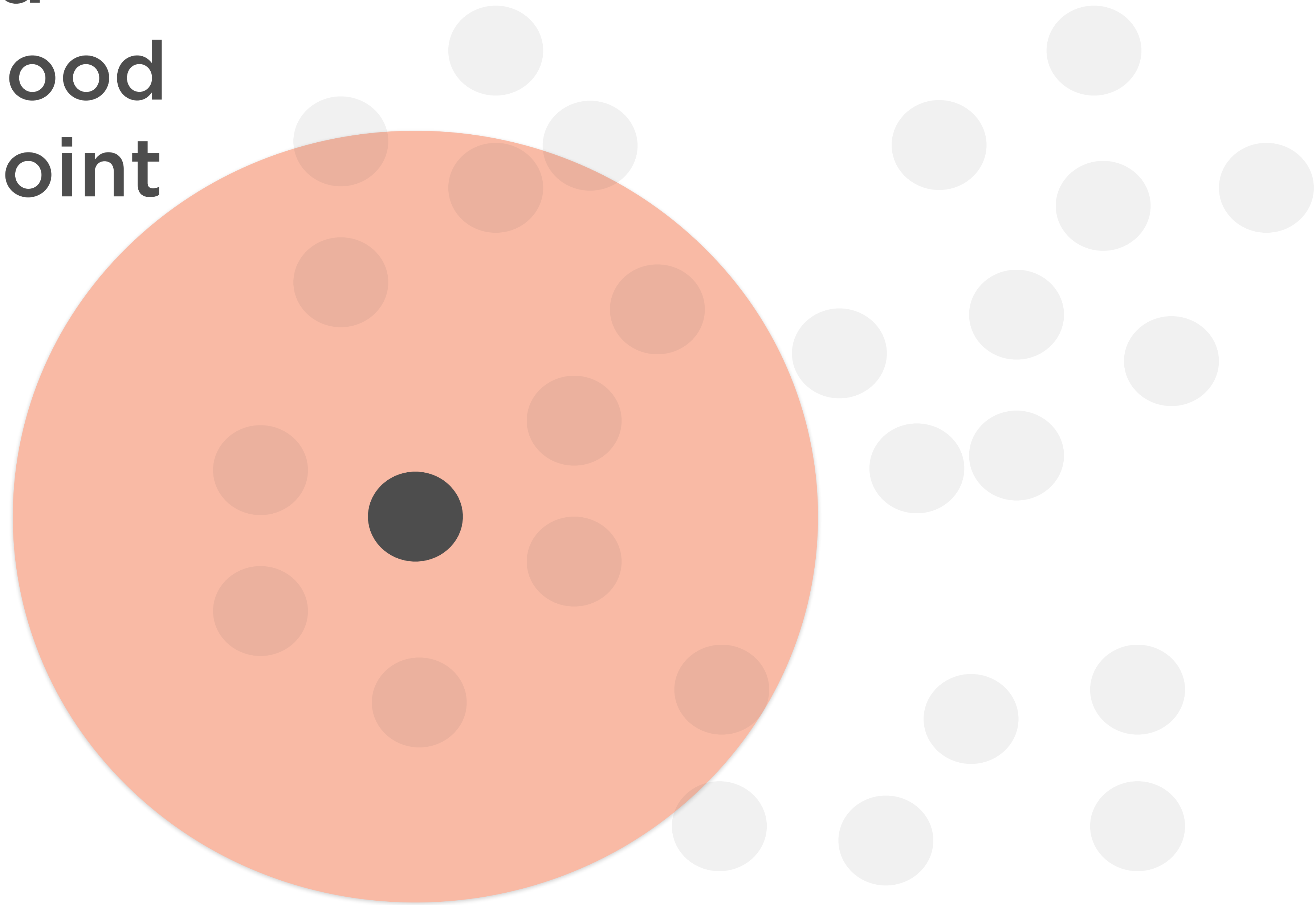
Mean Shift Clustering

Define a neighborhood for each point



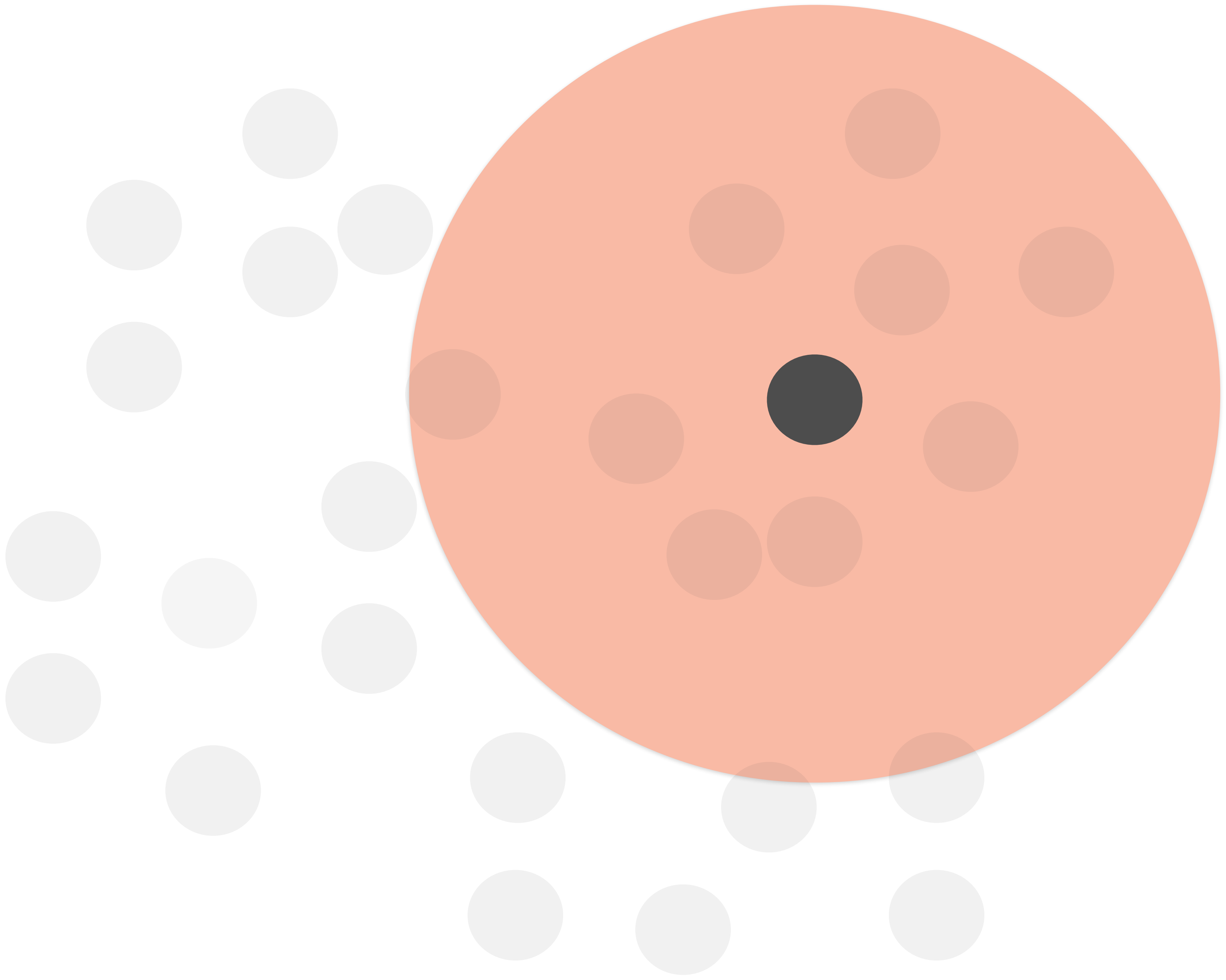
Mean Shift Clustering

Define a neighborhood for each point



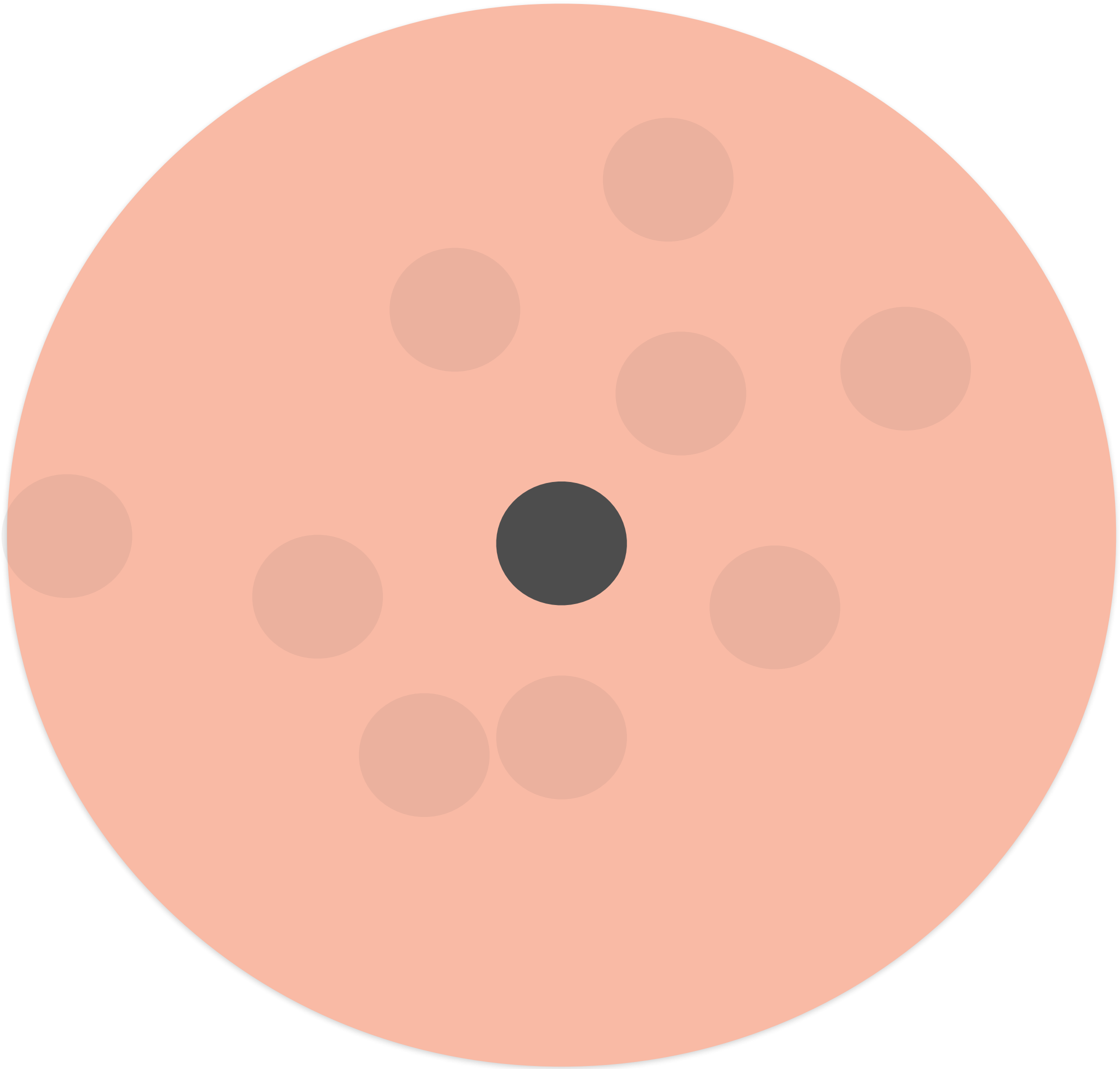
Mean Shift Clustering

Define a neighborhood for each point



Mean Shift Clustering

For each point, calculate a function based on all points in the neighborhood

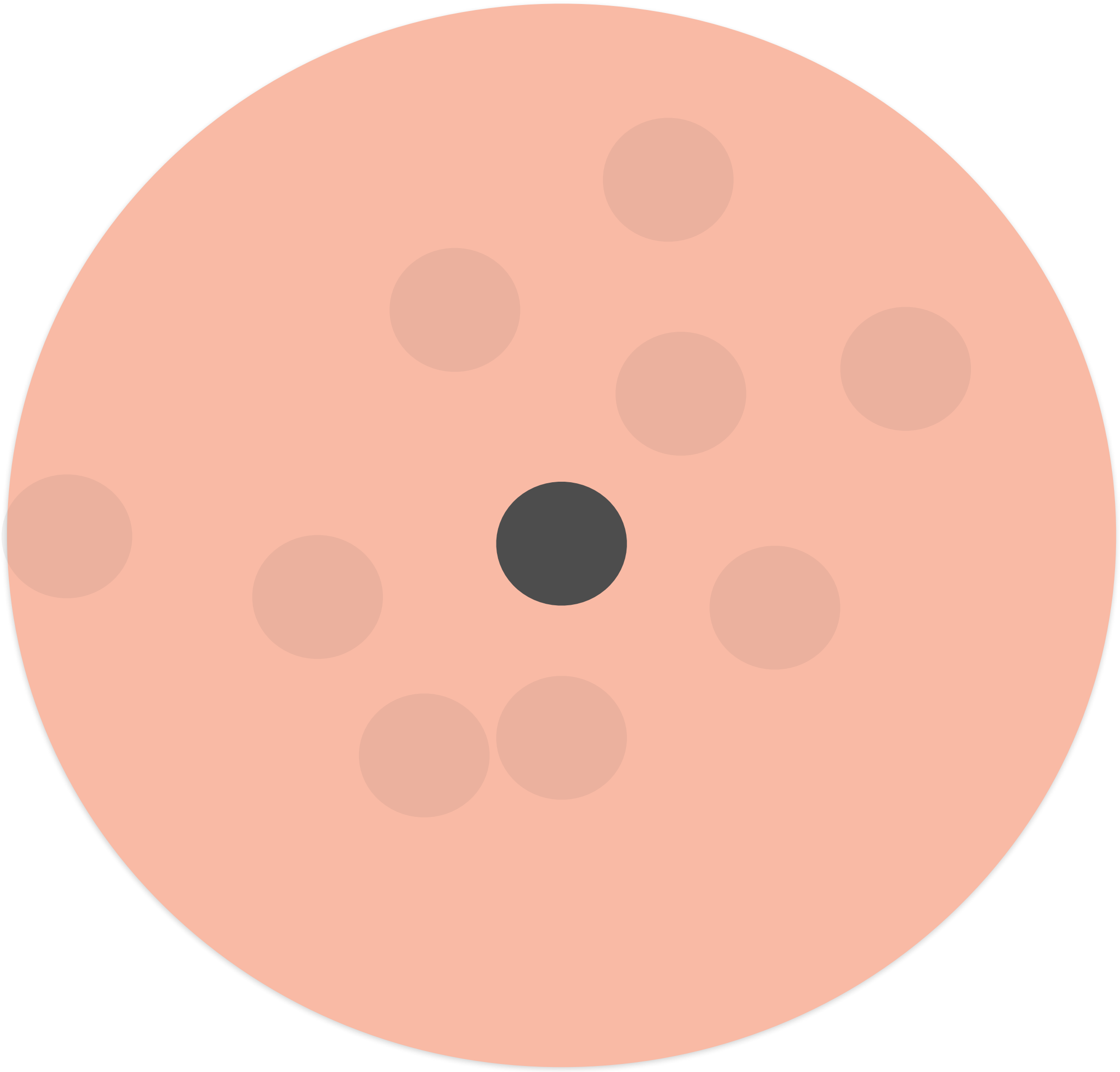


That function is called the **kernel**

Flat Kernel

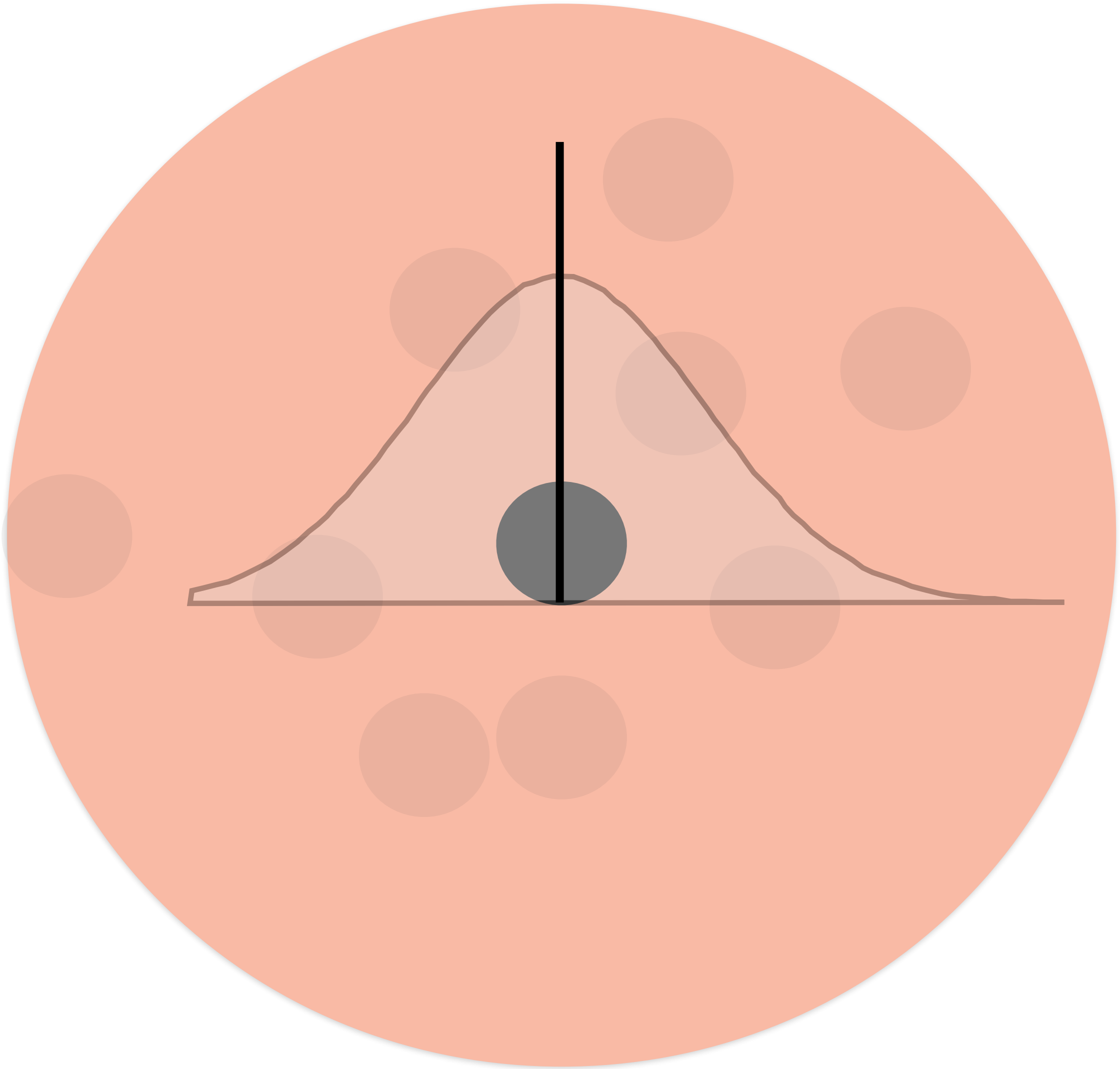
Flat kernel: sum of all points in neighborhood

Each point gets the same weight



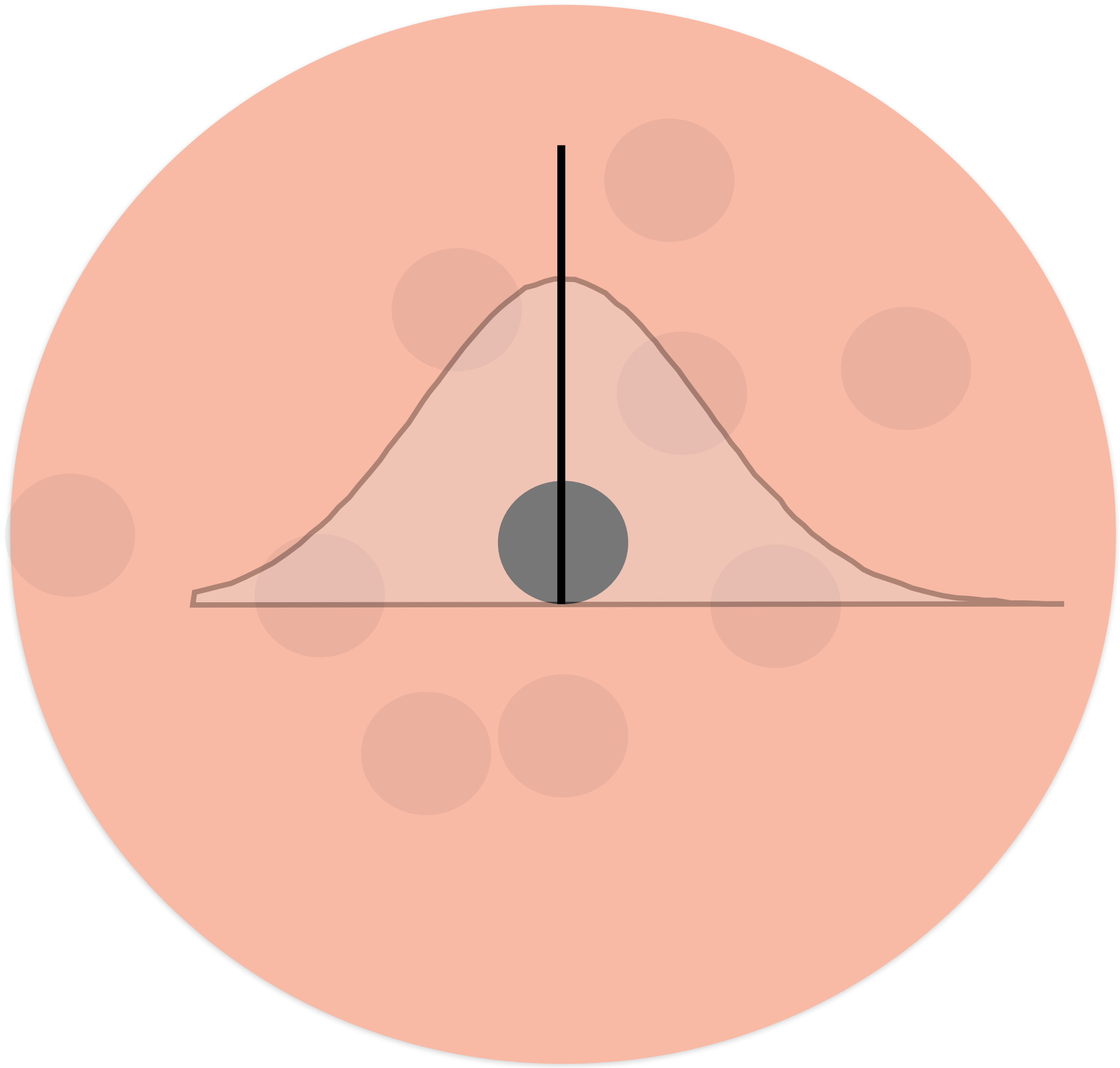
Gaussian (RBF) Kernel

Probability-weighted sum of points



What probability distribution?

Gaussian (RBF) Kernel

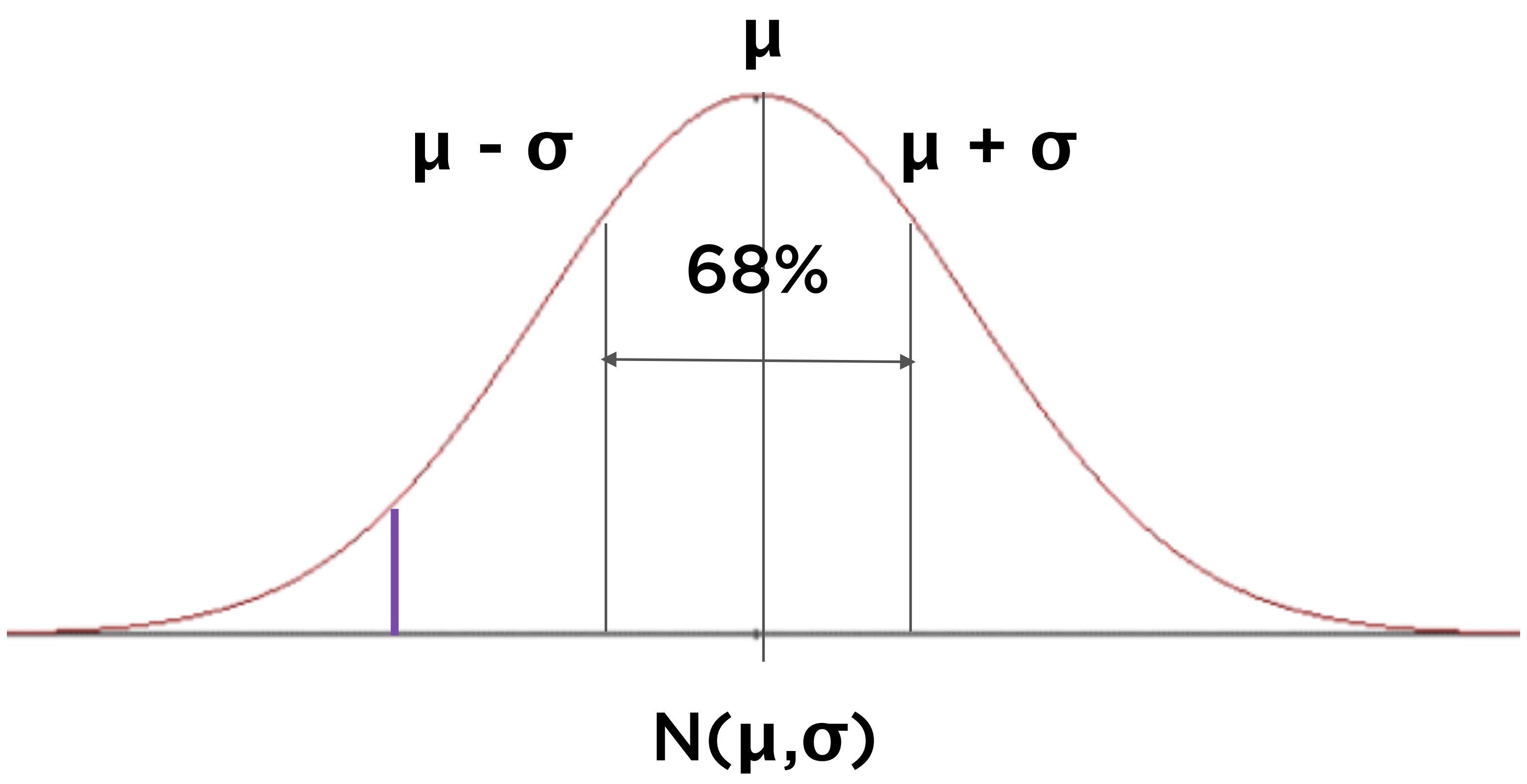


Gaussian probability distribution

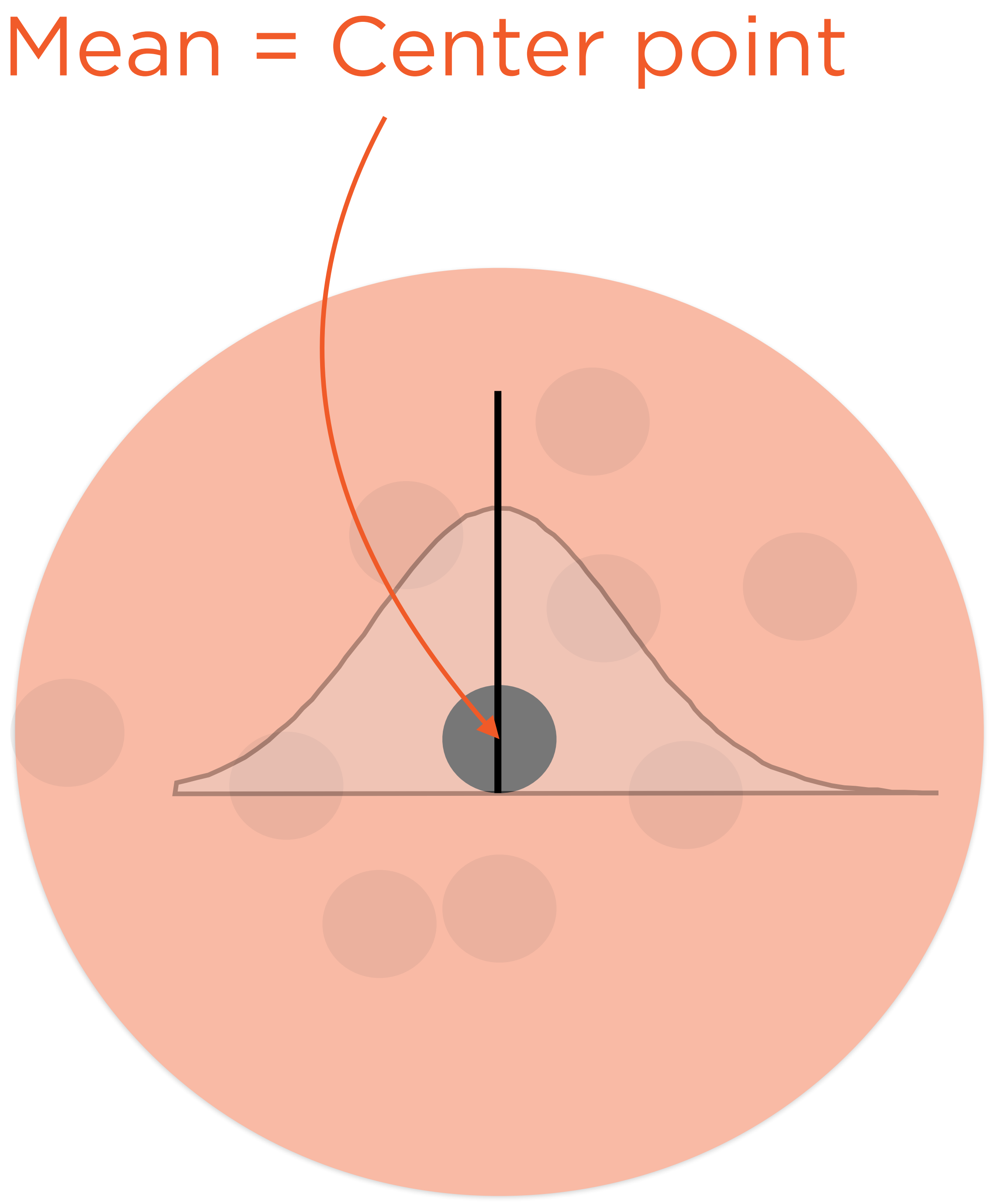
Defined by

- mean μ
- standard deviation σ

Gaussian Distribution



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



Gaussian (RBF) Kernel

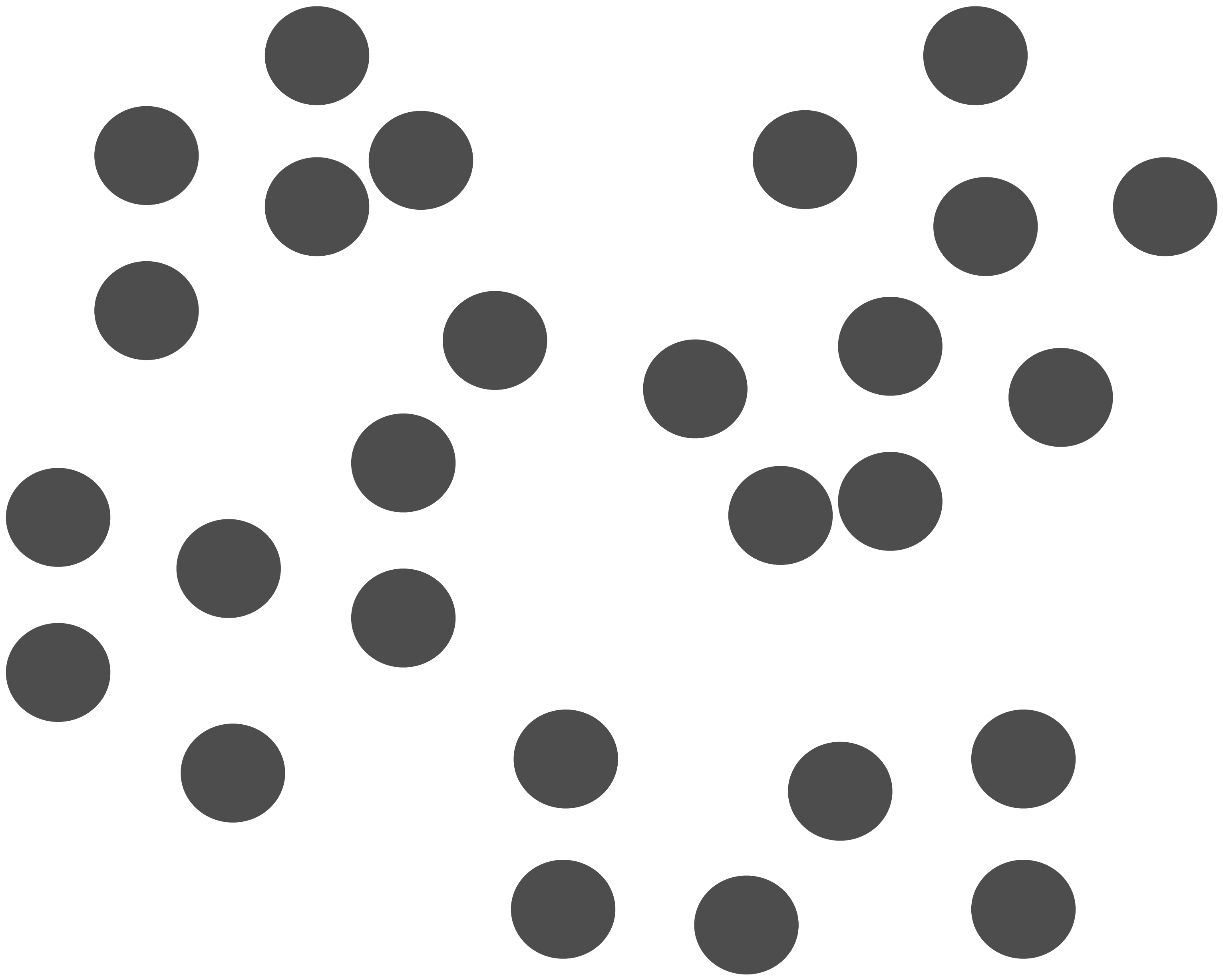
Mean μ = center point

Standard deviation σ ~ bandwidth

(Bandwidth is a hyperparameter)

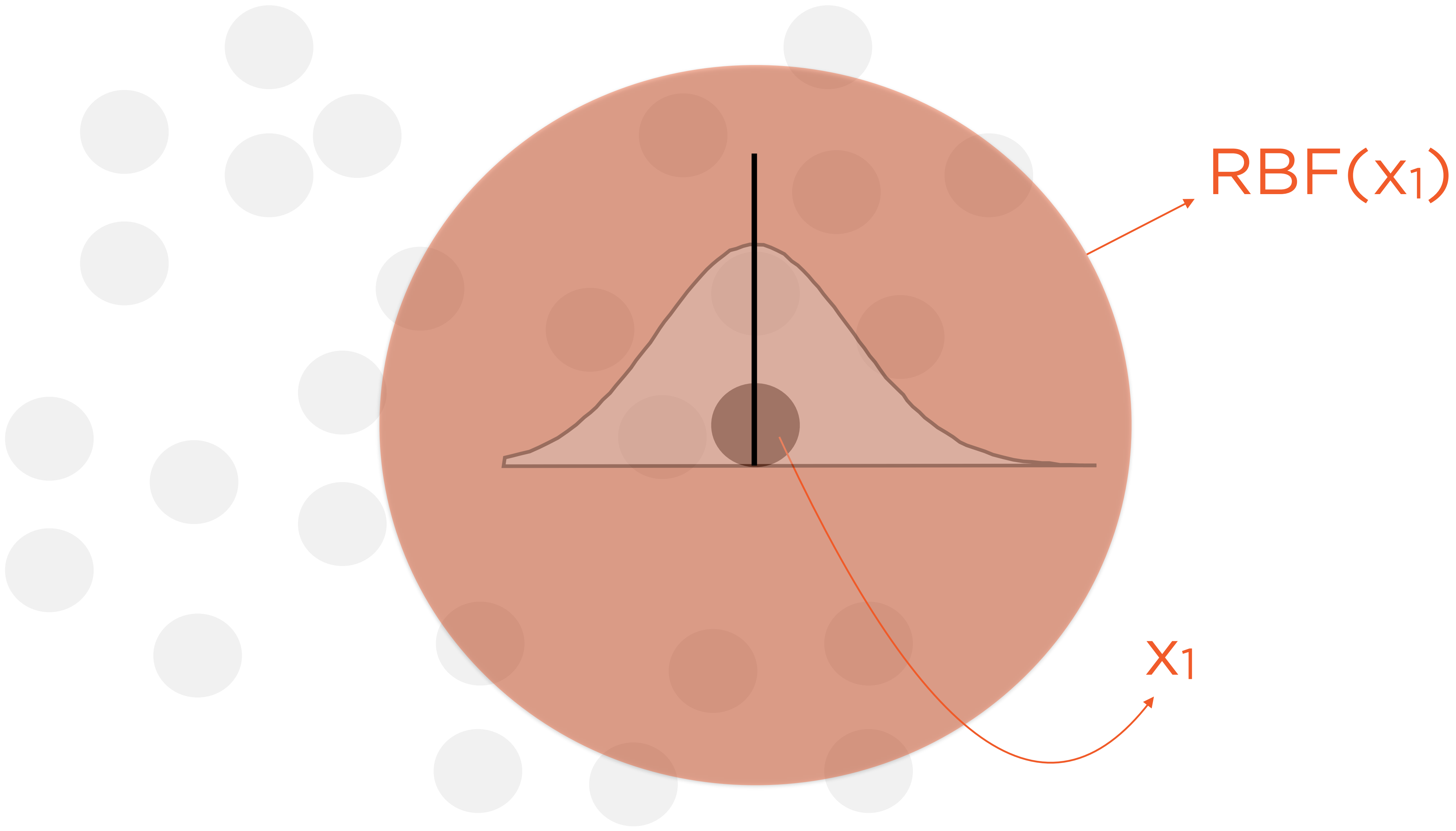
Mean Shift Clustering

Kernel is applied to each point



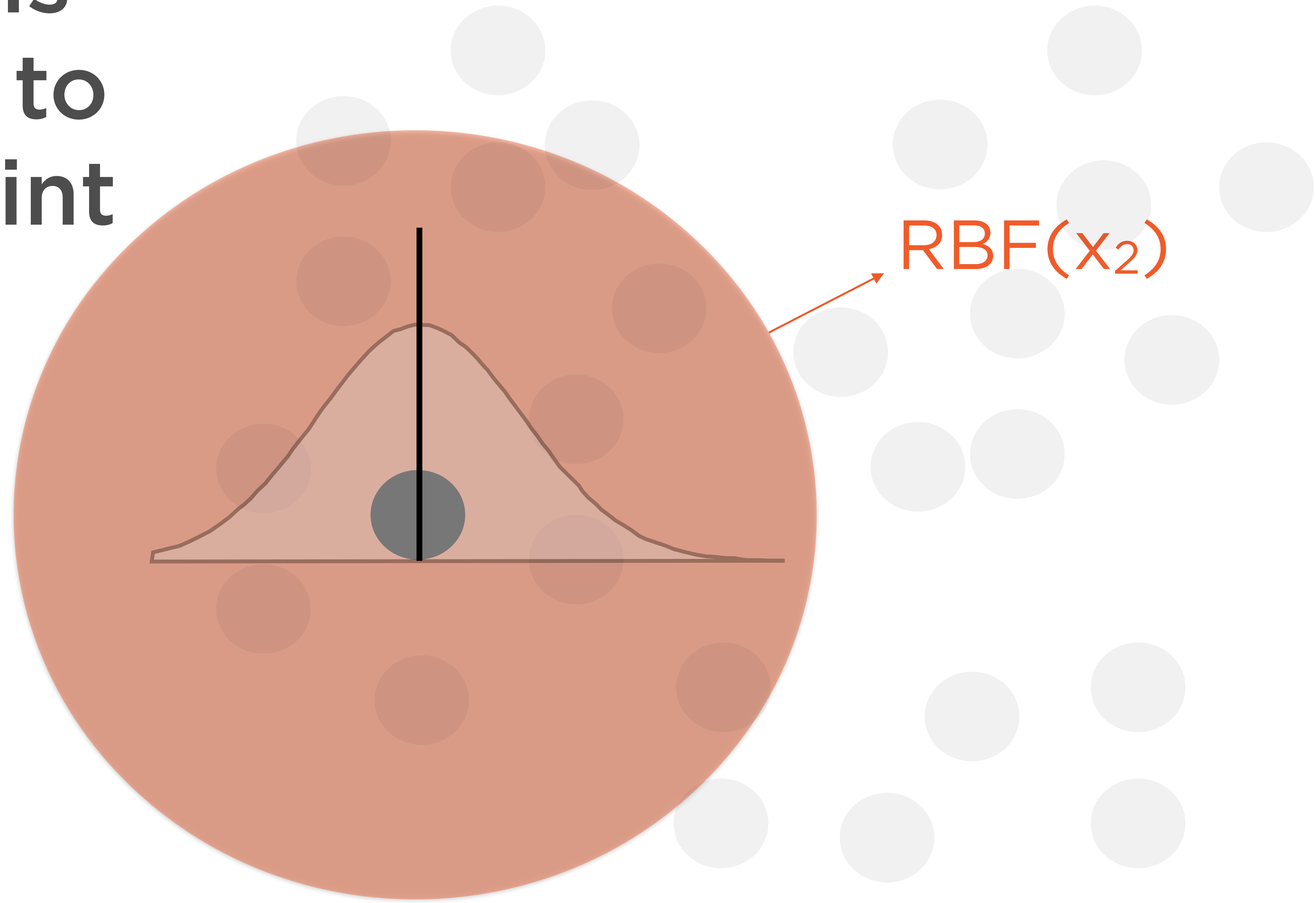
Mean Shift Clustering

Kernel is applied to each point



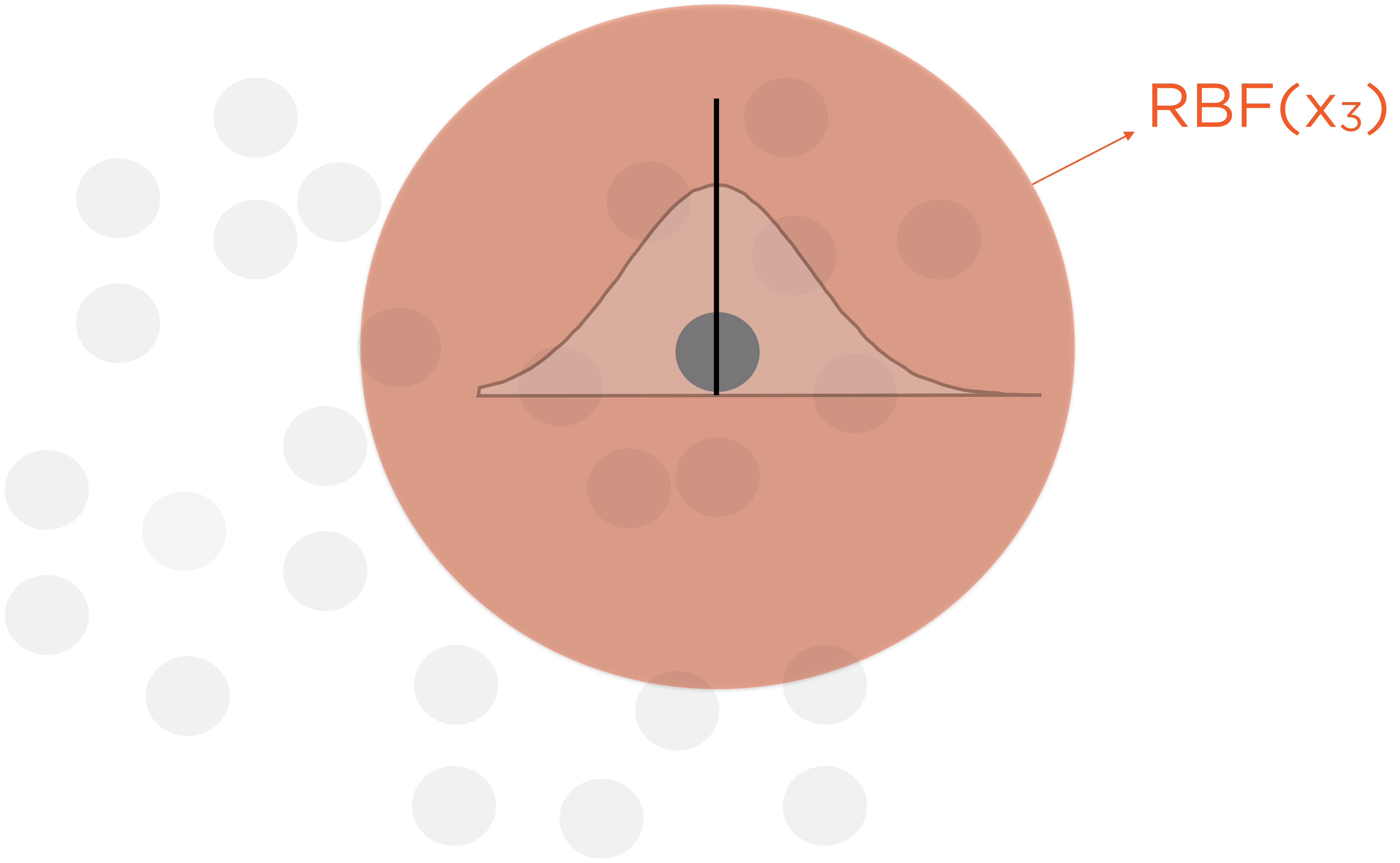
Mean Shift Clustering

Kernel is applied to each point



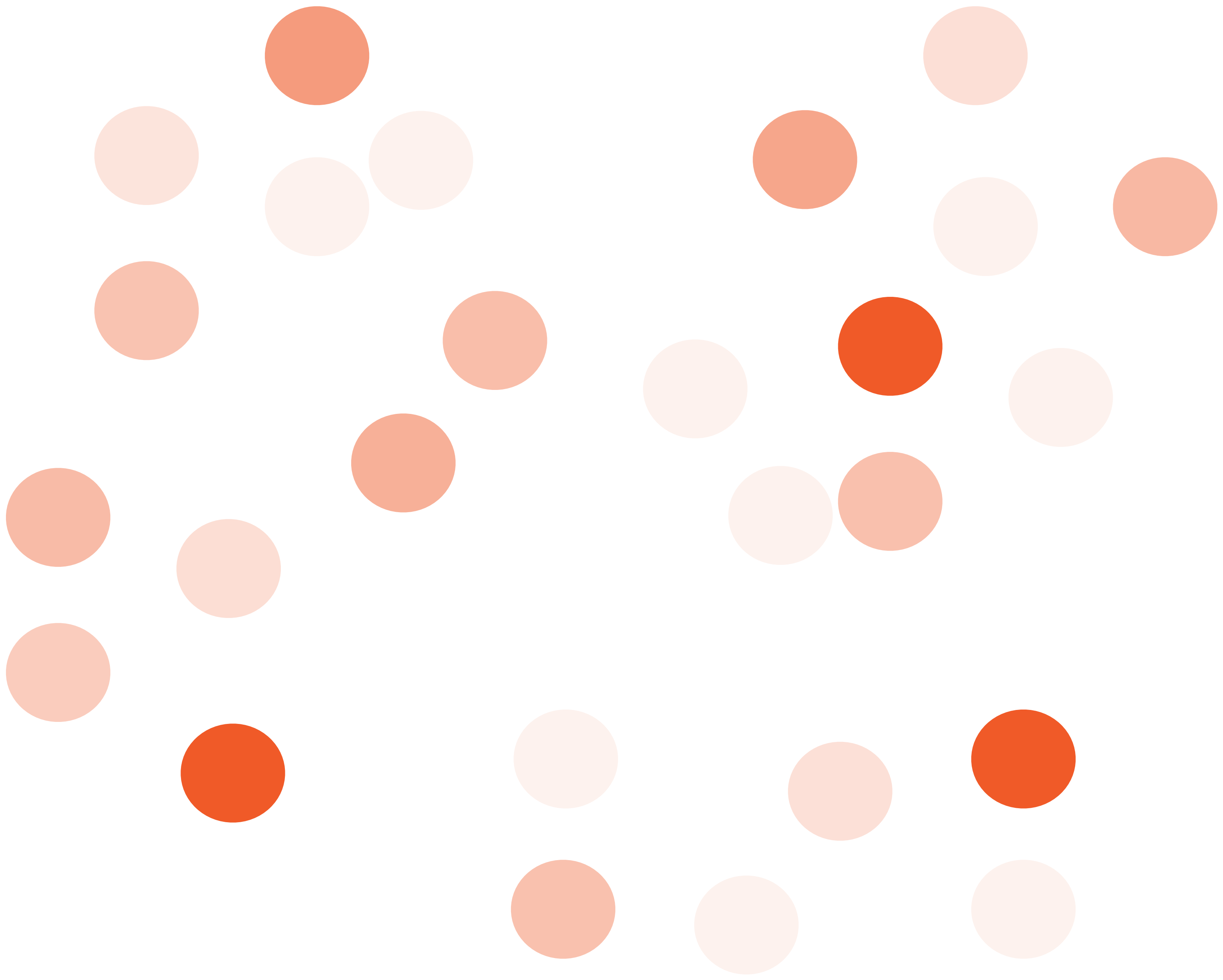
Mean Shift Clustering

Kernel is applied to each point

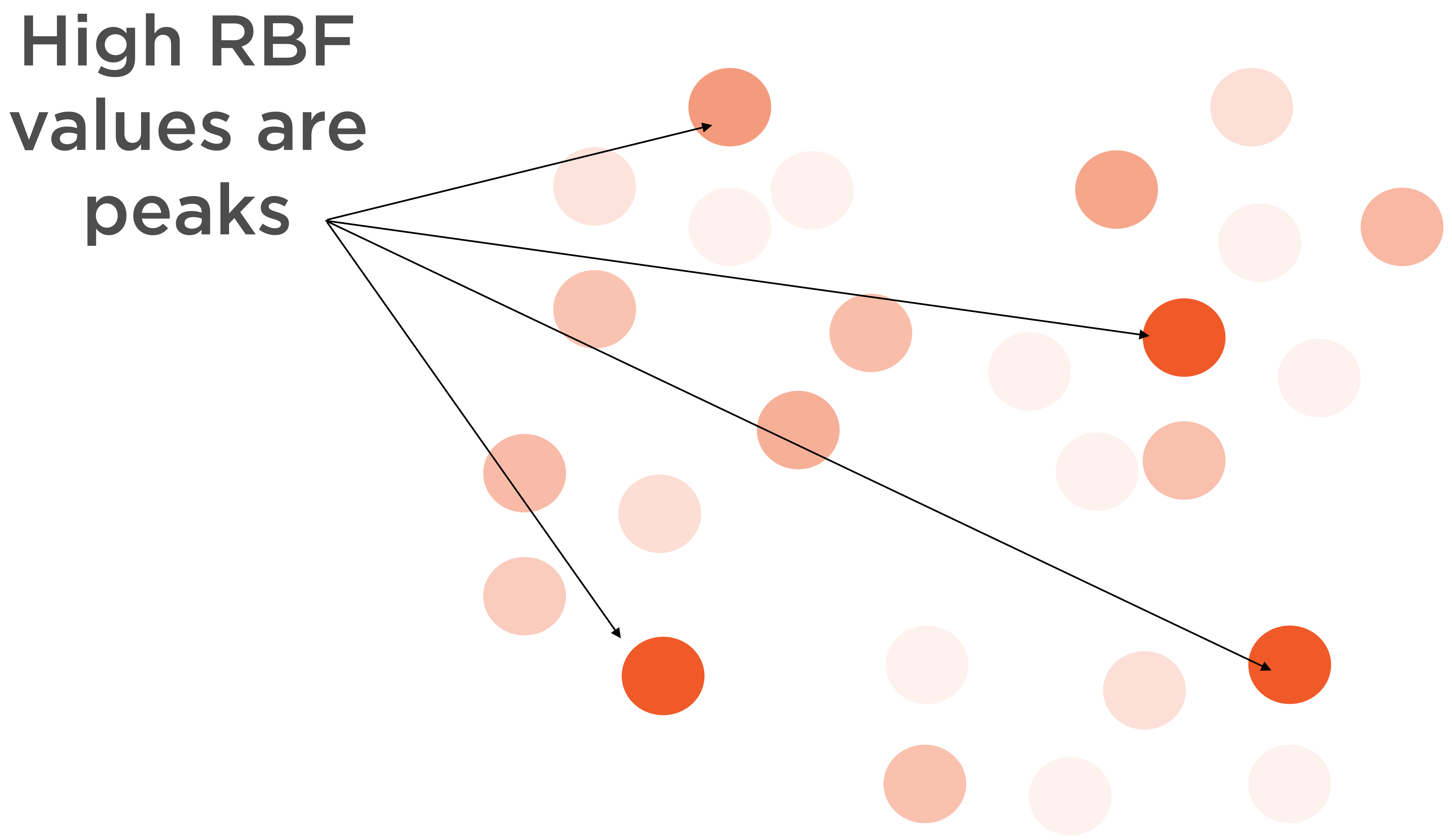


Mean Shift Clustering

Assume points are color-coded by magnitude of RBF

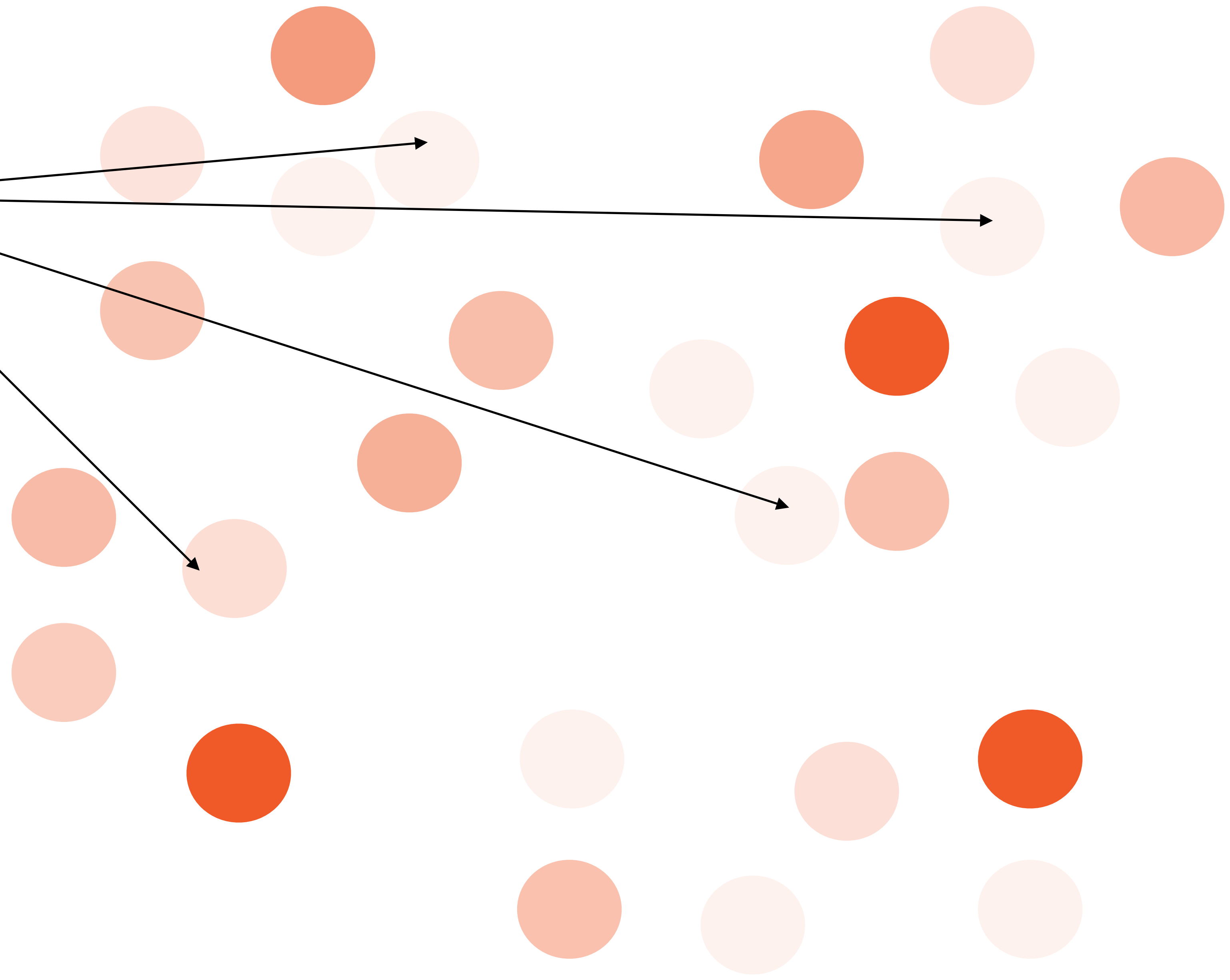


Mean Shift Clustering



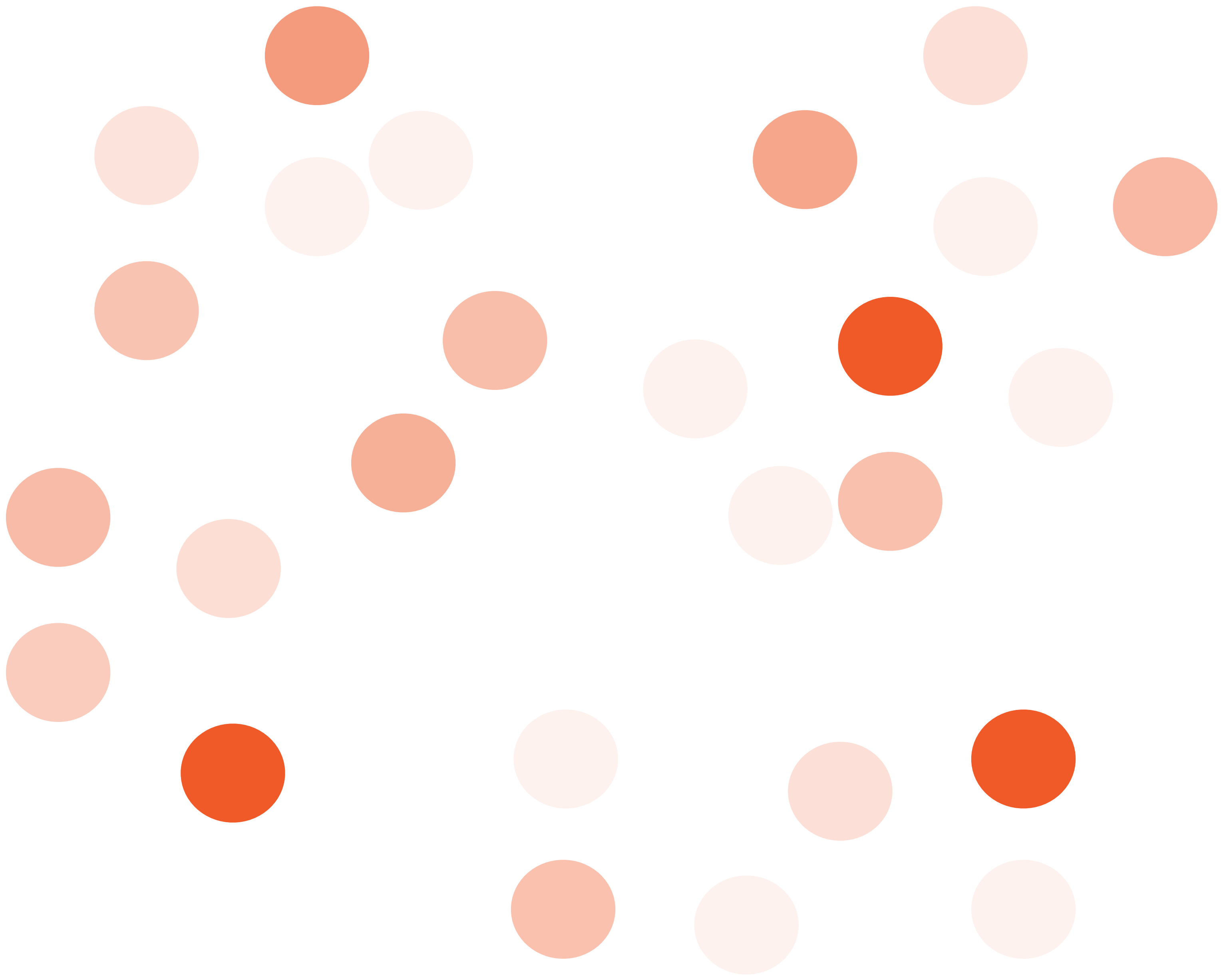
Mean Shift Clustering

Low RBF values are troughs



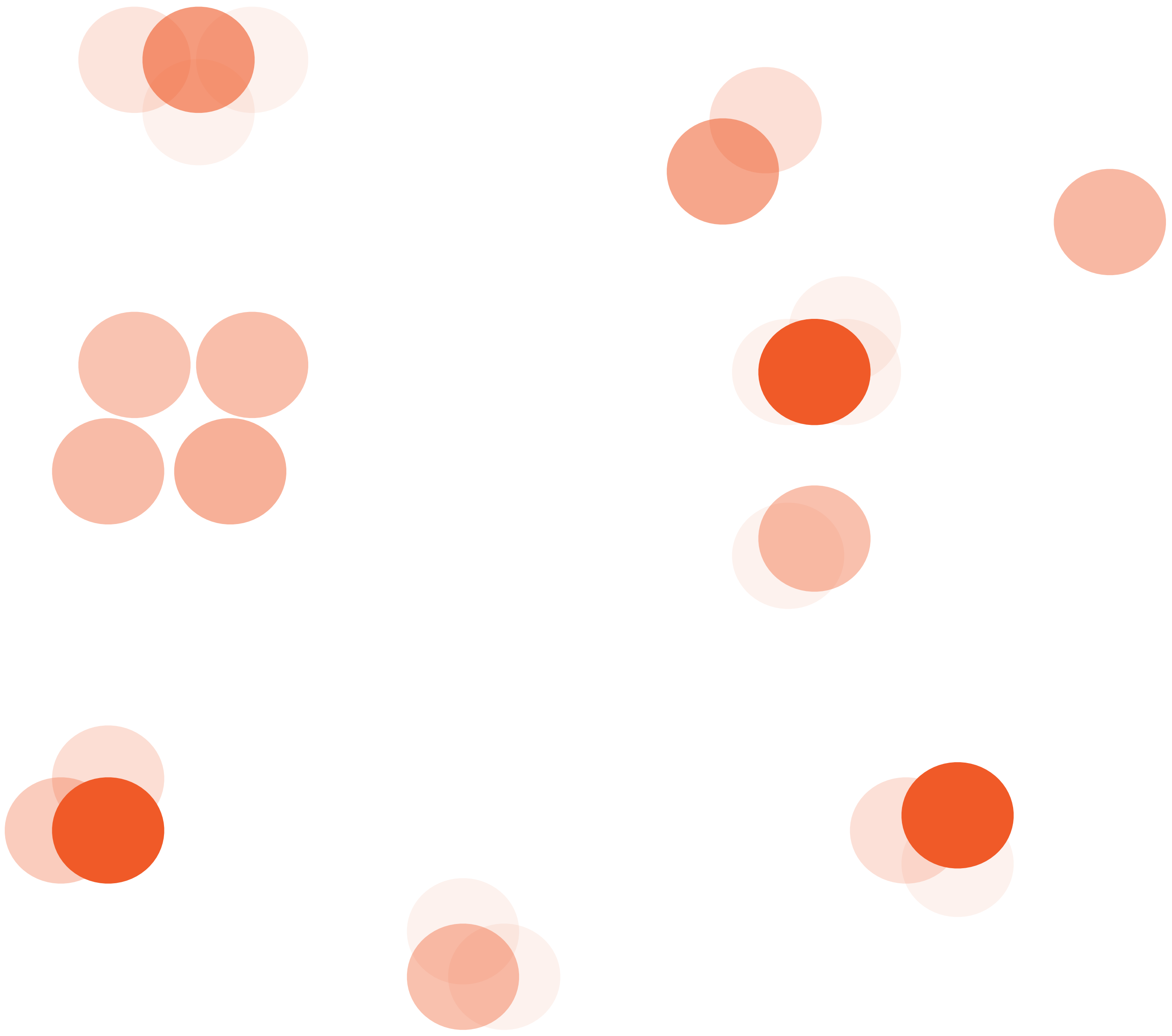
Mean Shift Clustering

Now, all points start to "shift" towards the nearest peak



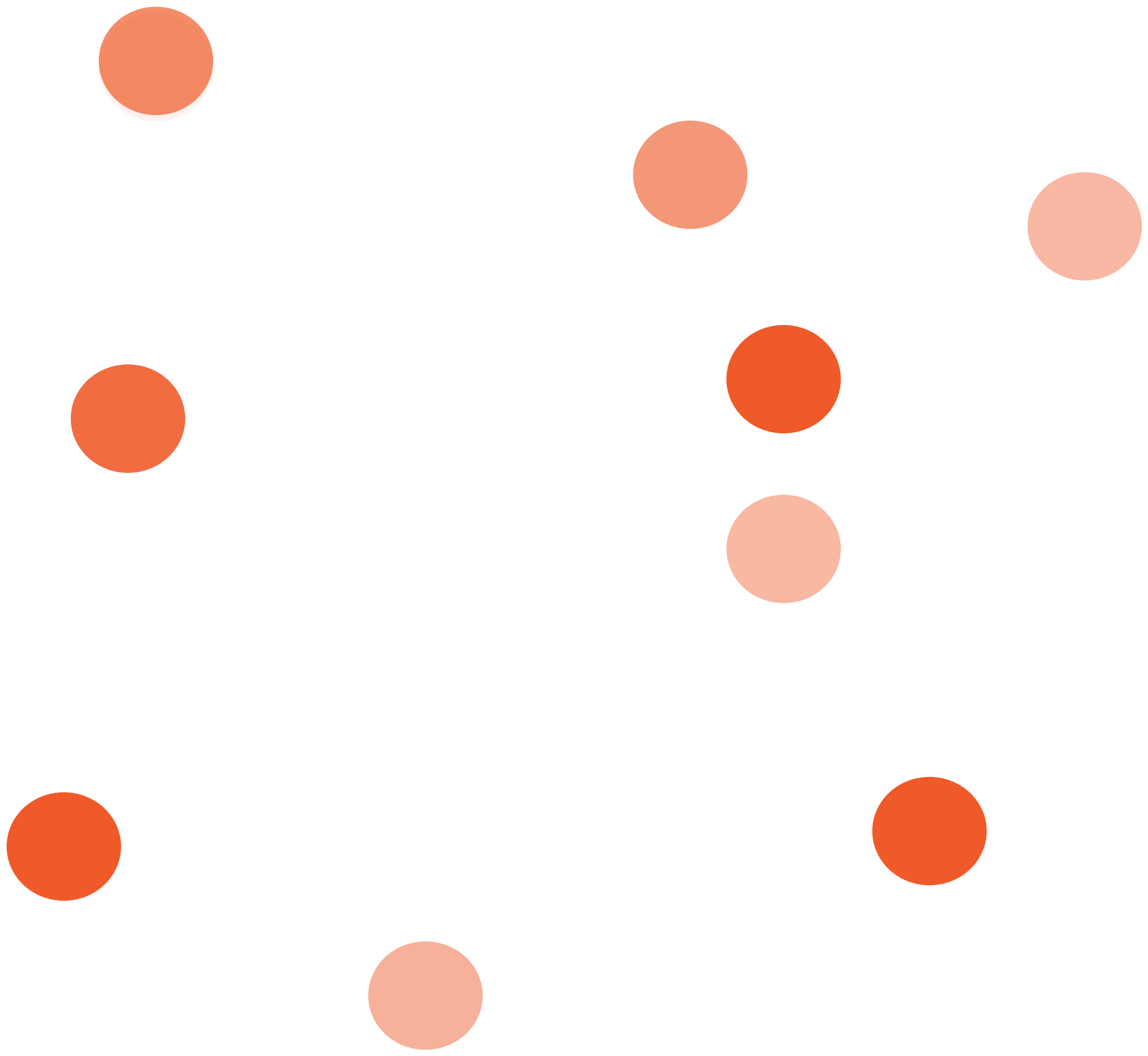
Mean Shift Clustering

Now, all points start to "shift" towards the nearest peak



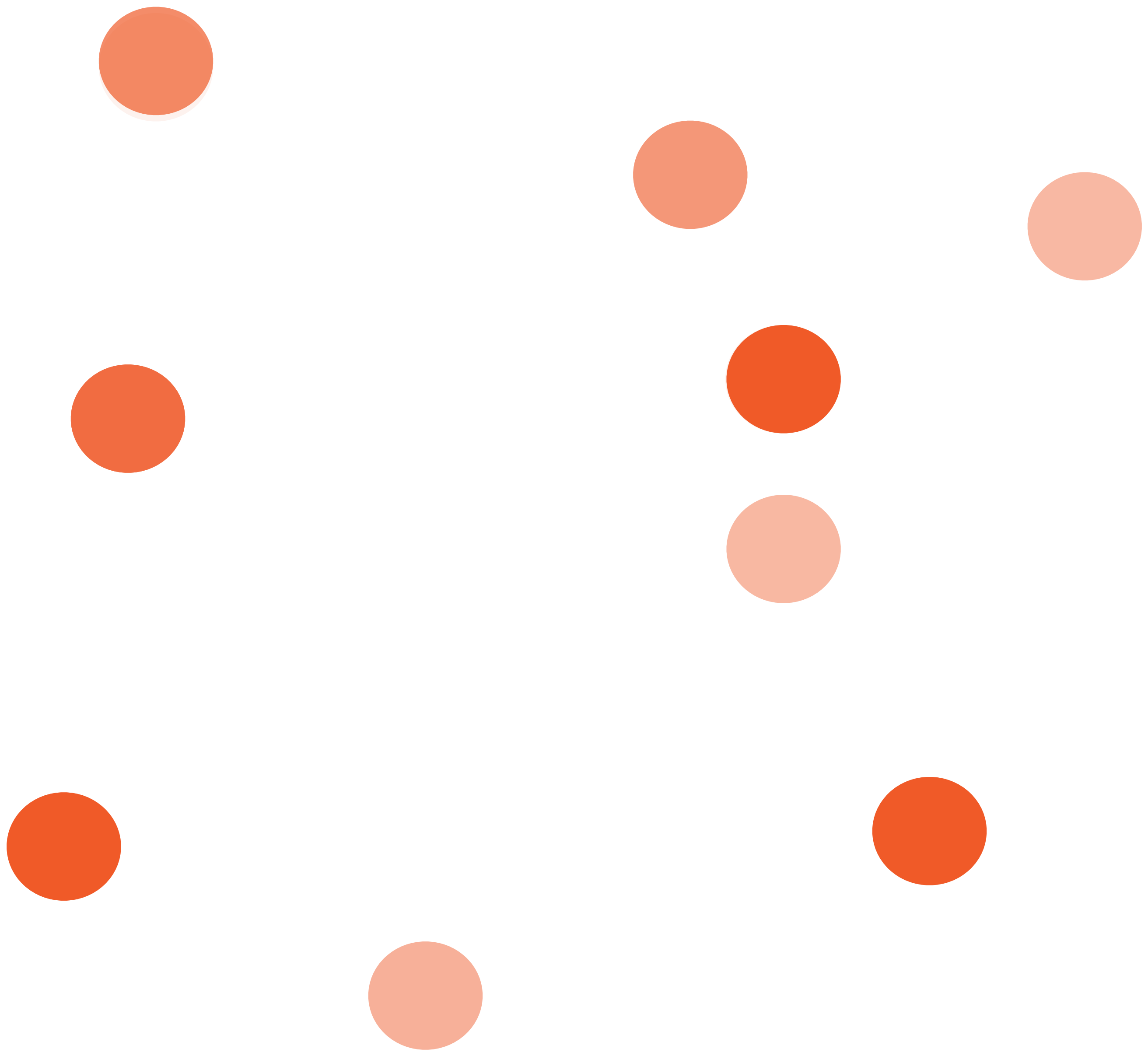
Mean Shift Clustering

Now, all points start to "shift" towards the nearest peak



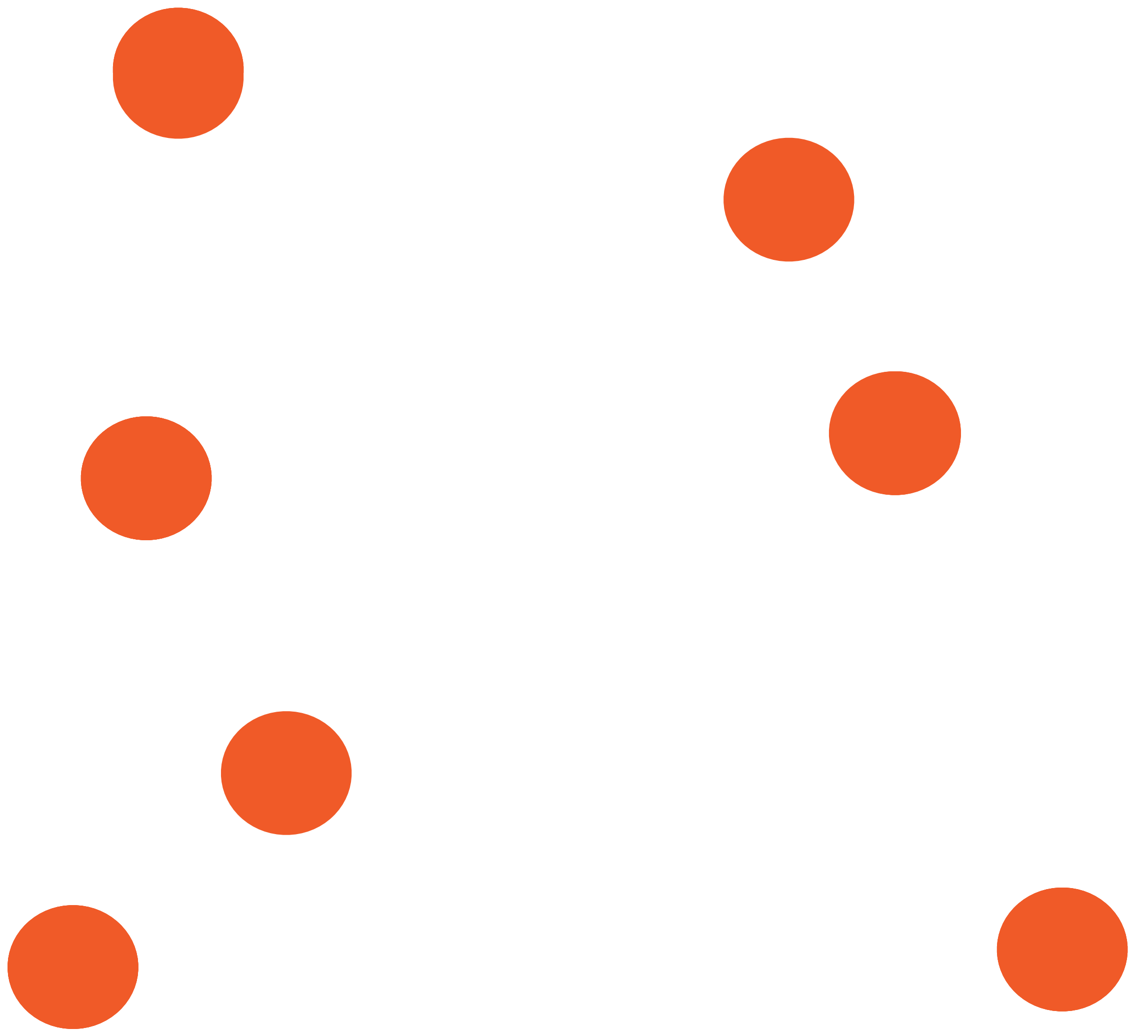
Mean Shift Clustering

This is the "mean shift"



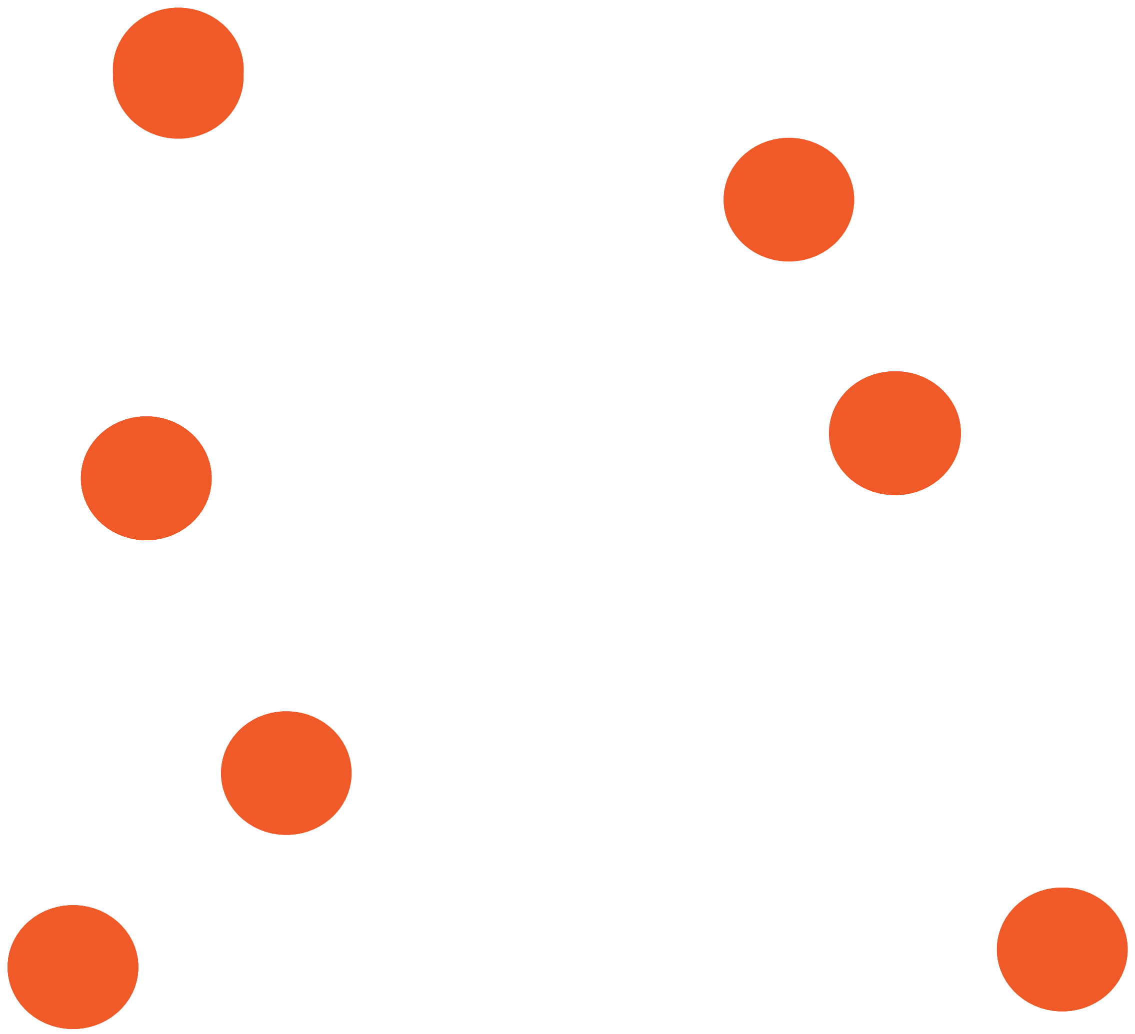
Mean Shift Clustering

This is the "mean shift"

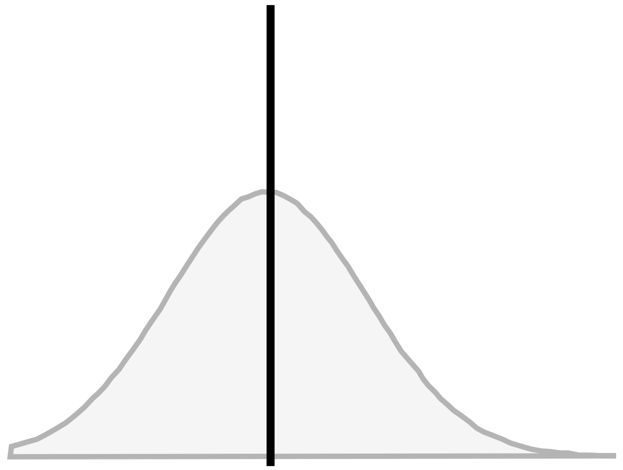


Mean Shift Clustering

Algorithm converges when points stop moving



Role of Bandwidth



$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

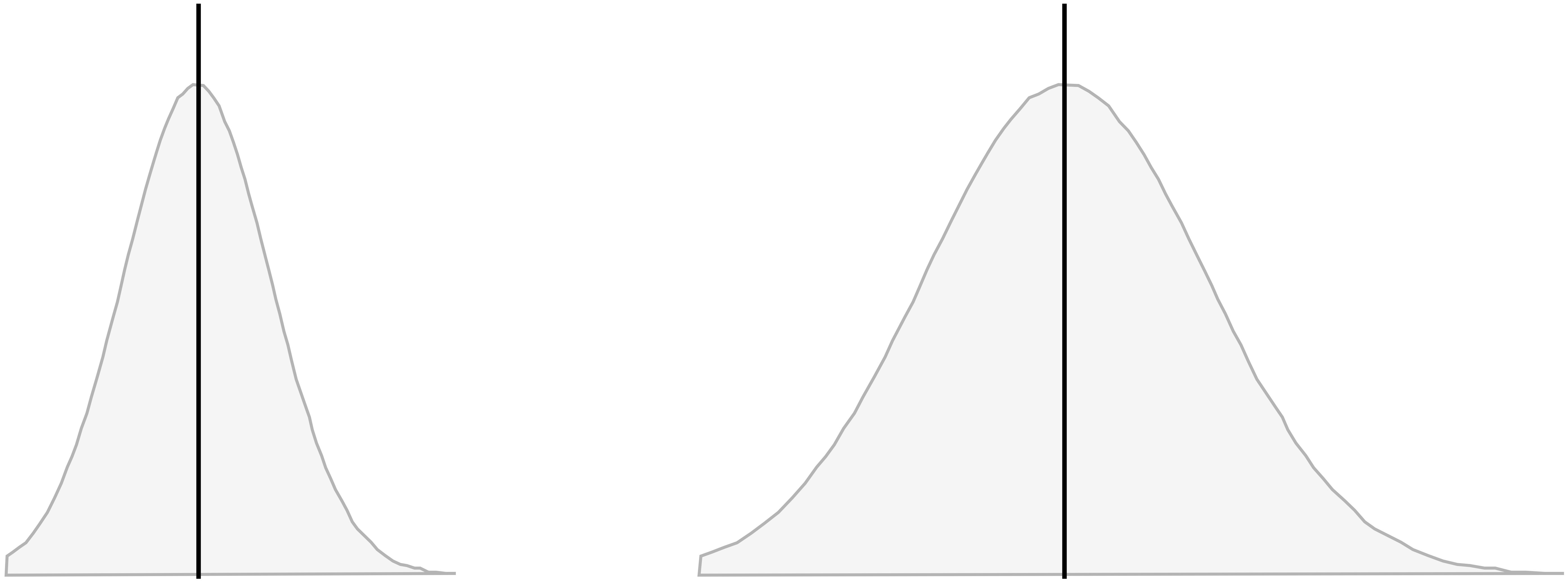
Standard deviation σ ~ bandwidth

Bandwidth is the only hyperparameter

Small bandwidth ~ tall skinny kernel

Large bandwidth ~ flat kernel

Role of Bandwidth



Tall skinny kernel

Ignore points far from the mean

Flatter kernel

Considers points far from the mean

Similar, yet Different

K-Means Clustering	Mean Shift Clustering
Need to specify number of clusters as hyperparameter	No need to specify number of clusters upfront as hyperparameter
Can’t handle some complex non-linear data	Uses density function to handle even complex non-linear data (e.g. pixels)
Less hyperparameter tuning needed	Hyperparameter tuning very important

Similar, yet Different

K-Means Clustering

Computationally less intensive

O(N) in number of data points

Struggles with outliers

Mean Shift Clustering

Computationally very intensive

$O(N^2)$ in number of data points

Copes better with outliers

Demo

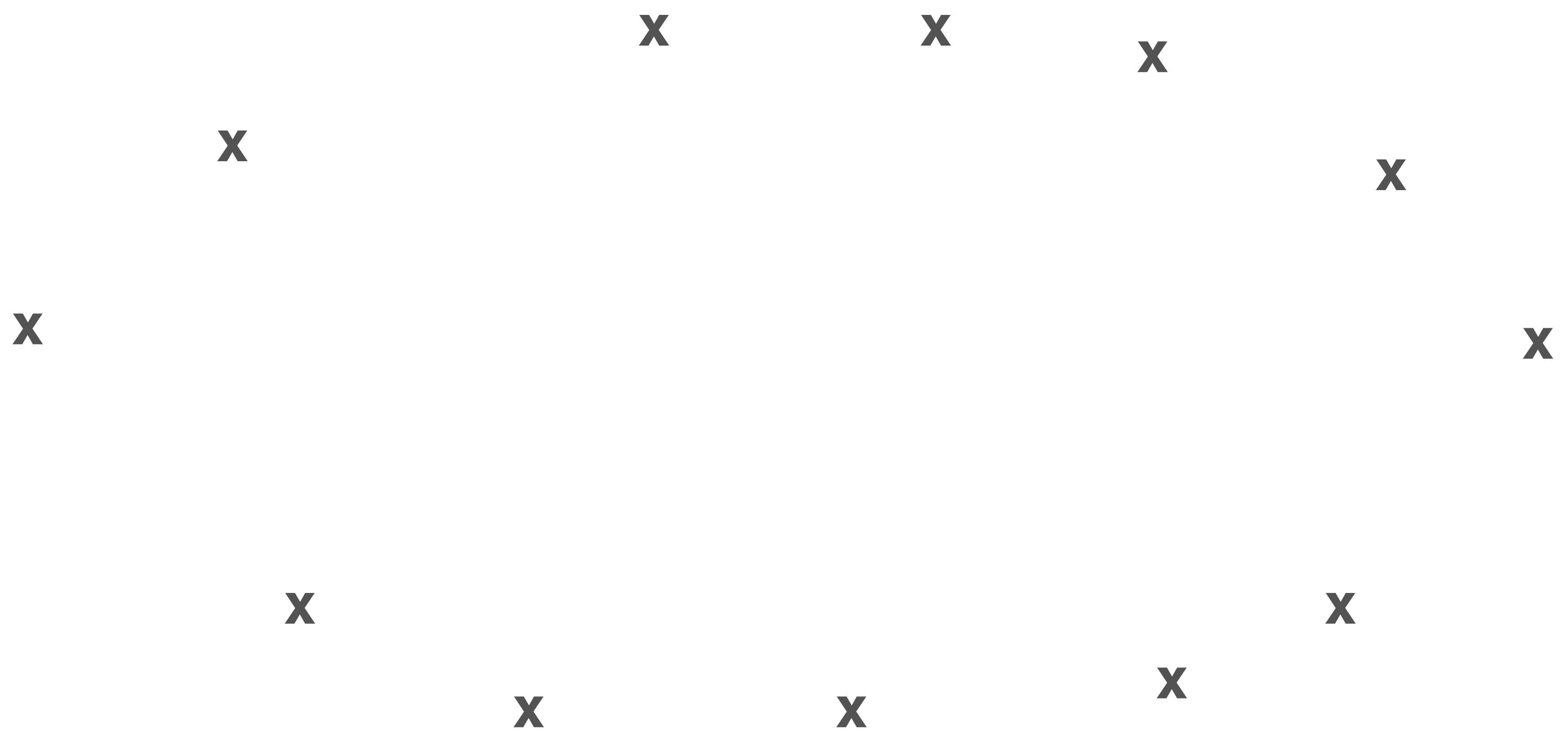
Implement mean-shift clustering in scikit-learn

Principal Components Analysis

Principal Components Analysis

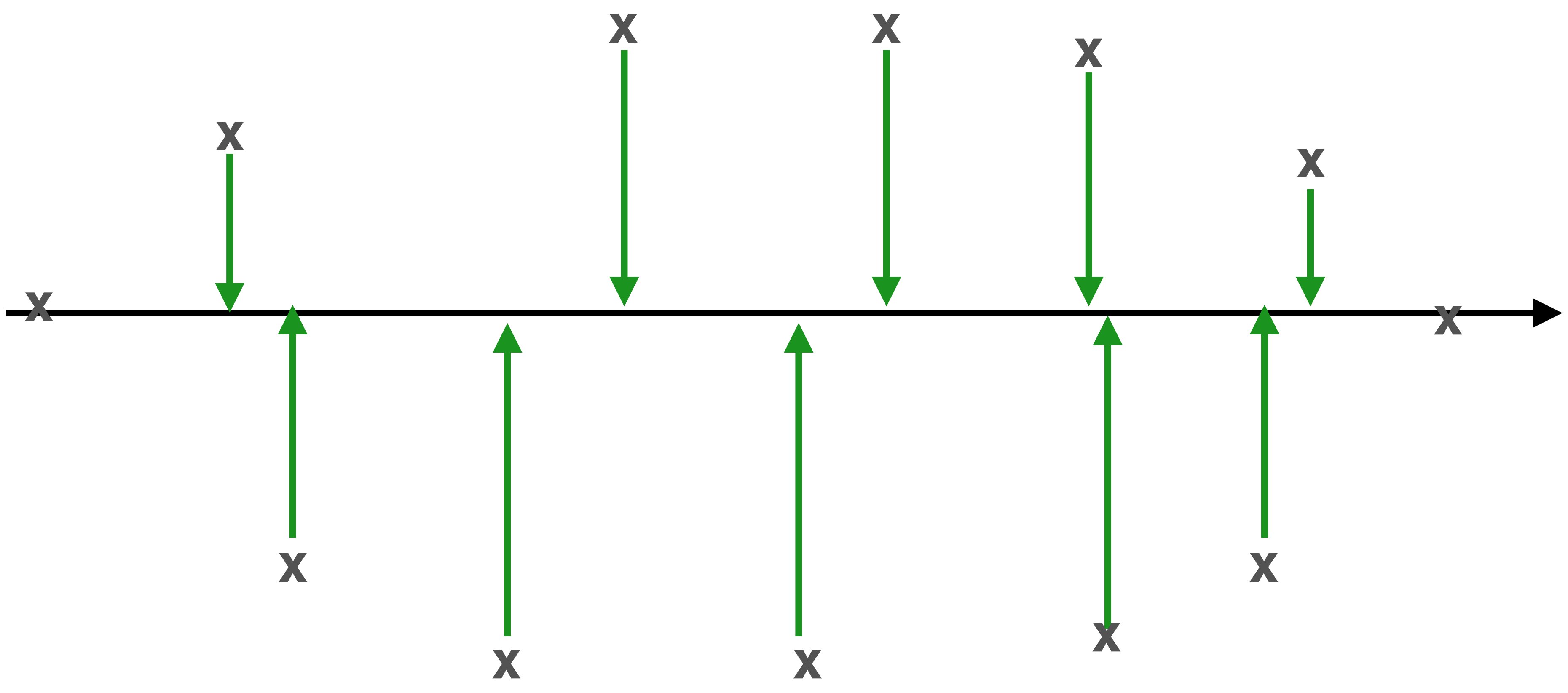
A technique to re-express complex data in terms of a few, well-chosen vectors (Principal Components) that most efficiently capture the variation in that data

Intuition Behind PCA



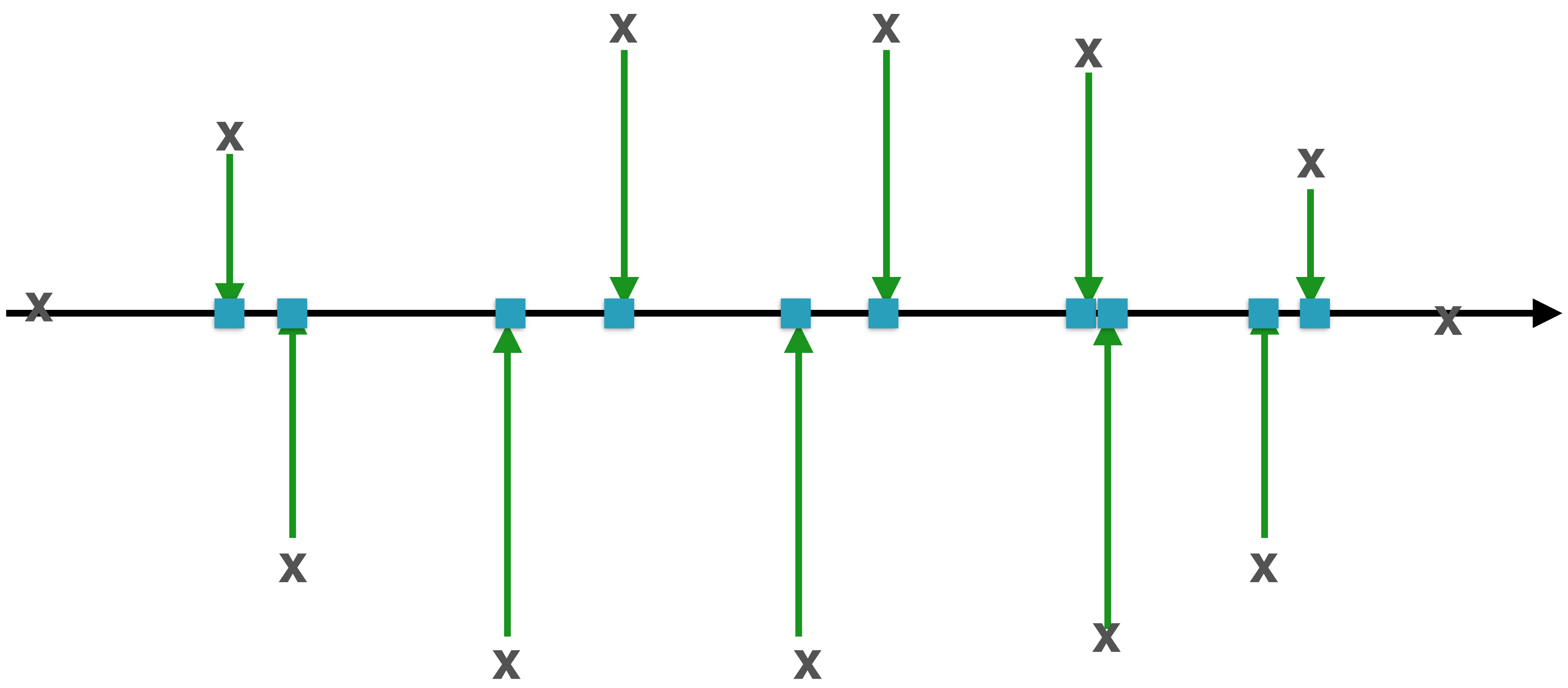
Objective: Find the “best” directions to represent this data

Intuition Behind PCA



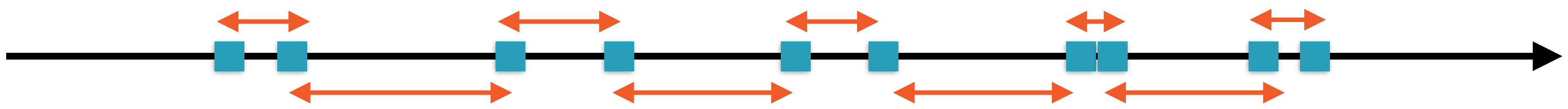
Start by "projecting" the data onto a line in some direction

Intuition Behind PCA



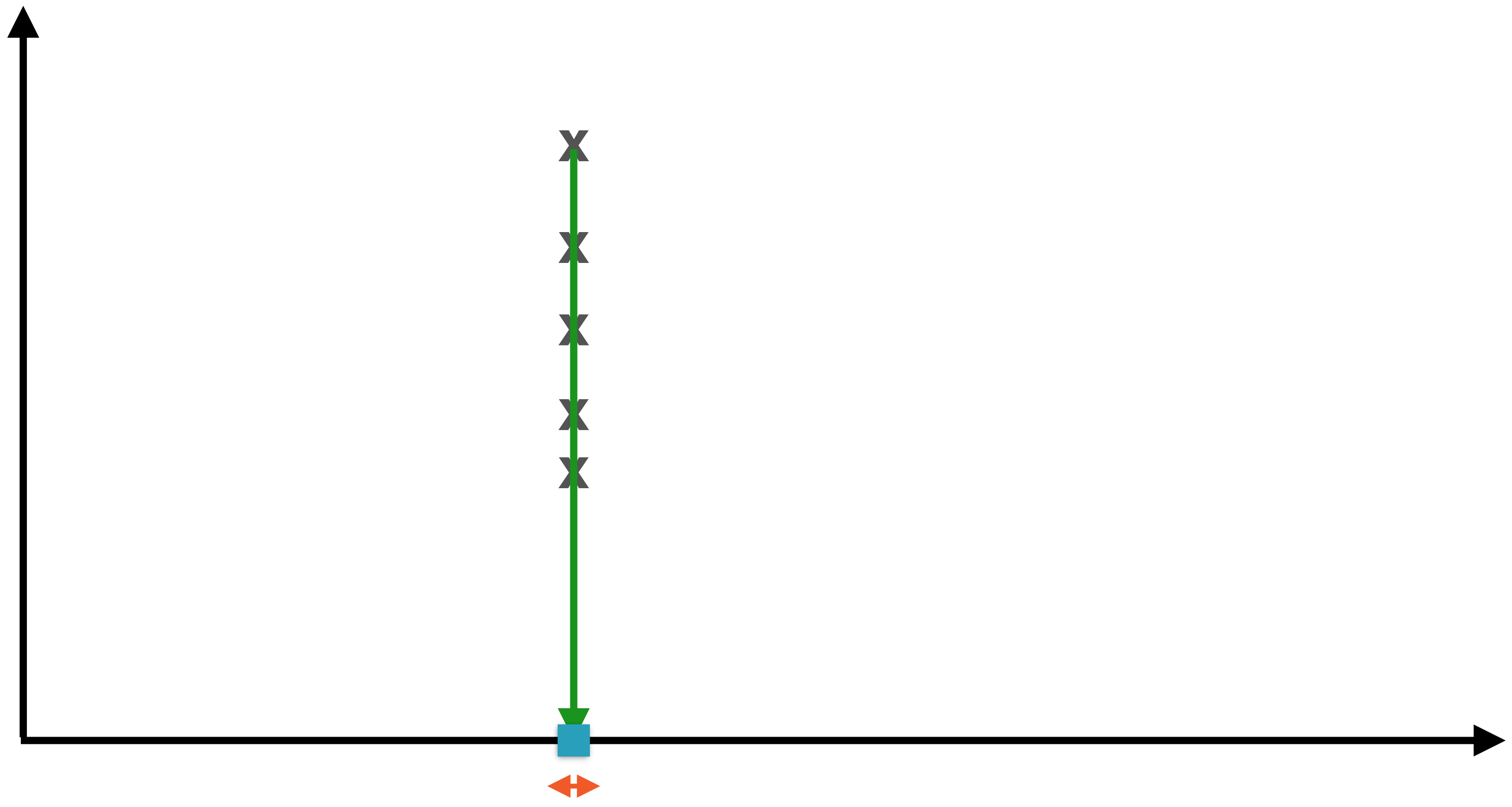
Start by "projecting" the data onto a line in some direction

Intuition Behind PCA



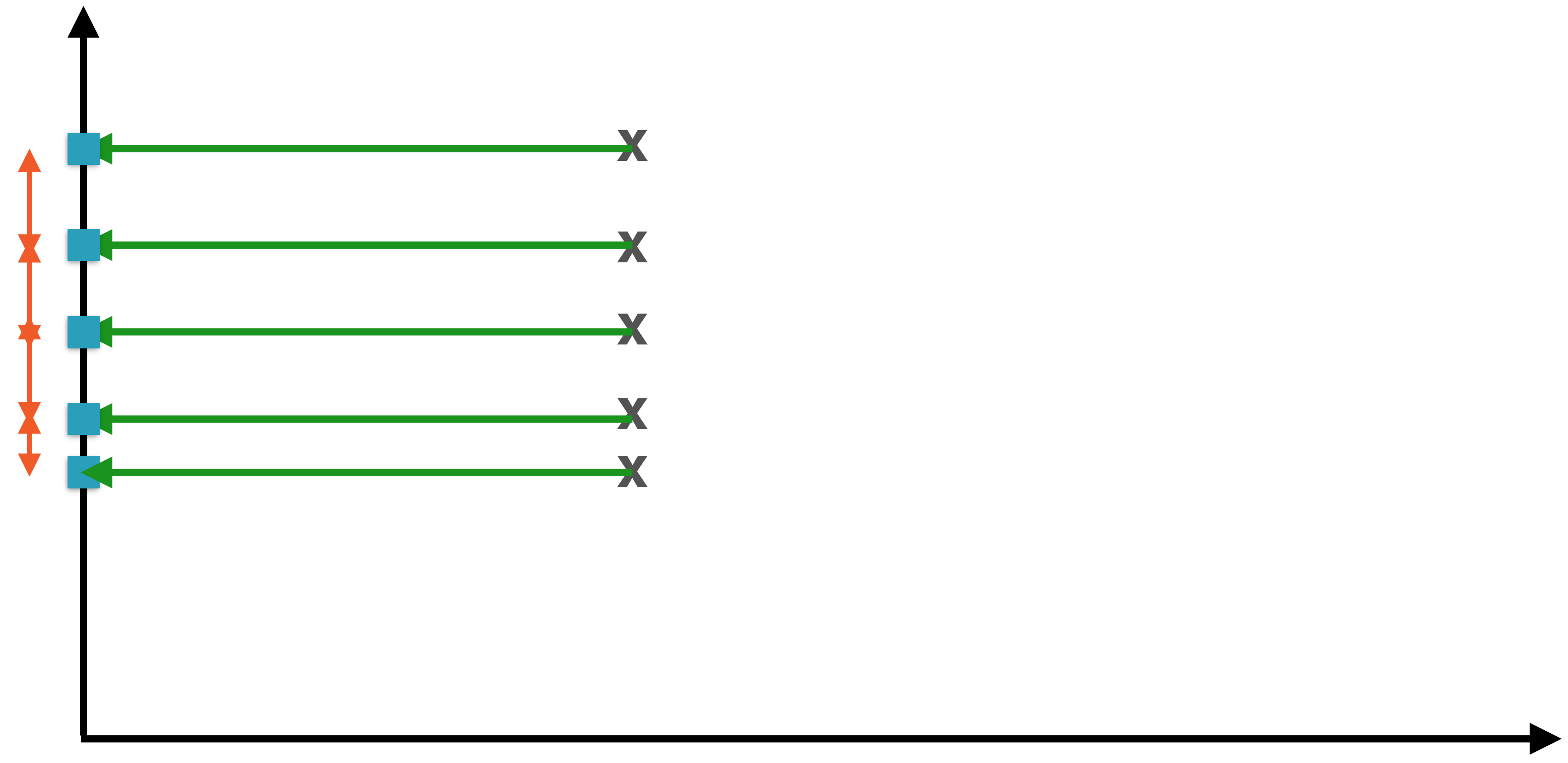
The greater the distances between these projections, the “better” the direction

Bad Projection



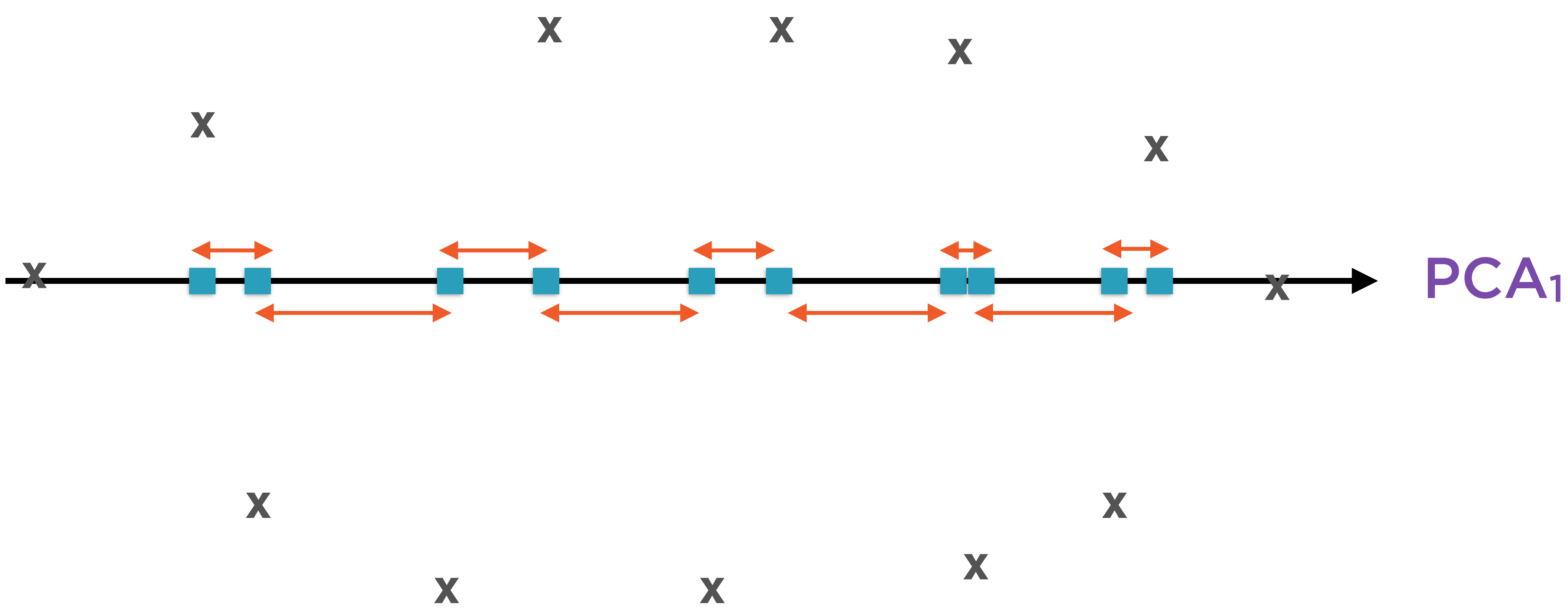
A projection where the distances are minimised is a bad one - information is lost

Good Projection



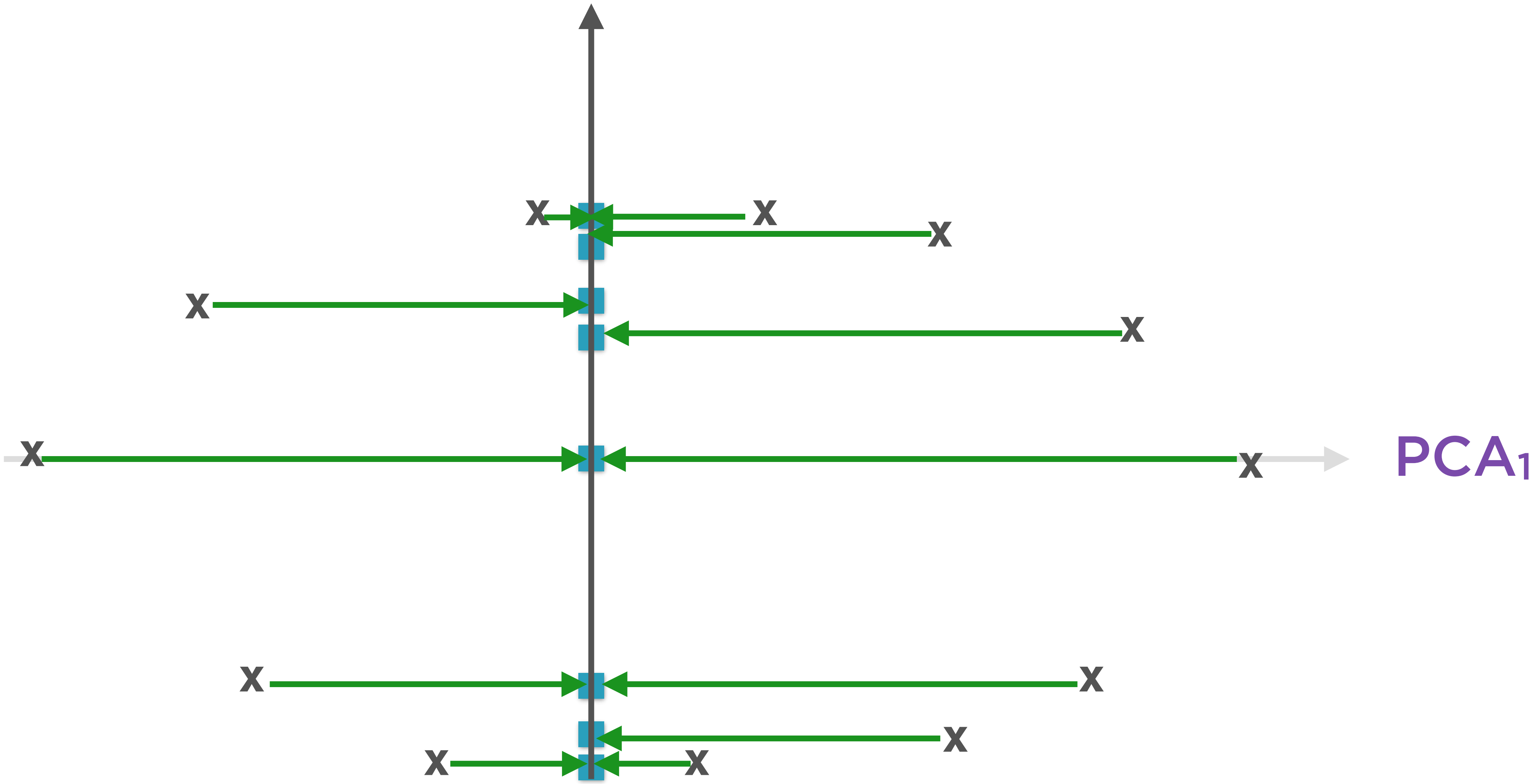
A projection where the distances are maximised is a good one - information is preserved

Intuition Behind PCA



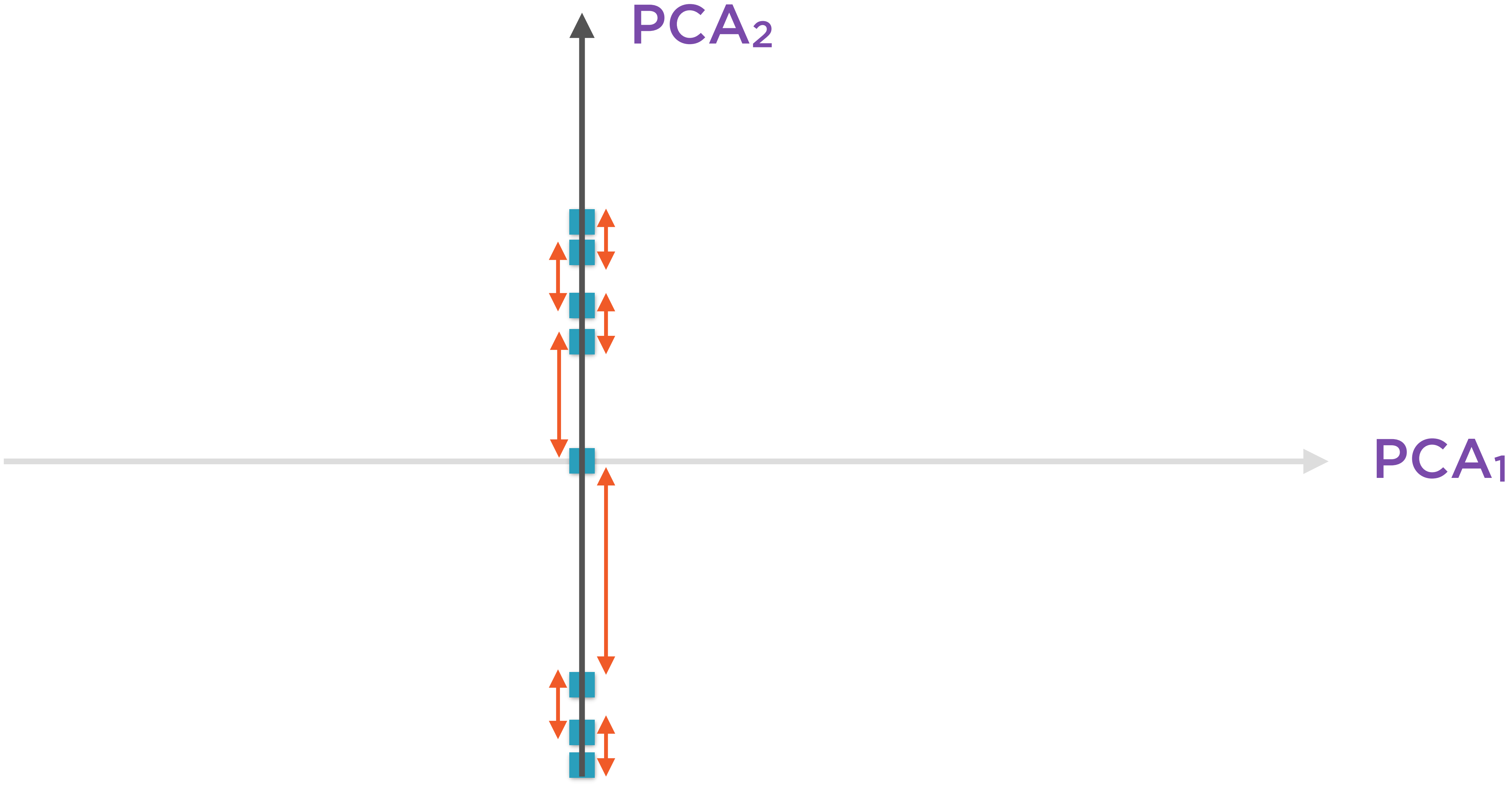
The direction along which this variance is maximised is the first principal component of the original data

Intuition Behind PCA



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

Intuition Behind PCA

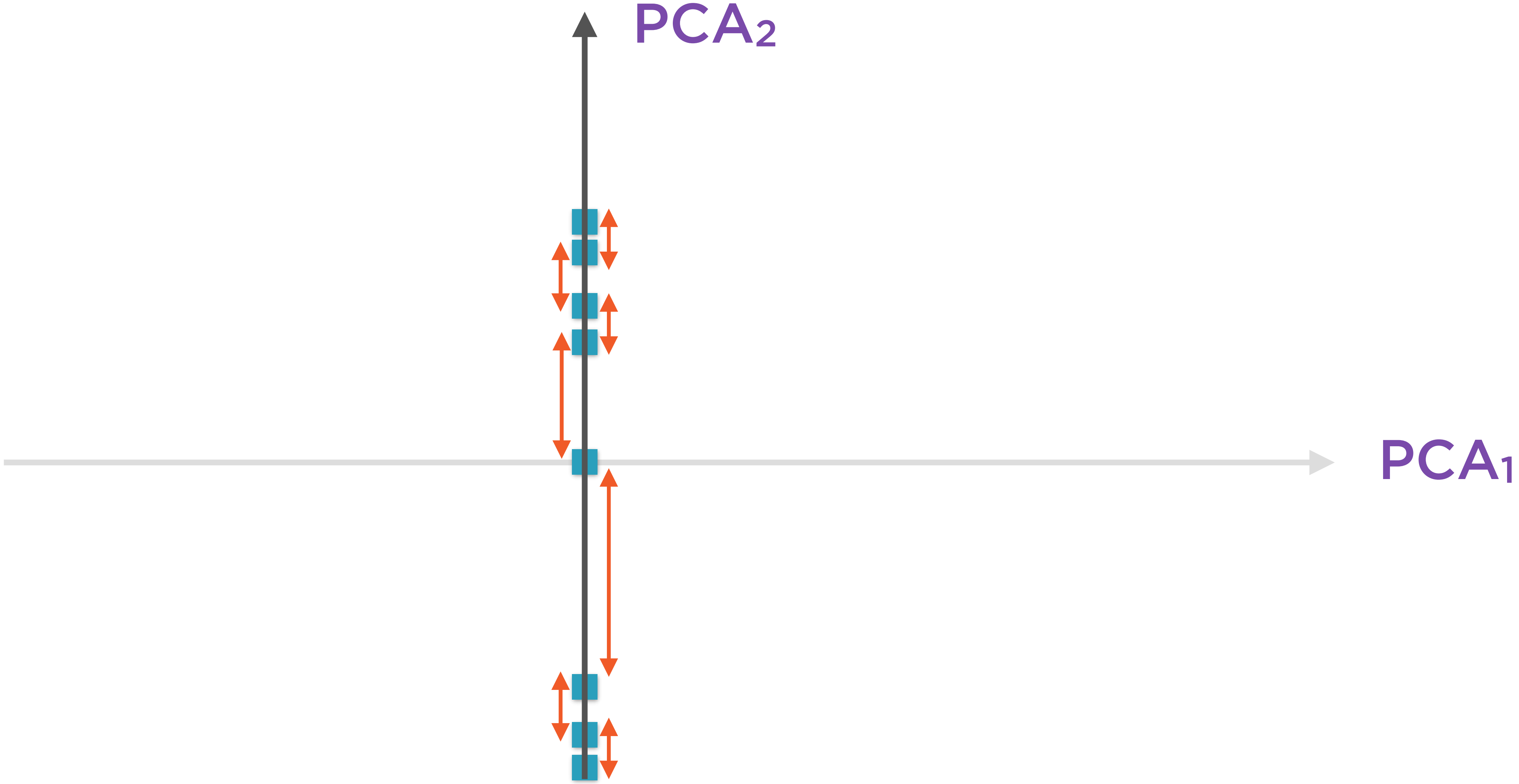


Find the next best direction, the **second principal component**, which must be at right angles to the first

Principal Components at Right Angles

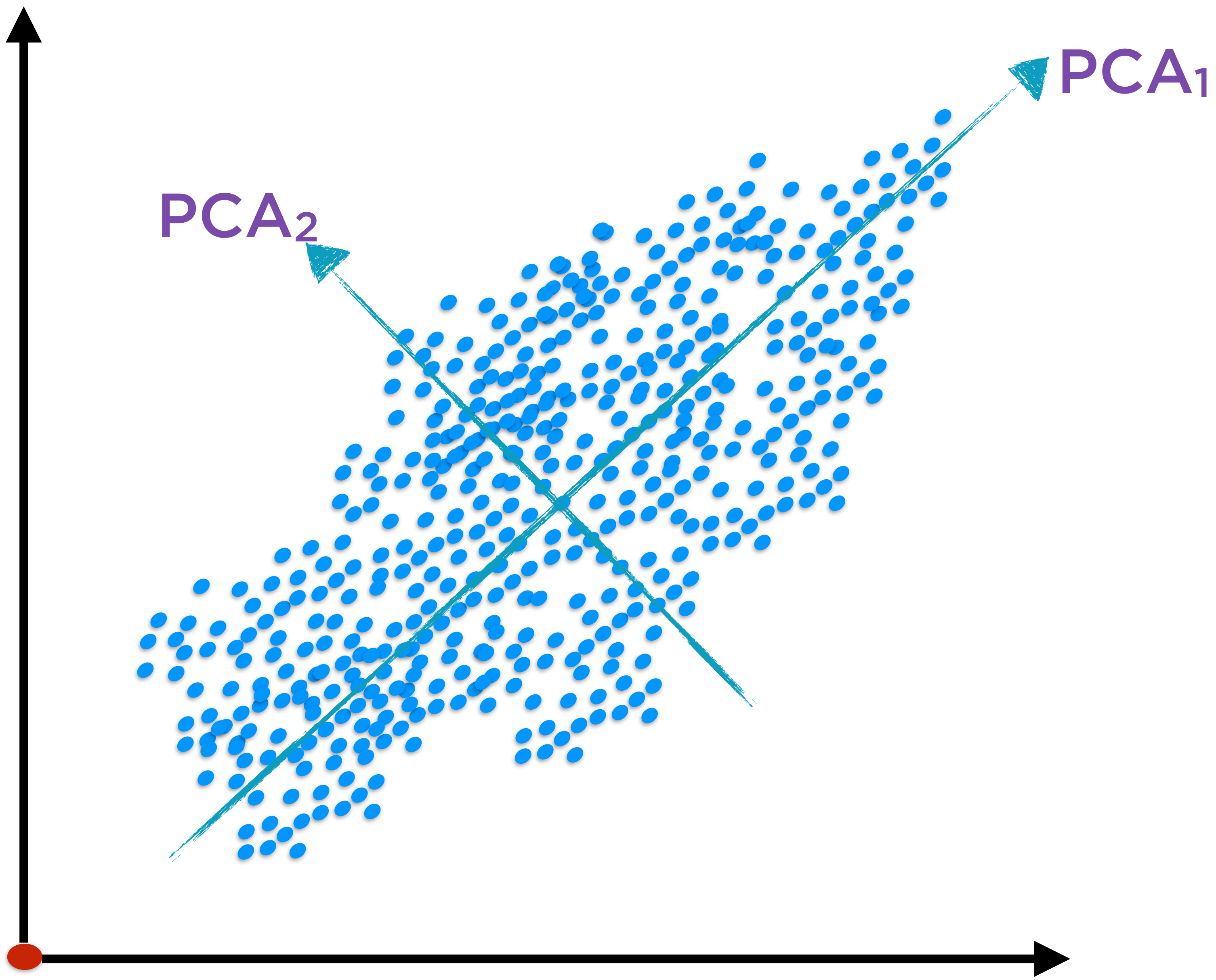
Directions at right angles help express the most variation with the smallest number of directions

Intuition Behind PCA



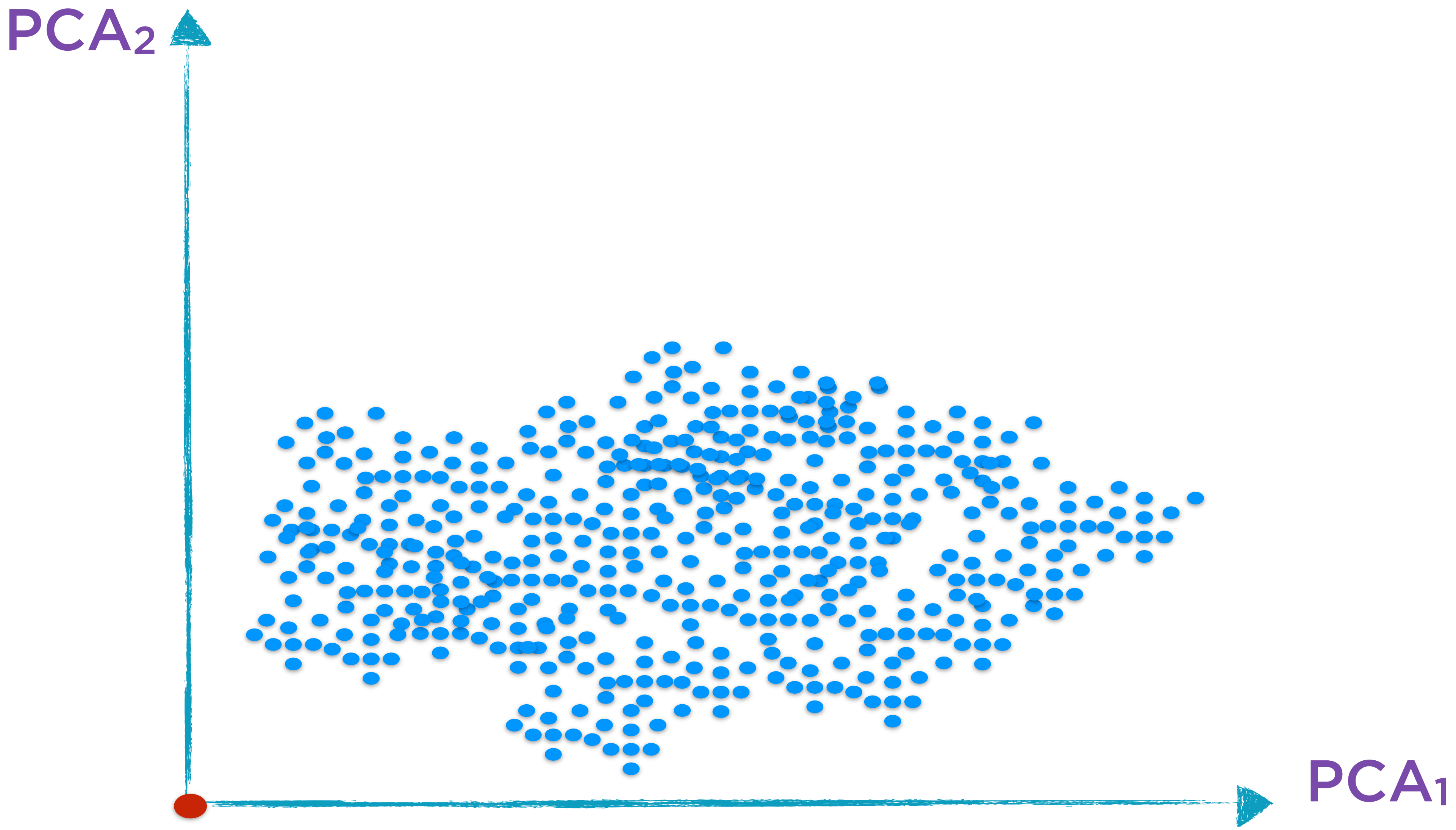
The variances are clearly smaller along this **second principal component** than along the first

Intuition Behind PCA



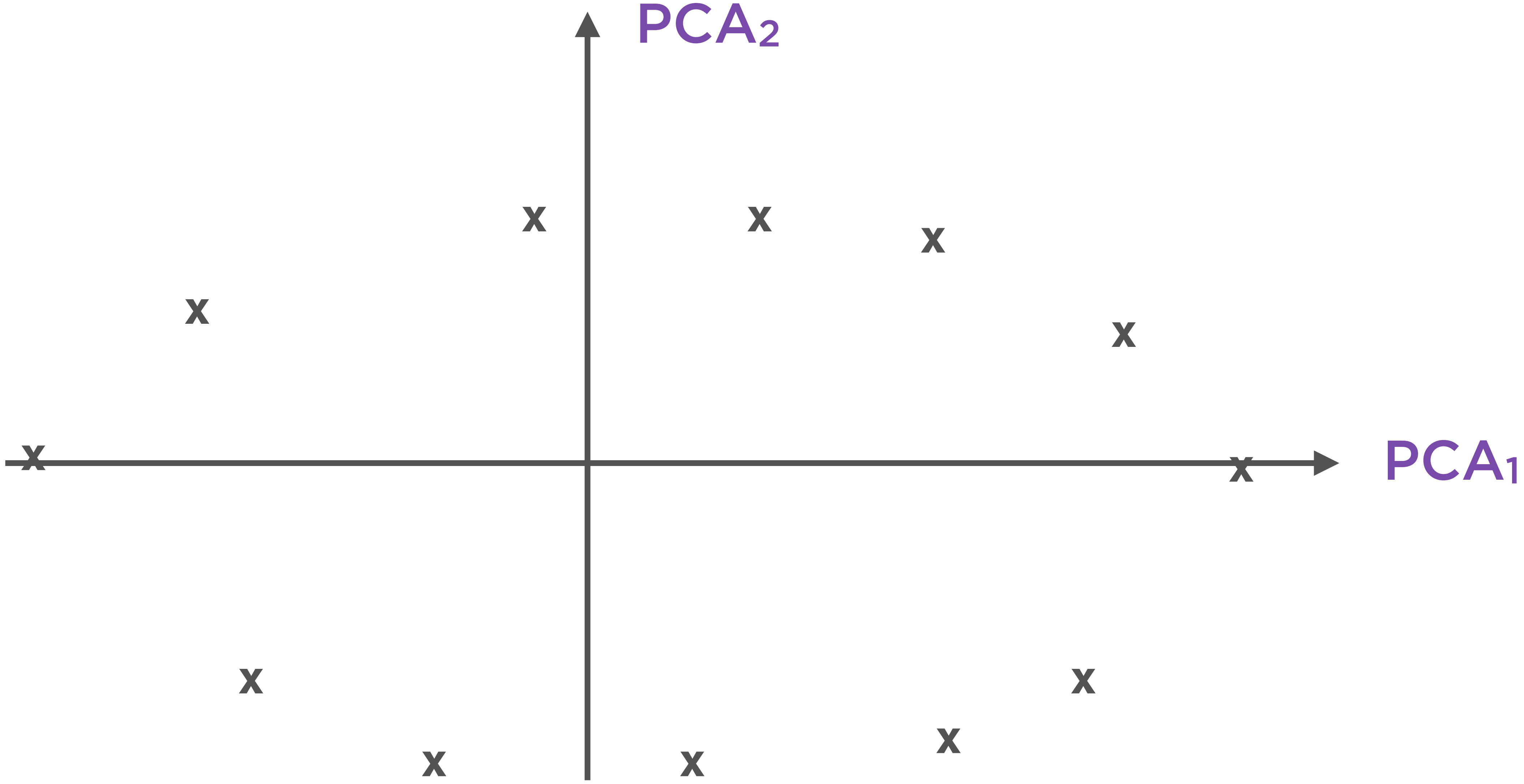
In general, there are as many principal components as there are dimensions in the original data

Intuition Behind PCA



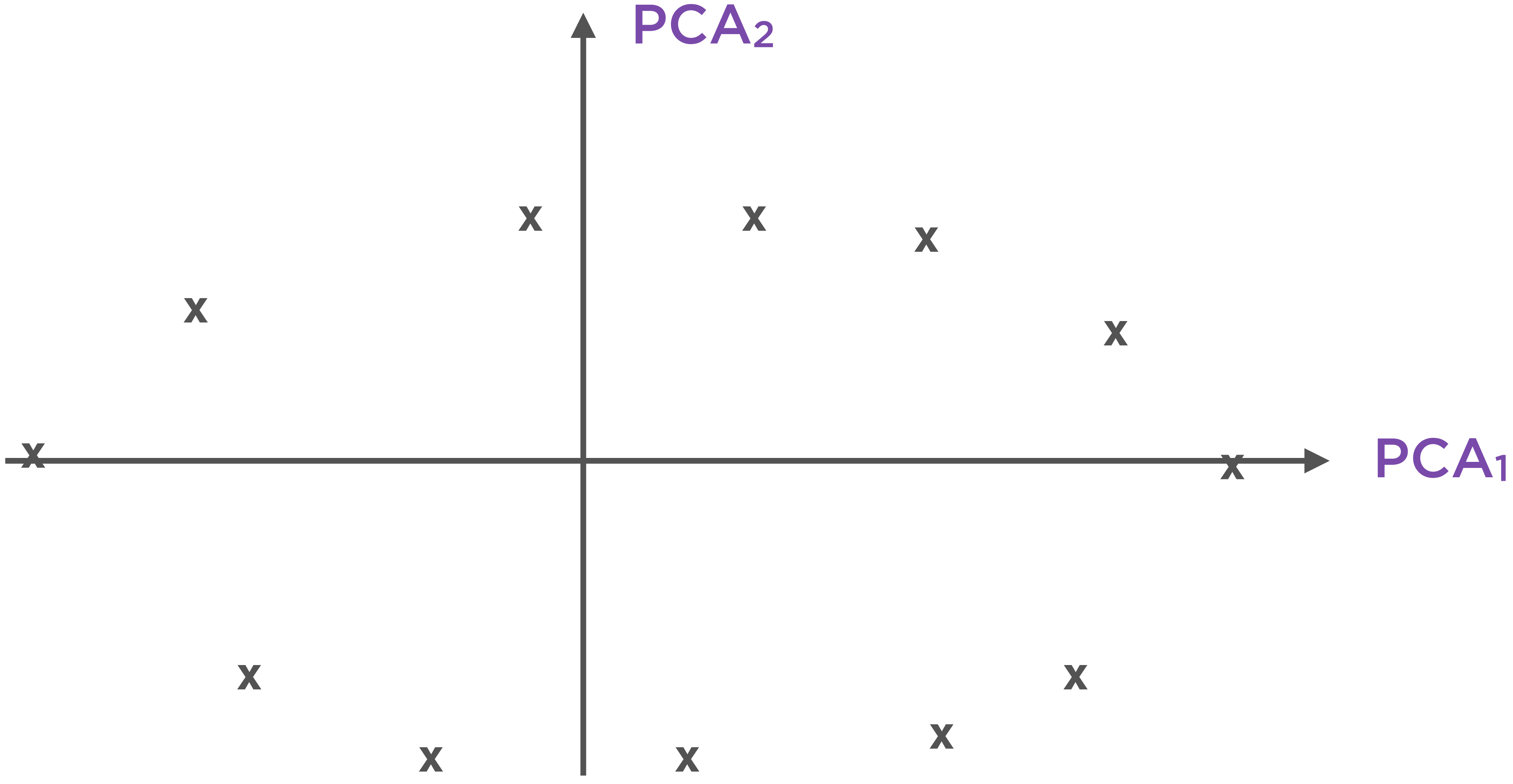
Re-orient the data along these new axes

Dimensionality Reduction



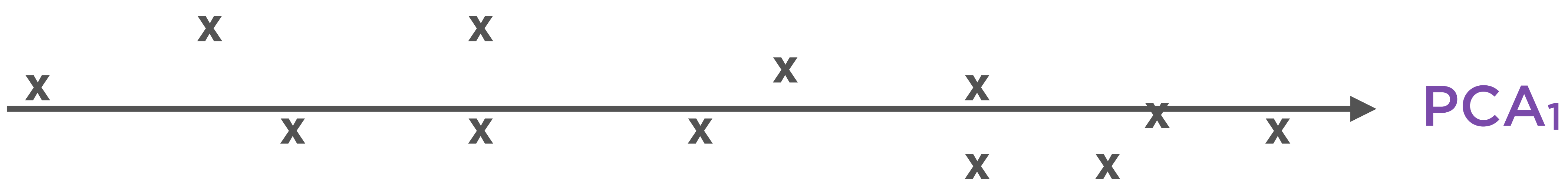
If the variance along the second principal component is small enough, we can just ignore it and use just 1 dimension to represent the data

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 ... 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

$$\underbrace{\begin{bmatrix} x_{11} & & x_{1k} \\ x_{21} & & x_{2k} \\ x_{31} & \cdots & x_{3k} \\ \cdots & & \cdots \\ x_{n1} & & x_{nk} \end{bmatrix}}_{k} \; n \quad \longrightarrow$$

PCA

$$\longrightarrow \quad \underbrace{\begin{bmatrix} f_{11} & & f_{1k} \\ f_{21} & & f_{2k} \\ f_{31} & \cdots & f_{3k} \\ \cdots & & \cdots \\ f_{n1} & & f_{nk} \end{bmatrix}}_{k} \; n$$

Original Data

Same number of columns

Principal Components

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 scikit-learn

Summary

Clustering is an unsupervised learning technique which helps find patterns in data

Common clustering algorithms are k-means, mean-shift clustering

Dimensionality reduction represents inputs in terms of their most significant features

PCA is a very commonly used technique for latent factor analysis

Books

Hands-On Machine Learning with Scikit-Learn and TensorFlow

by Aurélien Géron

Related Courses

How to Think About Machine Learning Algorithms

Understanding Machine Learning with Python

Understanding the Foundations of TensorFlow