Unsupervised Learning Techniques - Clustering

K-Means

Limits of K-Means

Using Clustering for Image Segmentation

Using Clustering for Preprocessing

Using Clustering for Semi-Supervised Learning

DBSCAN

Other Clustering Algorithms

Clustering

Identifying similar instances and assigning them to clusters. No universal definition because it’s context dependent and different algorithms capture different types of clusters.

It has several applications:

- Customer segmentation: Clustering customers based on purchases, website activity, etc. Recommender systems to suggest content to other users in the same cluster.

- Data analysis: Initial analysis to discover similarities among instances.

- Dimensionality reduction: Replacing each instance’s feature vector with a vector of its cluster affinities (how well an instance fits into a cluster). The new vector typically has lower dimensionality without significant information loss.

Clustering cont.

- Anomaly (Outlier) detection: Instances with low affinities to all clusters are likely to be anomalies. Useful for fraud detection.

- Semi-supervised learning: Propagating labels to all instances in the same cluster. Increases the number of labels for supervised learning and potentially increases algorithm performance.

- Search engines: Searching for images similar to a reference image. Cluster all images in a dataset, and return all images from the cluster that the reference image most closely fits to.

- Image segmentation: Cluster pixels according to color and replace each pixel color with the mean cluster color. Used in object detection and tracking systems because it allows for easier object contour detection.

K-Means

Centroid - a point around which instances cluster.

Hard clustering - assigning an instance to a particular cluster.

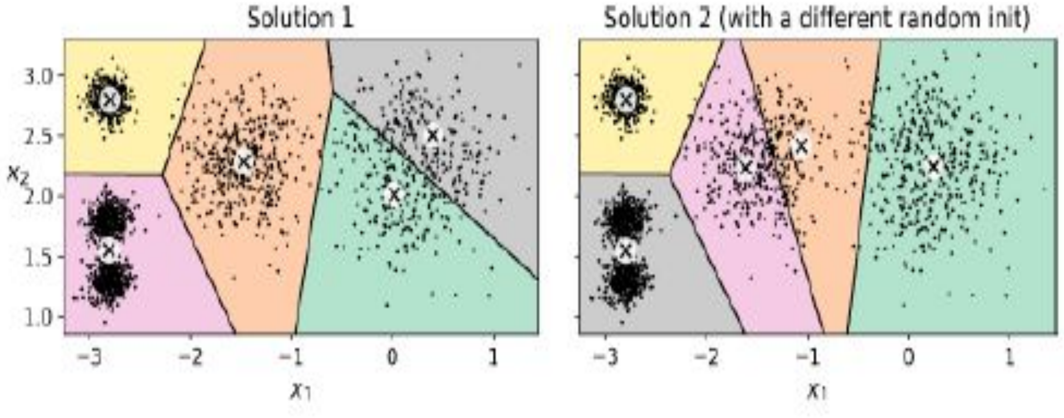
Soft clustering - giving each instance a score per cluster.

K-Means assigns instances based on their distance to a centroid, so it doesn’t work well with clusters of different diameters.

K-Means cont.

When given neither labels or centroids, start by placing k centroids randomly. Label the instances, update the centroid locations and continue repeating until centroids stop moving.

The algorithm is guaranteed to converge in a finite number of steps. However, it may not always converge to the best solution:



K-Means: Centroid Initialization Methods

- If you know where centroids should approximately be placed, the init hyperparameter allows for centroid placement.

- The n_init hyperparameter allows different random initializations. The algorithm keeps the best solution by determining the optimal interia, the mean squared distance between each instance and its closest centroid.

- KMeans runs the algorithm n_init times and keeps the model with the lowest inertia.

- In 2006, Arthur and Vassilvitskii introduced an initialization step that selects centroids distant from one another, making suboptimal solutions less likely. It's the default KMeans initialization method.

K-Means: Accelerated K-Means and Mini-batch K-Means

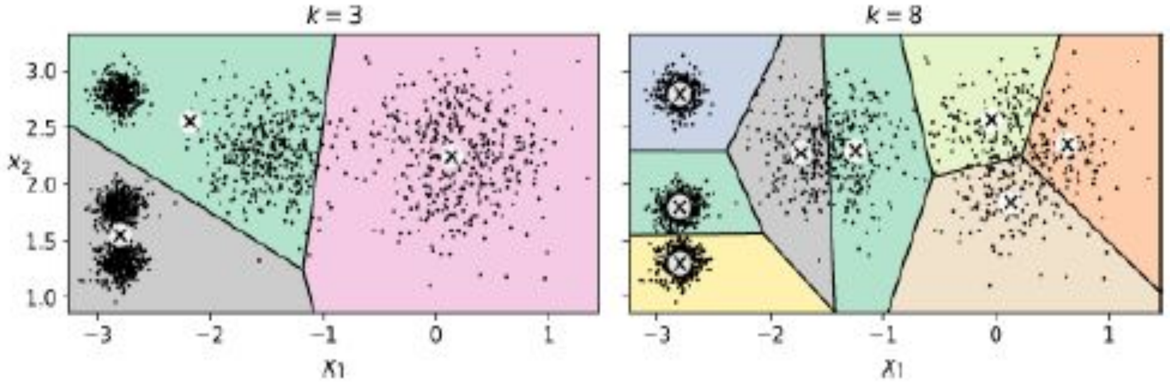
Accelerated K-Means: Proposed by Elkan in 2003, it avoids unnecessary distance calculations by using triangle inequality (straight line is the shortest) and keeping track of lower and upper bounds between instances and centroids. Default algorithm used by KMeans.

Mini-batch K-Means: Uses portion of dataset to move centroids. Slower centroid movement but speeds up the algorithm by a factor of three or four and allows for clustering of large datasets.

Accessed in Scikit-Learn using MiniBatchKMeans class. While it’s faster than regular K-Means, the inertia is slightly worse as the numbers of clusters increase.

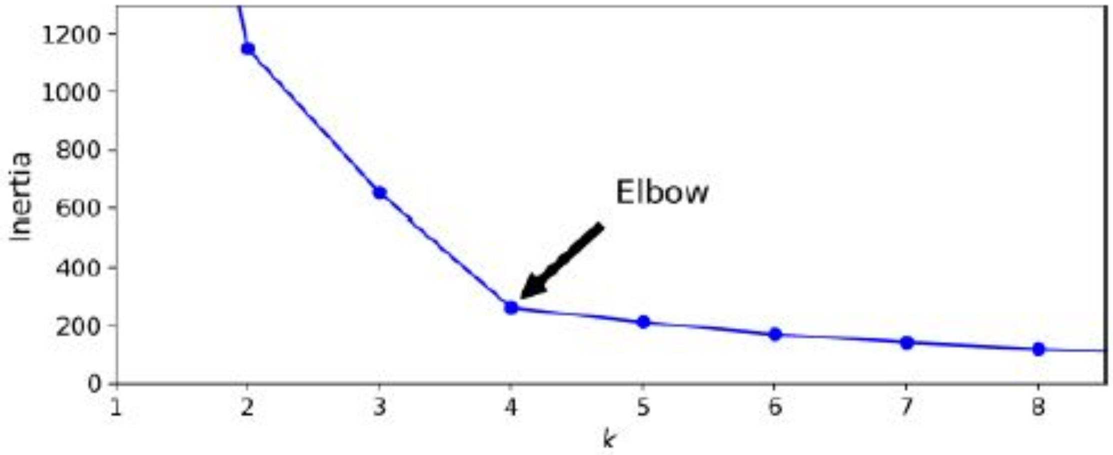
K-Means: Finding the Optimal Number of Clusters

Lowest inertia is not always good a good way to determine the optimal number of clusters. In the example below, the ideal number of clusters is 5, but k=5 has an inertia value of 211.6 while k=8 has a value of 119.1. As the number of clusters increases, inertia decreases because instances will be closer to the closest centroid.



K-Means: Finding the Optimal Number of Clusters cont.

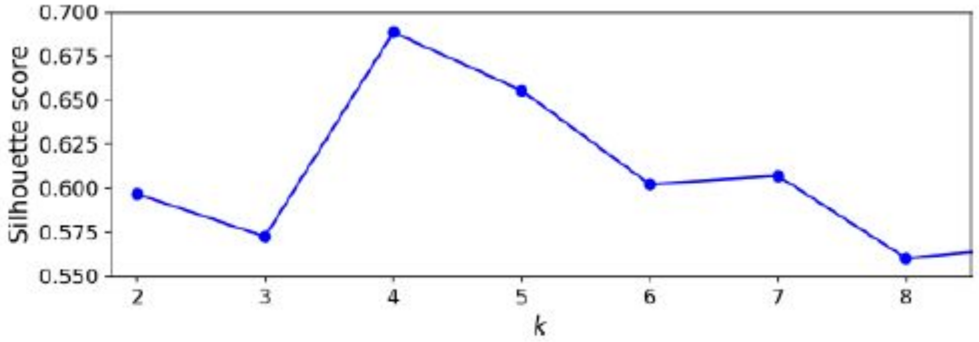
Using the elbow rule, by plotting interia against k, is a coarse way to determine the ideal number of clusters. Choose a value of k based on where the decrease in inertia starts to slow. Looking at the graph below, it seems like k=4 would be a good choice.



K-Means: Finding the Optimal Number of Clusters cont.

Using the silhouette score, which is the mean silhouette coefficient over all instances, is more precise.

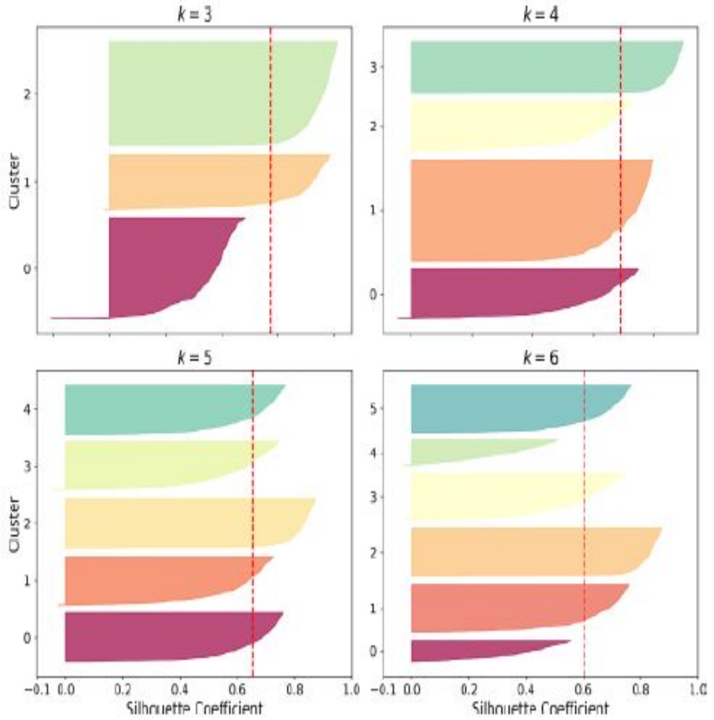
Silhouette coefficient: (b - a) / max(a, b), where a is the mean distance to the other instances in the same cluster (the mean intra-cluster distance) and b is the mean nearest-cluster distance (mean distance to the instances of the next closest cluster). It can vary between -1 (instance may be in the wrong cluster) and +1 (instance is well inside its own cluster). 0 means it's close to a cluster boundary.



K-Means: Finding the Optimal Number of Clusters cont.

Silhouette diagram - dashed lines represent the the silhouette score for each number of clusters. Clusters below the dashed lines have a majority of their instance coefficients below this score.

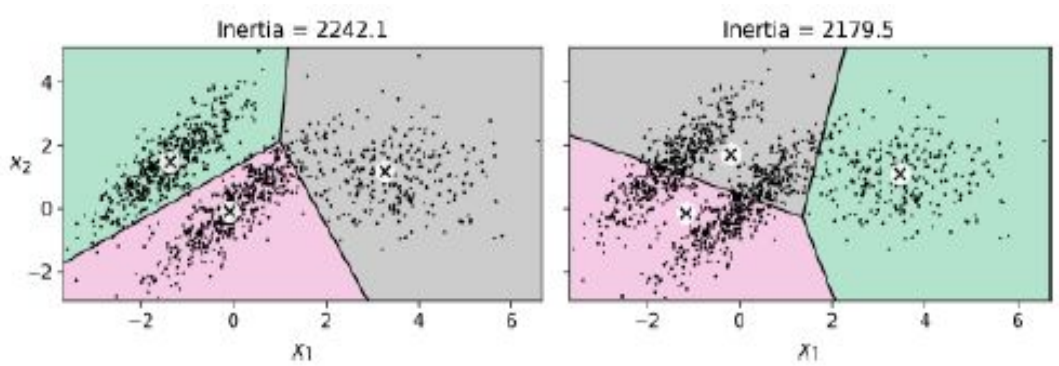
While the silhouette score for k=4 is higher than for k=5, k=5 might be the better option if we're interested in clusters of similar sizes.



Limits of K-Means

Algorithm often requires several runs to avoid suboptimal solutions.

Doesn't work very well when clusters vary in size and/or density or have non-spherical shapes.



Scaling input features before running K-Means generally helps improve cluster shape but doesn't guarantee that they'll be spherical.

Using Clustering for Image Segmentation

Image segmentation - partitioning an image into multiple segments.

Semantic segmentation - all pixels part of the same object type are assigned to the same segment. Ex., pedestrian image is assigned to a pedestrian segment. One segment for all pedestrians.

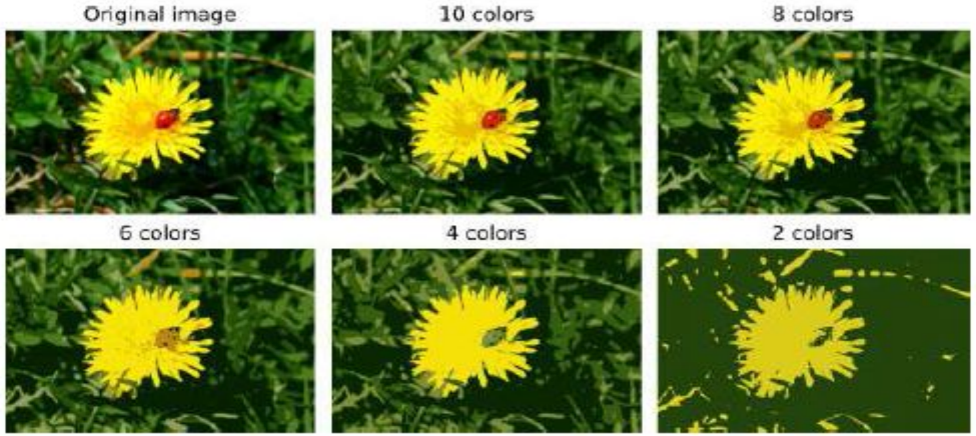
Instance segmentation - all pixels part of the same individual object are assigned to the same segment. Different segment for each pedestrian.

Color segmentation - assign pixels to the same segment if they have similar colors.

Using Clustering for Image Segmentation cont.

The image below has three color channels, but the number can vary depending on the type of image. Grayscale images have one channel, while satellite images can contain multiple channels for light frequencies, like infrared.

After clustering according to color all shades of a particular color can be replaced with one color. In the image below, when the number of clusters is small, K-Means can't distinguish the ladybug from its environment because it prefers clusters of similar sizes.



Using Clustering for Preprocessing

Clustering can be a efficient way to reduce the dimensionality of a dataset.

This can involve using KMeans as part of a supervised learning pipeline, as well as using GridSearchCV to find the optimal number of clusters.

Using Clustering for Semi-Supervised Learning

If only a portion of a dataset has labeled instances, label propagation can be a useful and time saving approach to increasing the number of labeled instances.

Label propagation involves labeling representative images of an initial set of clusters and then expanding that label to either all or a partial set of instances within the same cluster.

Often, a partial propagation can produce more accurate results than a full propagation because instances closer to the centroid of a cluster are more likely to be accurately clustered than those further away.

DBSCAN

DBSCAN defines clusters as high density regions.

For every instance, DBSCAN determines how many other instances are within a distance ε (epsilon), the instance's ε-neighborhood.

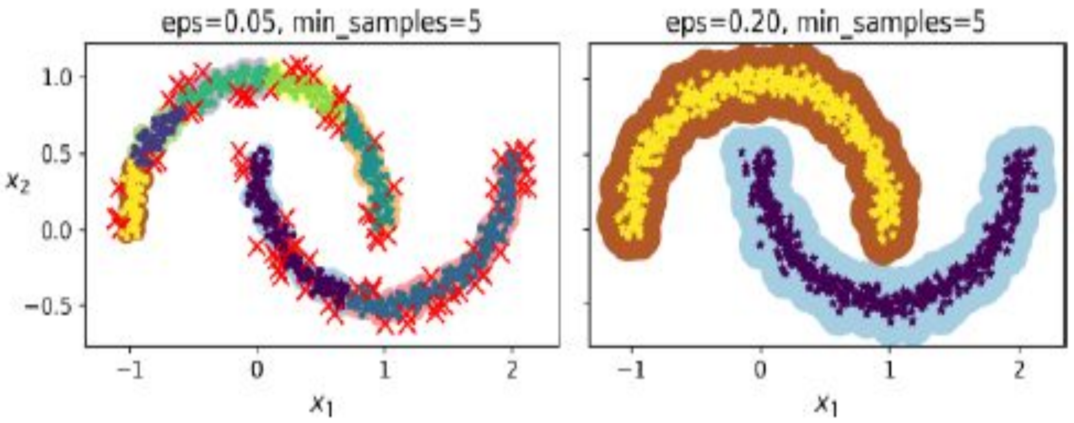
A instance is considered a core instance if there are min_samples number of instances (including itself) in its ε-neighborhood. As such, core instances are located in dense regions.

Any instance in the neighborhood of a core instance is part of the same cluster, including other core instances, Thus, several sequences of neighboring core instances can form a single cluster.

Anomalies are any instances that aren't core instances and that don't have core instances within their neighborhood.

DBSCAN cont.

DBSCAN is a versatile algorithm that can be fit to any number of clusters, a variety of cluster shapes, handles outliers well, and has only two hyperparameters - eps (max distance for two samples to be considered within each other's neighborhood) and min_samples.



Other Clustering Algorithms

Agglomerative clustering - hierarchy of clusters build from the bottom up, starting with individual instances. Scales well to large numbers of instances and clusters and can capture multiple cluster shapes. In order to scale to large datasets, it requires a connectivity matrix - a sparse m by m matrix that indicates which pairs of instances are neighbors.

Birch - designed specifically for large datasets and can be faster than batch K-Means, provided that the number of features is below 20. Builds a tree structure during training with just enough info to assign instances to a cluster without storing all instances in the tree.

Other Clustering Algorithms cont.

Mean-shift - starts by placing a circle on each instance and for every circle computes the mean of all instances located within it, then shifts the circle to center it on the mean. Circles are shifted in the direction of higher density until they each have a local density maximum. All circles that settle in the same place are assigned to the same cluster. It can find any number of clusters for a variety of shapes and has only one hyperparameter - bandwidth (the radius of the circles). It doesn't work well with large datasets due to its high computational complexity.

Affinity propagation - uses a voting system in which instances vote for similar instances to be their representatives. Each representative and its voters for a cluster. It can work well with many different cluster sizes but isn't suited for large datasets due to its computational complexity.

Spectral clustering - creates a similarity matrix between instances and reduces its dimensionality. It then uses another clustering algorithm on this low-dimensional space (K-Means in Scikit-Learn's version). It can capture complex cluster structures but doesn't scale well to large numbers of instances or clusters of different sizes.