Chapter 7: Unsupervised Learning: Clustering

Reza Rezazadegan

Sharif University of Technology

Fall 2022

Introducing Unsupervised Learning

- So far in the course, we were given the data features $(x_1, x_2, \ldots, x_d)$ as well as the labels y.

Introducing Unsupervised Learning

- So far in the course, we were given the data features $(x_1, x_2, \ldots, x_d)$ as well as the labels y. We wanted to know how the label y is conferred from the features.

Introducing Unsupervised Learning

- So far in the course, we were given the data features $(x_1, x_2, \ldots, x_d)$ as well as the labels y. We wanted to know how the label y is conferred from the features.
- In *Unsupervised Learning* we have only the features and not the labels.

Introducing Unsupervised Learning

- So far in the course, we were given the data features $(x_1, x_2, \ldots, x_d)$ as well as the labels y. We wanted to know how the label y is conferred from the features.
- In *Unsupervised Learning* we have only the features and not the labels. We want to obtain information about the data from the features alone.
- In a sense, in unsupervised learning we have to find the labels y ourselves.

Introducing Unsupervised Learning

- So far in the course, we were given the data features $(x_1, x_2, \ldots, x_d)$ as well as the labels y. We wanted to know how the label y is conferred from the features.
- In *Unsupervised Learning* we have only the features and not the labels. We want to obtain information about the data from the features alone.
- In a sense, in unsupervised learning we have to find the labels y ourselves.
- Note that most of the data we have is unlabeled. Labeling data is expensive and time consuming.

Introducing Unsupervised Learning

- So far in the course, we were given the data features $(x_1, x_2, \ldots, x_d)$ as well as the labels y. We wanted to know how the label y is conferred from the features.
- In *Unsupervised Learning* we have only the features and not the labels. We want to obtain information about the data from the features alone.
- In a sense, in unsupervised learning we have to find the labels y ourselves.
- Note that most of the data we have is unlabeled. Labeling data is expensive and time consuming.
- Andrew Ng: "[A] toddler can usually recognize a cat after just one encounter, but a computer still needs more than one example to learn... Effective "unsupervised learning" – learning without labelled data – remains a holy grail of AI."

Unsupervised Learning methods

- **Clustering:** grouping datapoints according to a similarity or closeness criterion.

Unsupervised Learning methods

- **Clustering:** grouping datapoints according to a similarity or closeness criterion.

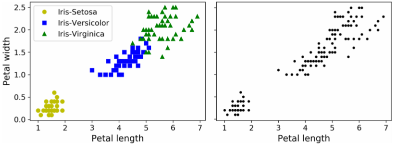


Figure: Classification(left) vs clustering (right). Credit: Aurelien Geron

Unsupervised Learning methods

- **Clustering:** grouping datapoints according to a similarity or closeness criterion.

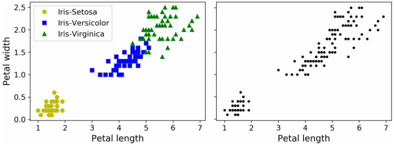


Figure: Classification(left) vs clustering (right). Credit: Aurelien Geron

- **Dimension reduction:** mapping high-dimensional data to lower dimensions in a way that preserves “important” information in data;

Unsupervised Learning methods

- **Clustering:** grouping datapoints according to a similarity or closeness criterion.

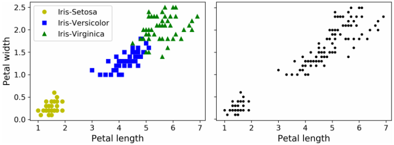


Figure: Classification(left) vs clustering (right). Credit: Aurelien Geron

- **Dimension reduction:** mapping high-dimensional data to lower dimensions in a way that preserves "important" information in data; for visualization or for improving performance of other ML methods.

Unsupervised Learning methods

- **Clustering:** grouping datapoints according to a similarity or closeness criterion.

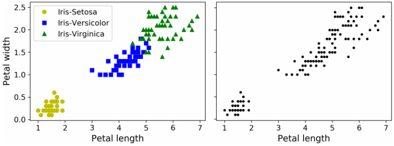


Figure: Classification(left) vs clustering (right). Credit: Aurelien Geron

- **Dimension reduction:** mapping high-dimensional data to lower dimensions in a way that preserves "important" information in data; for visualization or for improving performance of other ML methods.
- **Anomaly or outlier detection:** identifying rare items in data; items that are significantly different from other datapoints.

Unsupervised Learning methods

- **Clustering:** grouping datapoints according to a similarity or closeness criterion.

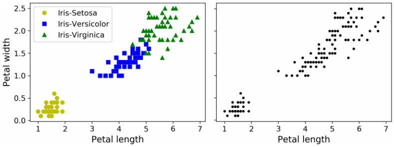


Figure: Classification(left) vs clustering (right). Credit: Aurelien Geron

- **Dimension reduction:** mapping high-dimensional data to lower dimensions in a way that preserves "important" information in data; for visualization or for improving performance of other ML methods.
- **Anomaly or outlier detection:** identifying rare items in data; items that are significantly different from other datapoints. For example finding faulty products from their images.

Unsupervised Learning methods

- **Clustering:** grouping datapoints according to a similarity or closeness criterion.

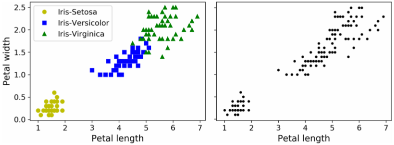


Figure: Classification(left) vs clustering (right). Credit: Aurelien Geron

- **Dimension reduction:** mapping high-dimensional data to lower dimensions in a way that preserves "important" information in data; for visualization or for improving performance of other ML methods.
- **Anomaly or outlier detection:** identifying rare items in data; items that are significantly different from other datapoints. For example finding faulty products from their images. Includes *fraud detection* in finance.

Unsupervised Learning methods

- **Clustering:** grouping datapoints according to a similarity or closeness criterion.

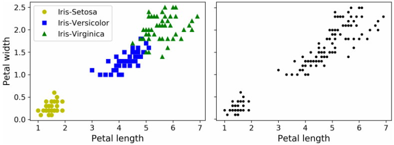


Figure: Classification(left) vs clustering (right). Credit: Aurelien Geron

- **Dimension reduction:** mapping high-dimensional data to lower dimensions in a way that preserves "important" information in data; for visualization or for improving performance of other ML methods.
- **Anomaly or outlier detection:** identifying rare items in data; items that are significantly different from other datapoints. For example finding faulty products from their images. Includes *fraud detection* in finance.

Unsupervised Learning methods cont.

- **Topological Data Analysis:** obtaining insights from the shape (topology) of data.

Unsupervised Learning methods cont.

- **Topological Data Analysis:** obtaining insights from the shape (topology) of data.
- **Word embeddings:** obtaining vector representations of natural language words in a way that words which usually occur together, have close vectors.

Methods related to Unsupervised Learning

Methods related to Unsupervised Learning

- Unsupervised methods can be combined with supervised methods such as:

Methods related to Unsupervised Learning

- Unsupervised methods can be combined with supervised methods such as:
- **Semi-supervised Learning:** using unsupervised methods such as clustering to aid classification.

Methods related to Unsupervised Learning

- Unsupervised methods can be combined with supervised methods such as:
- **Semi-supervised Learning:** using unsupervised methods such as clustering to aid classification. For example, if we have data in which only few instances are labeled, we can cluster the dataset and generalize the labels to all the instances in a cluster.

Methods related to Unsupervised Learning

- Unsupervised methods can be combined with supervised methods such as:
- **Semi-supervised Learning:** using unsupervised methods such as clustering to aid classification. For example, if we have data in which only few instances are labeled, we can cluster the dataset and generalize the labels to all the instances in a cluster.

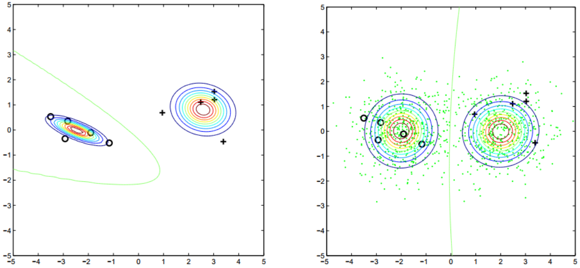


Figure: Learning from labeled data alone (left) vs learning from labeled and unlabeled data (right). Credit: Xiaojin Zhu, Semi-supervised Learning

- **Contrastive Learning:** obtaining vector representation $\Phi(\mathbf{x}) \in \mathbb{R}^n$ of data (e.g. images or texts) in such a way that minimizes the distance between the representation of $\mathbf{x}$ and those of its transformations.

Methods related to Unsupervised Learning cont.

- **Contrastive Learning:** obtaining vector representation $\Phi(\mathbf{x}) \in \mathbb{R}^n$ of data (e.g. images or texts) in such a way that minimizes the distance between the representation of $\mathbf{x}$ and those of its transformations. Doing this, the algorithm learns that an image of a cat should have a different representation that the image of a dog.

Methods related to Unsupervised Learning cont.

- **Contrastive Learning:** obtaining vector representation $\Phi(\mathbf{x}) \in \mathbb{R}^n$ of data (e.g. images or texts) in such a way that minimizes the distance between the representation of $\mathbf{x}$ and those of its transformations. Doing this, the algorithm learns that an image of a cat should have a different representation that the image of a dog.
- **Few-shot Learning:** training a classifier with only a few training instances. (Small Data)

Methods related to Unsupervised Learning cont.

- **Contrastive Learning:** obtaining vector representation $\Phi(\mathbf{x}) \in \mathbb{R}^n$ of data (e.g. images or texts) in such a way that minimizes the distance between the representation of **x** and those of its transformations. Doing this, the algorithm learns that an image of a cat should have a different representation that the image of a dog.
- **Few-shot Learning:** training a classifier with only a few training instances. (Small Data)
- **Zero-shot Learning:** classifying unseen classes without any training examples!

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.
- **Data Analysis:** clustering the data first and analyzing the clusters separately is often easier than studying the whole dataset.

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.
- **Data Analysis:** clustering the data first and analyzing the clusters separately is often easier than studying the whole dataset.
- **Outlier detection:** clustering can reveal which datapoints are outliers, i.e. do not belong to any major cluster.

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.
- **Data Analysis:** clustering the data first and analyzing the clusters separately is often easier than studying the whole dataset.
- **Outlier detection:** clustering can reveal which datapoints are outliers, i.e. do not belong to any major cluster.
- **Dimensionality reduction:** after clustering, we can replace a sample's feature vector with the vector of its *affinities* to the clusters.

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.
- **Data Analysis:** clustering the data first and analyzing the clusters separately is often easier than studying the whole dataset.
- **Outlier detection:** clustering can reveal which datapoints are outliers, i.e. do not belong to any major cluster.
- **Dimensionality reduction:** after clustering, we can replace a sample's feature vector with the vector of its *affinities* to the clusters.
- **Semi-supervised Learning:** mentioned above.

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.
- **Data Analysis:** clustering the data first and analyzing the clusters separately is often easier than studying the whole dataset.
- **Outlier detection:** clustering can reveal which datapoints are outliers, i.e. do not belong to any major cluster.
- **Dimensionality reduction:** after clustering, we can replace a sample's feature vector with the vector of its *affinities* to the clusters.
- **Semi-supervised Learning:** mentioned above.
- **Image-based search:** clustering a database of images first, we can see to which cluster a new image belongs.

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.
- **Data Analysis:** clustering the data first and analyzing the clusters separately is often easier than studying the whole dataset.
- **Outlier detection:** clustering can reveal which datapoints are outliers, i.e. do not belong to any major cluster.
- **Dimensionality reduction:** after clustering, we can replace a sample's feature vector with the vector of its *affinities* to the clusters.
- **Semi-supervised Learning:** mentioned above.
- **Image-based search:** clustering a database of images first, we can see to which cluster a new image belongs.
- **Color quantization:** reducing the number of colors in an image by clustering the pixel colors in it. Each color is regarded as a point in the 3-dimensional RGB space.

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.
- **Data Analysis:** clustering the data first and analyzing the clusters separately is often easier than studying the whole dataset.
- **Outlier detection:** clustering can reveal which datapoints are outliers, i.e. do not belong to any major cluster.
- **Dimensionality reduction:** after clustering, we can replace a sample's feature vector with the vector of its *affinities* to the clusters.
- **Semi-supervised Learning:** mentioned above.
- **Image-based search:** clustering a database of images first, we can see to which cluster a new image belongs.
- **Color quantization:** reducing the number of colors in an image by clustering the pixel colors in it. Each color is regarded as a point in the 3-dimensional RGB space. We then replace each color with the mean of the cluster it belongs to.

Applications of clustering

- **Behavioral Customer segmentation**: you can cluster your customers according to their purchasing data and then make offers and recommendations to them accordingly.
- **Data Analysis:** clustering the data first and analyzing the clusters separately is often easier than studying the whole dataset.
- **Outlier detection:** clustering can reveal which datapoints are outliers, i.e. do not belong to any major cluster.
- **Dimensionality reduction:** after clustering, we can replace a sample's feature vector with the vector of its *affinities* to the clusters.
- **Semi-supervised Learning:** mentioned above.
- **Image-based search:** clustering a database of images first, we can see to which cluster a new image belongs.
- **Color quantization:** reducing the number of colors in an image by clustering the pixel colors in it. Each color is regarded as a point in the 3-dimensional RGB space. We then replace each color with the mean of the cluster it belongs to. Used in image compression and image segmentation.

K-Means Clustering

- Imagine we have an unlabeled dataset $D = \{\mathbf{x}_i\}$.

K-Means Clustering

- Imagine we have an unlabeled dataset $D = \{\mathbf{x}_i\}$.
- The k-Means algorithm divides this dataset into k different clusters $C_1, C_2, \ldots, C_k$ where k is a hyperparameter.

K-Means Clustering

- Imagine we have an unlabeled dataset $D = \{\mathbf{x}_i\}$.
- The k-Means algorithm divides this dataset into k different clusters $C_1, C_2, \ldots, C_k$ where k is a hyperparameter. (Later we will see how to systematically choose k.)

K-Means Clustering

- Imagine we have an unlabeled dataset $D = \{\mathbf{x}_i\}$.
- The k-Means algorithm divides this dataset into k different clusters $C_1, C_2, \ldots, C_k$ where k is a hyperparameter. (Later we will see how to systematically choose k.)
- The goal of k-Means method is to find the clusters that minimize the *inertia* or *within-cluster sum of squares:*

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_i||^2 \tag{1}$$

where μ_i is the mean (a.k.a. *centroid*) of C_i.

K-Means Clustering

- Imagine we have an unlabeled dataset $D = \{\mathbf{x}_i\}$.
- The k-Means algorithm divides this dataset into k different clusters $C_1, C_2, \ldots, C_k$ where k is a hyperparameter. (Later we will see how to systematically choose k.)
- The goal of k-Means method is to find the clusters that minimize the *inertia* or *within-cluster sum of squares:*

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_i||^2 \quad (1)$$

  where μ_i is the mean (a.k.a. *centroid*) of C_i.
- Exercise: show that this problem is equivalent to minimizing $\sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{\mathbf{x},\mathbf{y} \in C_i} ||\mathbf{x} - \mathbf{y}||^2$. And this is in turn equivalent to maximizing the sum of squares of distances between points in different clusters.

K-Means Clustering

- Imagine we have an unlabeled dataset $D = \{\mathbf{x}_i\}$.
- The k-Means algorithm divides this dataset into k different clusters $C_1, C_2, \ldots, C_k$ where k is a hyperparameter. (Later we will see how to systematically choose k.)
- The goal of k-Means method is to find the clusters that minimize the *inertia* or *within-cluster sum of squares:*

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_i||^2 \tag{1}$$

  where μ_i is the mean (a.k.a. *centroid*) of C_i.
- Exercise: show that this problem is equivalent to minimizing $\sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{\mathbf{x},\mathbf{y} \in C_i} ||\mathbf{x} - \mathbf{y}||^2$. And this is in turn equivalent to maximizing the sum of squares of distances between points in different clusters.
- However this problem is NP-hard!

K-Means Clustering

- Imagine we have an unlabeled dataset $D = \{\mathbf{x}_i\}$.
- The k-Means algorithm divides this dataset into k different clusters $C_1, C_2, \ldots, C_k$ where k is a hyperparameter. (Later we will see how to systematically choose k.)
- The goal of k-Means method is to find the clusters that minimize the *inertia* or *within-cluster sum of squares:*

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_i||^2 \tag{1}$$

  where μ_i is the mean (a.k.a. *centroid*) of C_i.
- Exercise: show that this problem is equivalent to minimizing $\sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{\mathbf{x},\mathbf{y} \in C_i} ||\mathbf{x} - \mathbf{y}||^2$. And this is in turn equivalent to maximizing the sum of squares of distances between points in different clusters.
- However this problem is NP-hard! Therefore we use an iterative method to find a near-optimum.

K-Means Clustering Algorithm

- At first, k random points in the dataset are chosen, each one considered as a cluster.

K-Means Clustering Algorithm

- At first, k random points in the dataset are chosen, each one considered as a cluster.
- Each point $\mathbf{x}_i$ is assigned to the cluster C_i whose mean is closest to $\mathbf{x}_i$.

K-Means Clustering Algorithm

- At first, k random points in the dataset are chosen, each one considered as a cluster.
- Each point $\mathbf{x}_i$ is assigned to the cluster C_i whose mean is closest to $\mathbf{x}_i$.
- Then the means of the new clusters are computed and the assignment of points to the clusters is done anew.

K-Means Clustering Algorithm

- At first, k random points in the dataset are chosen, each one considered as a cluster.
- Each point $\mathbf{x}_i$ is assigned to the cluster C_i whose mean is closest to $\mathbf{x}_i$.
- Then the means of the new clusters are computed and the assignment of points to the clusters is done anew.
- This process is repeated until the assignments are not changed anymore.

K-Means Clustering Algorithm

- At first, k random points in the dataset are chosen, each one considered as a cluster.
- Each point $\mathbf{x}_i$ is assigned to the cluster C_i whose mean is closest to $\mathbf{x}_i$.
- Then the means of the new clusters are computed and the assignment of points to the clusters is done anew.
- This process is repeated until the assignments are not changed anymore.
- At each iteration, the loss in equation (1) gets smaller.

Limitations of K-Means Clustering

- The decision boundaries between K-means clusters are linear.

Limitations of K-Means Clustering

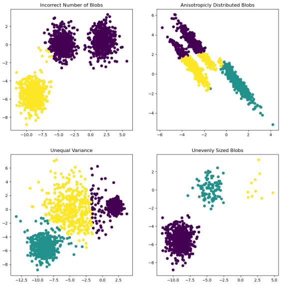
- The decision boundaries between K-means clusters are linear. Thus it cannot separate clusters that are not linearly separable.

Limitations of K-Means Clustering

- The decision boundaries between K-means clusters are linear. Thus it cannot separate clusters that are not linearly separable.

Limitations of K-Means Clustering

- The decision boundaries between K-means clusters are linear. Thus it cannot separate clusters that are not linearly separable.
- It does not behave well when the would-be clusters are not spherically shaped or have different densities.



Limitations of K-Means Clustering cont.

- It can fall into a local minimum.

Limitations of K-Means Clustering cont.

- It can fall into a local minimum. For example if the initial points miss some of the would-be clusters.

Limitations of K-Means Clustering cont.

- It can fall into a local minimum. For example if the initial points miss some of the would-be clusters.
- To remedy this, one can Run the algorithm several times with different initial points and choose the best one

Limitations of K-Means Clustering cont.

- It can fall into a local minimum. For example if the initial points miss some of the would-be clusters.
- To remedy this, one can Run the algorithm several times with different initial points and choose the best one according to the evaluation metrics below.

Limitations of K-Means Clustering cont.

- It can fall into a local minimum. For example if the initial points miss some of the would-be clusters.
- To remedy this, one can Run the algorithm several times with different initial points and choose the best one according to the evaluation metrics below.
- However unlike other clustering methods, the Scikit-Learn class for K-Means has a `predict` method.

Cluster evaluation metrics

To be able to evaluate the goodness of clusters $C_1, C_2, \ldots, C_k$ obtained using a clustering algorithm, we can use the following metrics.

- **Inertia:** the within-cluster sum of squares.

Cluster evaluation metrics

To be able to evaluate the goodness of clusters $C_1, C_2, \ldots, C_k$ obtained using a clustering algorithm, we can use the following metrics.

- **Inertia:** the within-cluster sum of squares.The problem with Inertia is that it assumes the clusters are spherically shaped.

Cluster evaluation metrics

To be able to evaluate the goodness of clusters $C_1, C_2, \ldots, C_k$ obtained using a clustering algorithm, we can use the following metrics.

- **Inertia:** the within-cluster sum of squares.The problem with Inertia is that it assumes the clusters are spherically shaped.
- **The silhouette score** of an instance **x** is $(b - a)/max(a, b)$ where b is the average distance of **x** to the points in its own cluster and b is the average distance to the points in the nearest cluster.

Cluster evaluation metrics

To be able to evaluate the goodness of clusters $C_1, C_2, \dots, C_k$ obtained using a clustering algorithm, we can use the following metrics.

- **Inertia:** the within-cluster sum of squares.The problem with Inertia is that it assumes the clusters are spherically shaped.
- **The silhouette score** of an instance **x** is $(b - a)/max(a, b)$ where b is the average distance of **x** to the points in its own cluster and b is the average distance to the points in the nearest cluster.
- Silhouette coefficient is between -1 and 1.

Cluster evaluation metrics

To be able to evaluate the goodness of clusters $C_1, C_2, \ldots, C_k$ obtained using a clustering algorithm, we can use the following metrics.

- **Inertia:** the within-cluster sum of squares.The problem with Inertia is that it assumes the clusters are spherically shaped.
- **The silhouette score** of an instance **x** is $(b - a)/max(a, b)$ where b is the average distance of **x** to the points in its own cluster and b is the average distance to the points in the nearest cluster.
- Silhouette coefficient is between -1 and 1. If its near 1 then **x** is well inside its cluster.

Cluster evaluation metrics

To be able to evaluate the goodness of clusters $C_1, C_2, \ldots, C_k$ obtained using a clustering algorithm, we can use the following metrics.

- **Inertia:** the within-cluster sum of squares.The problem with Inertia is that it assumes the clusters are spherically shaped.
- **The silhouette score** of an instance $\mathbf{x}$ is $(b - a)/max(a, b)$ where b is the average distance of $\mathbf{x}$ to the points in its own cluster and b is the average distance to the points in the nearest cluster.
- Silhouette coefficient is between -1 and 1. If its near 1 then $\mathbf{x}$ is well inside its cluster. If its near zero then it is near the boundary of its cluster

Cluster evaluation metrics

To be able to evaluate the goodness of clusters $C_1, C_2, \dots, C_k$ obtained using a clustering algorithm, we can use the following metrics.

- **Inertia:** the within-cluster sum of squares.The problem with Inertia is that it assumes the clusters are spherically shaped.
- **The silhouette score** of an instance $\mathbf{x}$ is $(b-a)/max(a,b)$ where b is the average distance of $\mathbf{x}$ to the points in its own cluster and b is the average distance to the points in the nearest cluster.
- Silhouette coefficient is between -1 and 1. If its near 1 then $\mathbf{x}$ is well inside its cluster. If its near zero then it is near the boundary of its clusterand if its near -1 then $\mathbf{x}$ may have been wrongly clustered.

Cluster evaluation metrics

To be able to evaluate the goodness of clusters $C_1, C_2, \ldots, C_k$ obtained using a clustering algorithm, we can use the following metrics.

- **Inertia:** the within-cluster sum of squares.The problem with Inertia is that it assumes the clusters are spherically shaped.
- **The silhouette score** of an instance **x** is $(b - a)/max(a, b)$ where b is the average distance of **x** to the points in its own cluster and b is the average distance to the points in the nearest cluster.
- Silhouette coefficient is between -1 and 1. If its near 1 then **x** is well inside its cluster. If its near zero then it is near the boundary of its clusterand if its near -1 then **x** may have been wrongly clustered.
- The *Silhouette score* of the whole clusters is the mean of the silhouette coefficient of the instances.

Choosing the value of k

Choosing the value of k

We choose k by plotting a cluster evaluation metric for different values of k.

Choosing the value of k

We choose k by plotting a cluster evaluation metric for different values of k.

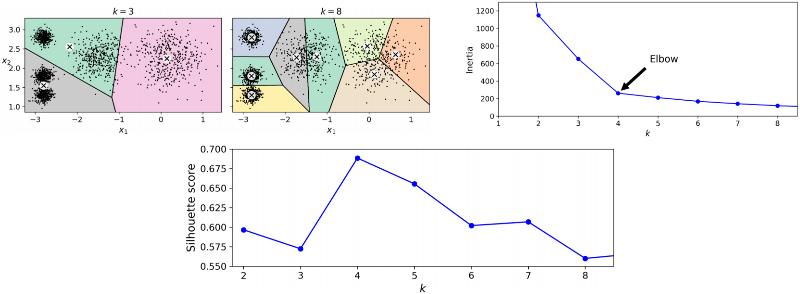


Figure: Using the elbow method (middle) and silhouette coefficient (bottom) to choose the value of k.

Applications of K-Means Clustering

- We can cluster the training dataset into some clusters and train classification on each.

Applications of K-Means Clustering

- We can cluster the training dataset into some clusters and train classification on each.
- Semi-supervised learning: We can find the instance closet to the centroid in each cluster.

Applications of K-Means Clustering

- We can cluster the training dataset into some clusters and train classification on each.
- Semi-supervised learning: We can find the instance closet to the centroid in each cluster. These are considered as *representatives* of their clusters.

Applications of K-Means Clustering

- We can cluster the training dataset into some clusters and train classification on each.
- Semi-supervised learning: We can find the instance closet to the centroid in each cluster. These are considered as *representatives* of their clusters. We label them manually.

Applications of K-Means Clustering

- We can cluster the training dataset into some clusters and train classification on each.
- Semi-supervised learning: We can find the instance closet to the centroid in each cluster. These are considered as *representatives* of their clusters. We label them manually.
- We can then generalize the label of the representatives to all the elements in the cluster

Applications of K-Means Clustering

- We can cluster the training dataset into some clusters and train classification on each.
- Semi-supervised learning: We can find the instance closet to the centroid in each cluster. These are considered as *representatives* of their clusters. We label them manually.
- We can then generalize the label of the representatives to all the elements in the cluster and train a classification model.

Applications of K-Means Clustering

- We can cluster the training dataset into some clusters and train classification on each.
- Semi-supervised learning: We can find the instance closet to the centroid in each cluster. These are considered as *representatives* of their clusters. We label them manually.
- We can then generalize the label of the representatives to all the elements in the cluster and train a classification model.
- We get better results if we generalize the labels only, say, the 20% of the elements in each cluster closest to the centroid.

Hierarchical Agglomerative Clustering

- In hierarchical clustering, we build a hierarchy of clusters.

Hierarchical Agglomerative Clustering

- In hierarchical clustering, we build a hierarchy of clusters.
- Hierarchical clustering can either be *agglomerative* or *divisive*.

Hierarchical Agglomerative Clustering

- In hierarchical clustering, we build a hierarchy of clusters.
- Hierarchical clustering can either be *agglomerative* or *divisive*.
- We have a parameter ϵ called the distance threshold.
- In **agglomerative clustering**, at first (corresponding to $\epsilon = 0$), each datapoint is a cluster.

Hierarchical Agglomerative Clustering

- In hierarchical clustering, we build a hierarchy of clusters.
- Hierarchical clustering can either be *agglomerative* or *divisive*.
- We have a parameter ϵ called the distance threshold.
- In **agglomerative clustering**, at first (corresponding to $\epsilon = 0$), each datapoint is a cluster.
- As the distance threshold grows, the clusters are merged until all the points belong to one cluster.

Hierarchical Agglomerative Clustering

- In hierarchical clustering, we build a hierarchy of clusters.
- Hierarchical clustering can either be *agglomerative* or *divisive*.
- We have a parameter ϵ called the distance threshold.
- In **agglomerative clustering**, at first (corresponding to $\epsilon = 0$), each datapoint is a cluster.
- As the distance threshold grows, the clusters are merged until all the points belong to one cluster.
- This way we obtain a *dendrogram* of clusters.

Hierarchical Agglomerative Clustering

- In hierarchical clustering, we build a hierarchy of clusters.
- Hierarchical clustering can either be *agglomerative* or *divisive*.
- We have a parameter ϵ called the distance threshold.
- In **agglomerative clustering**, at first (corresponding to $\epsilon = 0$), each datapoint is a cluster.
- As the distance threshold grows, the clusters are merged until all the points belong to one cluster.
- This way we obtain a *dendrogram* of clusters.

Hierarchical Agglomerative Clustering cont.

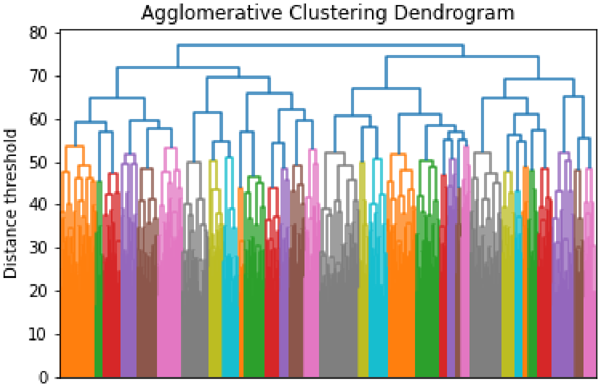


Figure: Dendrogram for the digits dataset, with complete linkage

Linkage types in Agglomerative Clustering

At distance threshold ϵ, two clusters C_1, C_2 are merged if:

- the distance of a point in C_1 is closer than ϵ to a point in C_2. (*Single linkage*)

Linkage types in Agglomerative Clustering

At distance threshold ϵ, two clusters C_1, C_2 are merged if:

- the distance of a point in C_1 is closer than ϵ to a point in C_2. (*Single linkage*)
- the average distance of points in C_1 is closer than ϵ to a point in C_2

Linkage types in Agglomerative Clustering

At distance threshold ϵ, two clusters C_1, C_2 are merged if:

- the distance of a point in C_1 is closer than ϵ to a point in C_2. (*Single linkage*)
- the average distance of points in C_1 is closer than ϵ to a point in C_2
- the distance between points of C_1 and C_2 is closer than ϵ. (*Average Linkage*)

Linkage types in Agglomerative Clustering

At distance threshold ϵ, two clusters C_1, C_2 are merged if:

- the distance of a point in C_1 is closer than ϵ to a point in C_2. (*Single linkage*)
- the average distance of points in C_1 is closer than ϵ to a point in C_2
- the distance between points of C_1 and C_2 is closer than ϵ. (*Average Linkage*)
- the maximum distance between points of C_1, C_2 is less than ϵ (*Complete Linkage*)

Linkage types in Agglomerative Clustering

At distance threshold ϵ, two clusters C_1, C_2 are merged if:

- the distance of a point in C_1 is closer than ϵ to a point in C_2. (*Single linkage*)
- the average distance of points in C_1 is closer than ϵ to a point in C_2
- the distance between points of C_1 and C_2 is closer than ϵ. (*Average Linkage*)
- the maximum distance between points of C_1, C_2 is less than ϵ (*Complete Linkage*)
- In *Ward linkage*, the distance between two clusters is defined as

$$\sum_{\mathbf{x} \in C_1 \cup C_2} ||\mathbf{x} - \mu_{C_1 \cup C_2}||^2 - \sum_{\mathbf{x} \in C_1} ||\mathbf{x} - \mu_{C_1}||^2 - \sum_{\mathbf{x} \in C_2} ||\mathbf{x} - \mu_{C_2}||^2 \quad (2)$$

Linkage types in Agglomerative Clustering

At distance threshold ϵ, two clusters C_1, C_2 are merged if:

- the distance of a point in C_1 is closer than ϵ to a point in C_2. (*Single linkage*)
- the average distance of points in C_1 is closer than ϵ to a point in C_2
- the distance between points of C_1 and C_2 is closer than ϵ. (*Average Linkage*)
- the maximum distance between points of C_1, C_2 is less than ϵ (*Complete Linkage*)
- In *Ward linkage*, the distance between two clusters is defined as

$$\sum_{\mathbf{x}\in C_1\cup C_2} ||\mathbf{x} - \mu_{C_1\cup C_2}||^2 - \sum_{\mathbf{x}\in C_1} ||\mathbf{x} - \mu_{C_1}||^2 - \sum_{\mathbf{x}\in C_2} ||\mathbf{x} - \mu_{C_2}||^2 \quad (2)$$

- The computational complexity of Agglomerative Clustering is

Linkage types in Agglomerative Clustering

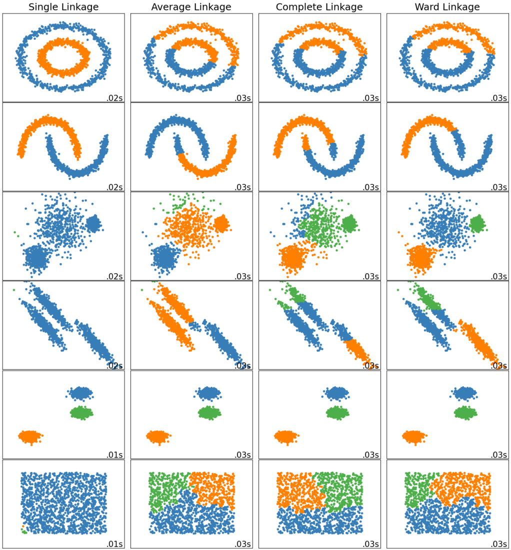
At distance threshold ϵ, two clusters C_1, C_2 are merged if:

- the distance of a point in C_1 is closer than ϵ to a point in C_2. (*Single linkage*)
- the average distance of points in C_1 is closer than ϵ to a point in C_2
- the distance between points of C_1 and C_2 is closer than ϵ. (*Average Linkage*)
- the maximum distance between points of C_1, C_2 is less than ϵ (*Complete Linkage*)
- In *Ward linkage*, the distance between two clusters is defined as

$$\sum_{\mathbf{x}\in C_1\cup C_2} ||\mathbf{x} - \mu_{C_1\cup C_2}||^2 - \sum_{\mathbf{x}\in C_1} ||\mathbf{x} - \mu_{C_1}||^2 - \sum_{\mathbf{x}\in C_2} ||\mathbf{x} - \mu_{C_2}||^2 \qquad (2)$$

- The computational complexity of Agglomerative Clustering is $\mathcal{O}(n^3)$.

Comparison of different linkage methods



DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a *density-based clustering* algorithm.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a *density-based clustering* algorithm.
- In density-based clustering, each cluster must have enough density.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a *density-based clustering* algorithm.
- In density-based clustering, each cluster must have enough density.
- DBSCAN has two parameters: a distance threshold ϵ and a minimum number of neighbors `min_samples`.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a *density-based clustering* algorithm.
- In density-based clustering, each cluster must have enough density.
- DBSCAN has two parameters: a distance threshold ϵ and a minimum number of neighbors `min_samples`.
- At the beginning, the neighbors of each element within distance ϵ are computed.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a *density-based clustering* algorithm.
- In density-based clustering, each cluster must have enough density.
- DBSCAN has two parameters: a distance threshold ϵ and a minimum number of neighbors `min_samples`.
- At the beginning, the neighbors of each element within distance ϵ are computed.
- Datapoints which have at least `min_samples` elements in their ϵ-neighborhood are called **core points**.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a *density-based clustering* algorithm.
- In density-based clustering, each cluster must have enough density.
- DBSCAN has two parameters: a distance threshold ϵ and a minimum number of neighbors `min_samples`.
- At the beginning, the neighbors of each element within distance ϵ are computed.
- Datapoints which have at least `min_samples` elements in their ϵ-neighborhood are called **core points**.
- A point in a cluster must be either a core point or in an ϵ-neighborhood of a core point.

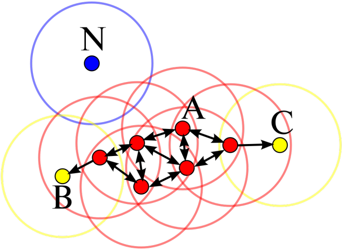
DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a *density-based clustering* algorithm.
- In density-based clustering, each cluster must have enough density.
- DBSCAN has two parameters: a distance threshold ϵ and a minimum number of neighbors `min_samples`.
- At the beginning, the neighbors of each element within distance ϵ are computed.
- Datapoints which have at least `min_samples` elements in their ϵ-neighborhood are called **core points**.
- A point in a cluster must be either a core point or in an ϵ-neighborhood of a core point.
- Datapoints which are neither core points nor in an ϵ-neighborhood of a core point, are considered as **noise**.

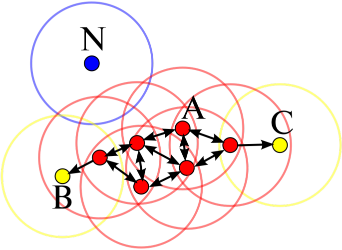
DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

- DBSCAN is a *density-based clustering* algorithm.
- In density-based clustering, each cluster must have enough density.
- DBSCAN has two parameters: a distance threshold ϵ and a minimum number of neighbors `min_samples`.
- At the beginning, the neighbors of each element within distance ϵ are computed.
- Datapoints which have at least `min_samples` elements in their ϵ-neighborhood are called **core points**.
- A point in a cluster must be either a core point or in an ϵ-neighborhood of a core point.
- Datapoints which are neither core points nor in an ϵ-neighborhood of a core point, are considered as **noise**.

DBSCAN cont.

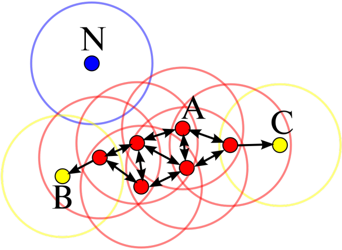


DBSCAN cont.



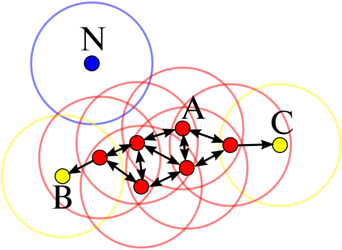
- Starting from a core point p, a point q is *reachable* from p if there is a sequence $p_0, p_1, \ldots, p_k$ such that $p_0 = p$, $p_k = q$, $p_1, p_2, \ldots, p_{k-1}$ are core points AND p_{i+1} is in the ϵ-neighborhood of p_i.

DBSCAN cont.



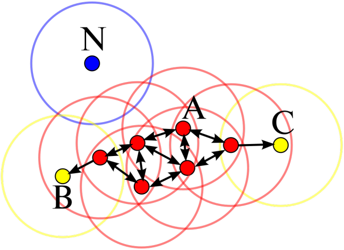
- Starting from a core point p, a point q is *reachable* from p if there is a sequence $p_0, p_1, \ldots, p_k$ such that $p_0 = p$, $p_k = q$, $p_1, p_2, \ldots, p_{k-1}$ are core points AND p_{i+1} is in the ϵ-neighborhood of p_i.
- All the points reachable from p are added to its cluster.

DBSCAN cont.



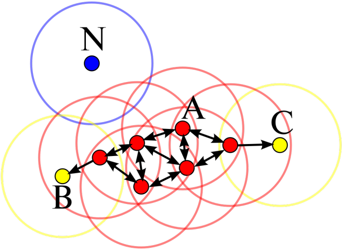
- Starting from a core point p, a point q is *reachable* from p if there is a sequence $p_0, p_1, \ldots, p_k$ such that $p_0 = p$, $p_k = q$, $p_1, p_2, \ldots, p_{k-1}$ are core points AND p_{i+1} is in the ϵ-neighborhood of p_i.
- All the points reachable from p are added to its cluster.
- Note: no points are reachable from non-core points

DBSCAN cont.



- Starting from a core point p, a point q is *reachable* from p if there is a sequence $p_0, p_1, \ldots, p_k$ such that $p_0 = p$, $p_k = q$, $p_1, p_2, \ldots, p_{k-1}$ are core points AND p_{i+1} is in the ϵ-neighborhood of p_i.
- All the points reachable from p are added to its cluster.
- Note: no points are reachable from non-core points
- As a result the border points (such as yellow ones in the above figure) can belong to either cluster, depending on the processing order of the data.

DBSCAN cont.



- Starting from a core point p, a point q is *reachable* from p if there is a sequence $p_0, p_1, \ldots, p_k$ such that $p_0 = p$, $p_k = q$, $p_1, p_2, \ldots, p_{k-1}$ are core points AND p_{i+1} is in the ϵ-neighborhood of p_i.
- All the points reachable from p are added to its cluster.
- Note: no points are reachable from non-core points
- As a result the border points (such as yellow ones in the above figure) can belong to either cluster, depending on the processing order of the data.

DBSCAN cont.

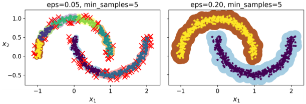


Figure: Credit: Aurelien Geron

- DBSCAN can find arbitrary shaped clusters, even if one is surrounded by another.

DBSCAN cont.

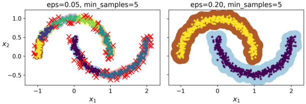


Figure: Credit: Aurelien Geron

- DBSCAN can find arbitrary shaped clusters, even if one is surrounded by another.
- It however, does not perform well if the clusters have varying densities.

DBSCAN cont.

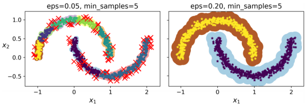


Figure: Credit: Aurelien Geron

- DBSCAN can find arbitrary shaped clusters, even if one is surrounded by another.
- It however, does not perform well if the clusters have varying densities.
- In Scikit-Learn, DBSCAN is provided by the class `sklearn.cluster.DBSCAN`.

DBSCAN cont.

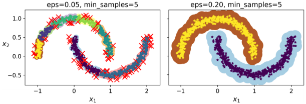


Figure: Credit: Aurelien Geron

- DBSCAN can find arbitrary shaped clusters, even if one is surrounded by another.
- It however, does not perform well if the clusters have varying densities.
- In Scikit-Learn, DBSCAN is provided by the class `sklearn.cluster.DBSCAN`. It does not have a `predict` method but we can use KNN to predict the cluster of new instances.

DBSCAN cont.

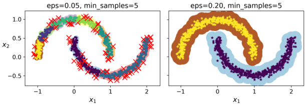


Figure: Credit: Aurelien Geron

- DBSCAN can find arbitrary shaped clusters, even if one is surrounded by another.
- It however, does not perform well if the clusters have varying densities.
- In Scikit-Learn, DBSCAN is provided by the class `sklearn.cluster.DBSCAN`. It does not have a `predict` method but we can use KNN to predict the cluster of new instances.
- The worst case computational complexity of DBSCAN is $\mathcal{O}(n^2)$.

DBSCAN cont.

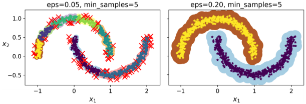


Figure: Credit: Aurelien Geron

- DBSCAN can find arbitrary shaped clusters, even if one is surrounded by another.
- It however, does not perform well if the clusters have varying densities.
- In Scikit-Learn, DBSCAN is provided by the class `sklearn.cluster.DBSCAN`. It does not have a `predict` method but we can use KNN to predict the cluster of new instances.
- The worst case computational complexity of DBSCAN is $\mathcal{O}(n^2)$.
- If `min_samples` equals 2, DBSCAN is equivalent to single-linkage hierarchical clustering with distance threshold ϵ.

DBSCAN cont.

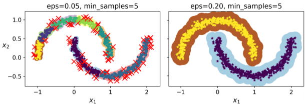


Figure: Credit: Aurelien Geron

- DBSCAN can find arbitrary shaped clusters, even if one is surrounded by another.
- It however, does not perform well if the clusters have varying densities.
- In Scikit-Learn, DBSCAN is provided by the class `sklearn.cluster.DBSCAN`. It does not have a `predict` method but we can use KNN to predict the cluster of new instances.
- The worst case computational complexity of DBSCAN is $\mathcal{O}(n^2)$.
- If `min_samples` equals 2, DBSCAN is equivalent to single-linkage hierarchical clustering with distance threshold ϵ.
- The value of `min_samples` should be larger for higher dimensional data.