COMP4121 Advanced Algorithms

Aleks Ignjatović

School of Computer Science and Engineering
University of New South Wales

Clustering algorithms

What is clustering?

- Fundamentally important for data science
- Making sense of data on its own
- Data preprocessing for other algorithms
- It is a type of *unsupervised learning.*

What is clustering?

- Fundamentally important for data science
- Making sense of data on its own
- Data preprocessing for other algorithms
- It is a type of *unsupervised learning.*

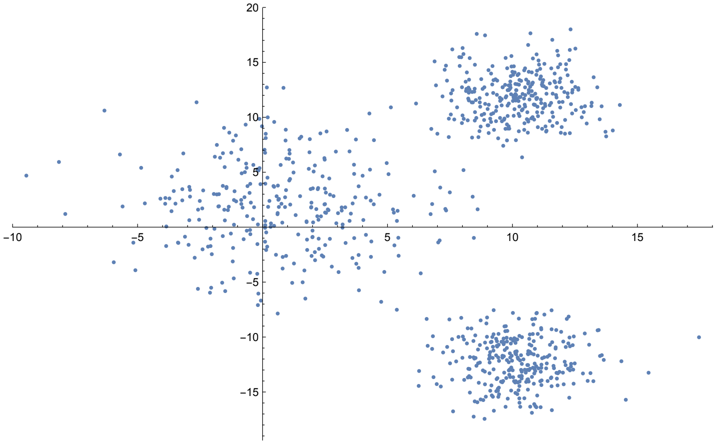
What is clustering?

- Fundamentally important for data science
- Making sense of data on its own
- Data preprocessing for other algorithms
- It is a type of *unsupervised learning.*

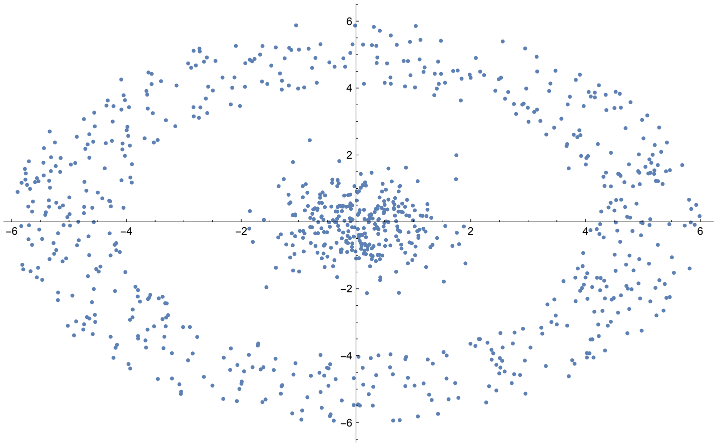
What is clustering?

- Fundamentally important for data science
- Making sense of data on its own
- Data preprocessing for other algorithms
- It is a type of *unsupervised learning*.

How many clusters are there?

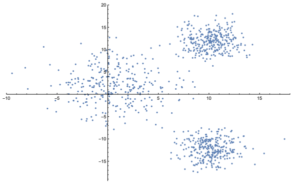


How many clusters are there?

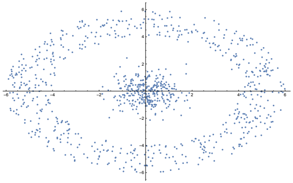


What are clusters?

- Two kinds of clusters:
  1. center - based clusters



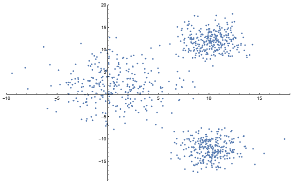
  2. high density clusters



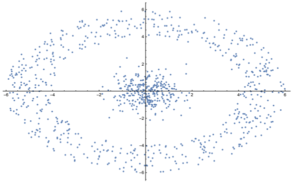
- A good clustering algorithm should be able to handle both kinds.

What are clusters?

- Two kinds of clusters:
  1. center - based clusters



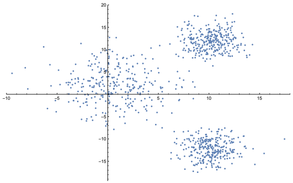
  2. high density clusters



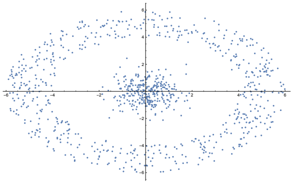
- A good clustering algorithm should be able to handle both kinds.

What are clusters?

- Two kinds of clusters:
  1. center - based clusters



  2. high density clusters



- A good clustering algorithm should be able to handle both kinds.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- We have to make sure that the data is adequately represented.
- In general, there are two most common representations:
  1. as vectors in $\mathbb{R}^d$
     - This is suitable when you have several numerical measurements of each object, such as the red blood cell count, white blood cell count, haemoglobin content, etc for each patient in a group of patients.
     - Another example might be the relative frequencies of the key words in each document from a collection.
     - Note that d can be extremely large, corresponding to thousands or more of possible keywords.
     - This can be a problem due to complexities of storing and handling such high dimensional data. (this is where Johnson - Lindenstrauss Theorem come to play)
  2. as a weighted graph
     - Data points are represented as vertices of the graph;
     - The weights of the edges reflect the degree of similarity (or dissimilarity) of the data points.
     - Data represented as vectors in $\mathbb{R}^d$ can be represented as a weighted graph where the weights of edges reflect dissimilarity of the end points as measured by some form of distance between the end points.

Data representation

- The distance between two data points $x, y \in \mathbb{R}^d$ can be defined as either

$$d(x,y) = \sum_{i=1}^{d} |x_i - y_i| \quad \text{or as} \quad d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Such a distance $d(x,y)$ is then taken as a measure of dissimilarity of x and y.
- If the scales of the i^{th} and j^{th} coordinates x_i and x_j differ significantly, or if they are not of equal importance, we might consider instead

$$d(x,y)^2 = \sum_{i=1}^{d} w_i (x_i - y_i)^2$$

- The weights w_i are chosen to normalise different variances of x_i and x_j or to encode their relative significances.
- Graph representation of data is often much a more compact representation than as vectors in $\mathbb{R}^d$, which does not suffer from problems of high dimensionality.
- Note that in the graph representation of data the geometry of the data points is lost, so the clustering is based only on mutual distances of pairs of points, which is good when clustering is not center based.

Data representation

- The distance between two data points $x, y \in \mathbb{R}^d$ can be defined as either

$$d(x,y) = \sum_{i=1}^{d} |x_i - y_i| \quad \text{or as} \quad d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Such a distance $d(x, y)$ is then taken as a measure of dissimilarity of x and y.
- If the scales of the i^{th} and j^{th} coordinates x_i and x_j differ significantly, or if they are not of equal importance, we might consider instead

$$d(x,y)^2 = \sum_{i=1}^{d} w_i (x_i - y_i)^2$$

- The weights w_i are chosen to normalise different variances of x_i and x_j or to encode their relative significances.
- Graph representation of data is often much a more compact representation than as vectors in $\mathbb{R}^d$, which does not suffer from problems of high dimensionality.
- Note that in the graph representation of data the geometry of the data points is lost, so the clustering is based only on mutual distances of pairs of points, which is good when clustering is not center based.

Data representation

- The distance between two data points $x, y \in \mathbb{R}^d$ can be defined as either

$$d(x,y) = \sum_{i=1}^{d} |x_i - y_i| \quad \text{or as} \quad d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Such a distance $d(x,y)$ is then taken as a measure of dissimilarity of x and y.
- If the scales of the i^{th} and j^{th} coordinates x_i and x_j differ significantly, or if they are not of equal importance, we might consider instead

$$d(x,y)^2 = \sum_{i=1}^{d} w_i (x_i - y_i)^2$$

- The weights w_i are chosen to normalise different variances of x_i and x_j or to encode their relative significances.
- Graph representation of data is often much a more compact representation than as vectors in $\mathbb{R}^d$, which does not suffer from problems of high dimensionality.
- Note that in the graph representation of data the geometry of the data points is lost, so the clustering is based only on mutual distances of pairs of points, which is good when clustering is not center based.

Data representation

- The distance between two data points $x, y \in \mathbb{R}^d$ can be defined as either

$$d(x,y) = \sum_{i=1}^{d} |x_i - y_i| \quad \text{or as} \quad d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Such a distance $d(x, y)$ is then taken as a measure of dissimilarity of x and y.
- If the scales of the i^{th} and j^{th} coordinates x_i and x_j differ significantly, or if they are not of equal importance, we might consider instead

$$d(x,y)^2 = \sum_{i=1}^{d} w_i (x_i - y_i)^2$$

- The weights w_i are chosen to normalise different variances of x_i and x_j or to encode their relative significances.
- Graph representation of data is often much a more compact representation than as vectors in $\mathbb{R}^d$, which does not suffer from problems of high dimensionality.
- Note that in the graph representation of data the geometry of the data points is lost, so the clustering is based only on mutual distances of pairs of points, which is good when clustering is not center based.

Data representation

- The distance between two data points $x, y \in \mathbb{R}^d$ can be defined as either

$$d(x,y) = \sum_{i=1}^{d} |x_i - y_i| \quad \text{or as} \quad d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Such a distance $d(x, y)$ is then taken as a measure of dissimilarity of x and y.
- If the scales of the i^{th} and j^{th} coordinates x_i and x_j differ significantly, or if they are not of equal importance, we might consider instead

$$d(x,y)^2 = \sum_{i=1}^{d} w_i (x_i - y_i)^2$$

- The weights w_i are chosen to normalise different variances of x_i and x_j or to encode their relative significances.
- Graph representation of data is often much a more compact representation than as vectors in $\mathbb{R}^d$, which does not suffer from problems of high dimensionality.
- Note that in the graph representation of data the geometry of the data points is lost, so the clustering is based only on mutual distances of pairs of points, which is good when clustering is not center based.

Data representation

- The distance between two data points $x, y \in \mathbb{R}^d$ can be defined as either

$$d(x,y) = \sum_{i=1}^{d} |x_i - y_i| \quad \text{or as} \quad d(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Such a distance $d(x,y)$ is then taken as a measure of dissimilarity of x and y.
- If the scales of the i^{th} and j^{th} coordinates x_i and x_j differ significantly, or if they are not of equal importance, we might consider instead

$$d(x,y)^2 = \sum_{i=1}^{d} w_i (x_i - y_i)^2$$

- The weights w_i are chosen to normalise different variances of x_i and x_j or to encode their relative significances.
- Graph representation of data is often much a more compact representation than as vectors in $\mathbb{R}^d$, which does not suffer from problems of high dimensionality.
- Note that in the graph representation of data the geometry of the data points is lost, so the clustering is based only on mutual distances of pairs of points, which is good when clustering is not center based.

Center-based clustering algorithms

We assume data points are represented as vectors in $\mathbb{R}^d$.

1. **k-center clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimize the maximum distance between any data point and the center of its cluster.
   - That is, we want to minimize

$$\Phi(\mathcal{C}) = \max_{j=1}^{k} \max_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$

   - This is the "fire-station location problem" since one can think of it as the problem of building k fire-stations in a city so as to minimise the maximum distance a fire-truck needs to travel to put out a fire.
   - The *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.
   - Thus, center-based clustering algorithms try to minimise the radius of the resulting clustering.

Center-based clustering algorithms

We assume data points are represented as vectors in $\mathbb{R}^d$.

1. **k-center clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimize the maximum distance between any data point and the center of its cluster.
   - That is, we want to minimize

$$\Phi(\mathcal{C}) = \max_{j=1}^{k} \max_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$

   - This is the "fire-station location problem" since one can think of it as the problem of building k fire-stations in a city so as to minimise the maximum distance a fire-truck needs to travel to put out a fire.
   - The *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.
   - Thus, center-based clustering algorithms try to minimise the radius of the resulting clustering.

Center-based clustering algorithms

We assume data points are represented as vectors in $\mathbb{R}^d$.

1. **k-center clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimize the maximum distance between any data point and the center of its cluster.
   - That is, we want to minimize

$$\Phi(\mathcal{C}) = \max_{j=1}^{k} \max_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$

   - This is the "fire-station location problem" since one can think of it as the problem of building k fire-stations in a city so as to minimise the maximum distance a fire-truck needs to travel to put out a fire.
   - The *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.
   - Thus, center-based clustering algorithms try to minimise the radius of the resulting clustering.

Center-based clustering algorithms

We assume data points are represented as vectors in $\mathbb{R}^d$.

1. **k-center clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimize the maximum distance between any data point and the center of its cluster.
   - That is, we want to minimize

$$\Phi(\mathcal{C}) = \max_{j=1}^{k} \max_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$

   - This is the "fire-station location problem" since one can think of it as the problem of building k fire-stations in a city so as to minimise the maximum distance a fire-truck needs to travel to put out a fire.
   - The *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.
   - Thus, center-based clustering algorithms try to minimise the radius of the resulting clustering.

Center-based clustering algorithms

We assume data points are represented as vectors in $\mathbb{R}^d$.

1. **k-center clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \ldots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$, which minimize the maximum distance between any data point and the center of its cluster.
   - That is, we want to minimize

$$\Phi(\mathcal{C}) = \max_{j=1}^{k} \max_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$

   - This is the "fire-station location problem" since one can think of it as the problem of building k fire-stations in a city so as to minimise the maximum distance a fire-truck needs to travel to put out a fire.
   - The *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.
   - Thus, center-based clustering algorithms try to minimise the radius of the resulting clustering.

Center-based clustering algorithms

We assume data points are represented as vectors in $\mathbb{R}^d$.

1. **k-center clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimize the maximum distance between any data point and the center of its cluster.
   - That is, we want to minimize
   $$\Phi(\mathcal{C}) = \max_{j=1}^{k} \max_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$
   - This is the "fire-station location problem" since one can think of it as the problem of building k fire-stations in a city so as to minimise the maximum distance a fire-truck needs to travel to put out a fire.
   - The *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.
   - Thus, center-based clustering algorithms try to minimise the radius of the resulting clustering.

Center-based clustering

1. **k-median clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \ldots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$, which minimizes the sum of distances between data points and their corresponding cluster centres.
   - That is, we want to minimise

$$\Phi(\mathcal{C}) = \sum_{j=1}^{k} \sum_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$

   - Note that $d(\mathbf{a}, \mathbf{c}_j)$ can be any distance metric, such as ℓ^1 metric $d_1(\mathbf{a}, \mathbf{c}_j) = \sum_{k=1}^{d} |(\mathbf{a})_k - (\mathbf{c}_j)_k|$ or ℓ^2 metric $d_2(\mathbf{a}, \mathbf{c}_j) = \sqrt{\sum_{k=1}^{d} ((\mathbf{a})_k - (\mathbf{c}_j)_k)^2}$.
   - If the distance is the ℓ_1 distance, one can show that the coordinates of the optimal centers are the coordinate-wise medians of points in each cluster.

Center-based clustering

1. **k-median clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimizes the sum of distances between data points and their corresponding cluster centres.
   - That is, we want to minimise

$$\Phi(\mathcal{C}) = \sum_{j=1}^{k} \sum_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$

   - Note that $d(\mathbf{a}, \mathbf{c}_j)$ can be any distance metric, such as ℓ^1 metric $d_1(\mathbf{a}, \mathbf{c}_j) = \sum_{k=1}^{d} |(\mathbf{a})_k - (\mathbf{c}_j)_k|$ or ℓ^2 metric $d_2(\mathbf{a}, \mathbf{c}_j) = \sqrt{\sum_{k=1}^{d} ((\mathbf{a})_k - (\mathbf{c}_j)_k)^2}$.
   - If the distance is the ℓ_1 distance, one can show that the coordinates of the optimal centers are the coordinate-wise medians of points in each cluster.

Center-based clustering

1. **k-median clustering:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimizes the sum of distances between data points and their corresponding cluster centres.
   - That is, we want to minimise

$$\Phi(\mathcal{C}) = \sum_{j=1}^{k} \sum_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)$$

   - Note that $d(\mathbf{a}, \mathbf{c}_j)$ can be any distance metric, such as ℓ^1 metric $d_1(\mathbf{a}, \mathbf{c}_j) = \sum_{k=1}^{d} |(\mathbf{a})_k - (\mathbf{c}_j)_k|$ or ℓ^2 metric $d_2(\mathbf{a}, \mathbf{c}_j) = \sqrt{\sum_{k=1}^{d} ((\mathbf{a})_k - (\mathbf{c}_j)_k)^2}$.
   - If the distance is the ℓ_1 distance, one can show that the coordinates of the optimal centers are the coordinate-wise medians of points in each cluster.

Center-based clustering algorithms

The most frequently used center-based clustering algorithm is the k-means algorithm.

1. k**-means clustering problem:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \ldots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$, which minimize the sum of the squares of distances between data points and their corresponding cluster centers.
   - That is, we want to minimize

$$\Phi(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)^2$$

   - k-means clustering penalises more for larger distances than the k-median clustering.
   - k-means clustering has other nice properties; for example, if $d(\mathbf{a}, \mathbf{c}_j)^2 = \sum_{j=1}^{d} (a_i - c_{ji})^2$ then $\mathbf{c}_j$ must be the centroids of the points in their cluster.

Center-based clustering algorithms

The most frequently used center-based clustering algorithm is the k-means algorithm.

1. **k-means clustering problem:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimize the sum of the squares of distances between data points and their corresponding cluster centers.
   - That is, we want to minimize

$$\Phi(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)^2$$

   - k-means clustering penalises more for larger distances than the k-median clustering.
   - k-means clustering has other nice properties; for example, if $d(\mathbf{a}, \mathbf{c}_j)^2 = \sum_{j=1}^{d} (a_i - c_{ji})^2$ then $\mathbf{c}_j$ must be the centroids of the points in their cluster.

Center-based clustering algorithms

The most frequently used center-based clustering algorithm is the k-means algorithm.

1. **k-means clustering problem:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \ldots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$, which minimize the sum of the squares of distances between data points and their corresponding cluster centers.
   - That is, we want to minimize

$$\Phi(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)^2$$

   - k-means clustering penalises more for larger distances than the k-median clustering.
   - k-means clustering has other nice properties; for example, if $d(\mathbf{a}, \mathbf{c}_j)^2 = \sum_{j=1}^{d} (a_i - c_{ji})^2$ then $\mathbf{c}_j$ must be the centroids of the points in their cluster.

Center-based clustering algorithms

The most frequently used center-based clustering algorithm is the k-means algorithm.

1. ***k*-means clustering problem:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \dots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \dots, \mathbf{c}_k$, which minimize the sum of the squares of distances between data points and their corresponding cluster centers.
   - That is, we want to minimize
   $$\Phi(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)^2$$
   - k-means clustering penalises more for larger distances than the k-median clustering.
   - k-means clustering has other nice properties; for example, if $d(\mathbf{a}, \mathbf{c}_j)^2 = \sum_{j=1}^{d} (a_i - c_{ji})^2$ then $\mathbf{c}_j$ must be the centroids of the points in their cluster.

Center-based clustering algorithms

The most frequently used center-based clustering algorithm is the k-means algorithm.

1. **k-means clustering problem:** Find a partition $\mathcal{C} = \{\mathcal{C}_1, \ldots \mathcal{C}_k\}$ of a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ into k clusters, with the corresponding centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$, which minimize the sum of the squares of distances between data points and their corresponding cluster centers.
    - That is, we want to minimize

$$\Phi(\mathcal{C}) = \sum_{i=1}^{k} \sum_{a \in \mathcal{C}_j} d(\mathbf{a}, \mathbf{c}_j)^2$$

    - k-means clustering penalises more for larger distances than the k-median clustering.
    - k-means clustering has other nice properties; for example, if $d(\mathbf{a}, \mathbf{c}_j)^2 = \sum_{j=1}^{d} (a_i - c_{ji})^2$ then $\mathbf{c}_j$ must be the centroids of the points in their cluster.

Center-based clustering algorithms

- What is the centroid of a set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$?
- Keep in mind that each $\mathbf{a}_i$ is a vector in $\mathbb{R}^d$, so

$$\mathbf{a}_i = (a_{i1}, \ldots, a_{id}).$$

- Let $\mathbf{c} = (c_1, \ldots, c_d)$ with for all $1 \leq k \leq d$,

$$c_k = (a_{1k} + \ldots + a_{nk})/n$$

  i.e., c_k is the arithmetic mean of the k^{th} coordinates of all of the points $\mathbf{a}_1, \ldots, \mathbf{a}_n$.
- Then $\mathbf{c}$ is called *the centroid* of the set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.

Center-based clustering algorithms

- What is the centroid of a set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$?
- Keep in mind that each $\mathbf{a}_i$ is a vector in $\mathbb{R}^d$, so

$$\mathbf{a}_i = (a_{i1}, \ldots, a_{id}).$$

- Let $\mathbf{c} = (c_1, \ldots, c_d)$ with for all $1 \leq k \leq d$,

$$c_k = (a_{1k} + \ldots + a_{nk})/n$$

  i.e., c_k is the arithmetic mean of the k^{th} coordinates of all of the points $\mathbf{a}_1, \ldots, \mathbf{a}_n$.
- Then $\mathbf{c}$ is called *the centroid* of the set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.

Center-based clustering algorithms

- What is the centroid of a set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$?
- Keep in mind that each $\mathbf{a}_i$ is a vector in $\mathbb{R}^d$, so

$$\mathbf{a}_i = (a_{i1}, \ldots, a_{id}).$$

- Let $\mathbf{c} = (c_1, \ldots, c_d)$ with for all $1 \leq k \leq d$,

$$c_k = (a_{1k} + \ldots + a_{nk})/n$$

  i.e., c_k is the arithmetic mean of the k^{th} coordinates of all of the points $\mathbf{a}_1, \ldots, \mathbf{a}_n$.
- Then $\mathbf{c}$ is called *the centroid* of the set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.

Center-based clustering algorithms

- What is the centroid of a set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$?
- Keep in mind that each $\mathbf{a}_i$ is a vector in $\mathbb{R}^d$, so

$$\mathbf{a}_i = (a_{i1}, \ldots, a_{id}).$$

- Let $\mathbf{c} = (c_1, \ldots, c_d)$ with for all $1 \leq k \leq d$,

$$c_k = (a_{1k} + \ldots + a_{nk})/n$$

  i.e., c_k is the arithmetic mean of the k^{th} coordinates of all of the points $\mathbf{a}_1, \ldots, \mathbf{a}_n$.
- Then $\mathbf{c}$ is called *the centroid* of the set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.

Center-based clustering algorithms

- We denote by $\mathbf{x} \cdot \mathbf{y}$ the scalar product of vectors $\mathbf{x}$ and $\mathbf{y}$ and by $\|\mathbf{x}\|$ the norm of a vector $\mathbf{x}$, i.e.

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{d} x_i y_i \quad \text{and} \quad \|\mathbf{x}\| = \sqrt{\sum_{i=1}^{d} x_i^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$$

- Note that $\|\mathbf{x} - \mathbf{y}\|$ is the Euclidean distance of points $\mathbf{x}$ and $\mathbf{y}$:

$$\|\mathbf{x} - \mathbf{y}\| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Note also that

$$\|\mathbf{x} + \mathbf{y}\|^2 = (\mathbf{x} + \mathbf{y}) \cdot (\mathbf{x} + \mathbf{y}) = \|\mathbf{x}\|^2 + 2\,\mathbf{x} \cdot \mathbf{y} + \|\mathbf{y}\|^2$$

Center-based clustering algorithms

- We denote by $\mathbf{x} \cdot \mathbf{y}$ the scalar product of vectors $\mathbf{x}$ and $\mathbf{y}$ and by $\|\mathbf{x}\|$ the norm of a vector $\mathbf{x}$, i.e.

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{d} x_i y_i \quad \text{and} \quad \|\mathbf{x}\| = \sqrt{\sum_{i=1}^{d} x_i^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}}$$

- Note that $\|\mathbf{x} - \mathbf{y}\|$ is the Euclidean distance of points $\mathbf{x}$ and $\mathbf{y}$:

$$\|\mathbf{x} - \mathbf{y}\| = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Note also that

$$\|\mathbf{x} + \mathbf{y}\|^2 = (\mathbf{x} + \mathbf{y}) \cdot (\mathbf{x} + \mathbf{y}) = \|\mathbf{x}\|^2 + 2\,\mathbf{x} \cdot \mathbf{y} + \|\mathbf{y}\|^2$$

Center-based clustering algorithms

- We denote by $\mathbf{x}\cdot\mathbf{y}$ the scalar product of vectors $\mathbf{x}$ and $\mathbf{y}$ and by $\|\mathbf{x}\|$ the norm of a vector $\mathbf{x}$, i.e.

$$\mathbf{x}\cdot\mathbf{y}=\sum_{i=1}^{d}x_iy_i \quad \text{and} \quad \|\mathbf{x}\|=\sqrt{\sum_{i=1}^{d}x_i^2}=\sqrt{\mathbf{x}\cdot\mathbf{x}}$$

- Note that $\|\mathbf{x}-\mathbf{y}\|$ is the Euclidean distance of points $\mathbf{x}$ and $\mathbf{y}$:

$$\|\mathbf{x}-\mathbf{y}\|=\sqrt{\sum_{i=1}^{d}(x_i-y_i)^2}$$

- Note also that

$$\|\mathbf{x}+\mathbf{y}\|^2=(\mathbf{x}+\mathbf{y})\cdot(\mathbf{x}+\mathbf{y})=\|\mathbf{x}\|^2+2\,\mathbf{x}\cdot\mathbf{y}+\|\mathbf{y}\|^2$$

Center-based clustering algorithms

- **Theorem:** Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points and $\mathbf{x}$ another point, all in $\mathbb{R}^d$. Let also $\mathbf{c}$ be the centroid of A. Then

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + n\|\mathbf{c} - \mathbf{x}\|^2$$

- **Proof:** Using $\|\mathbf{x}+\mathbf{y}\|^2 = (\mathbf{x}+\mathbf{y}) \cdot (\mathbf{x}+\mathbf{y}) = \|\mathbf{x}\|^2 + 2\,\mathbf{x} \cdot \mathbf{y} + \|\mathbf{y}\|^2$

$$\begin{aligned}\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 &= \sum_{i=1}^{n} \|(\mathbf{a}_i - \mathbf{c}) + (\mathbf{c} - \mathbf{x})\|^2 \\ &= \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + 2(\mathbf{c} - \mathbf{x}) \cdot \sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{c}) + n\|\mathbf{c} - \mathbf{x}\|^2\end{aligned}$$

Since $\mathbf{c}$ is the centroid, $\mathbf{c} = \left(\sum_{i=1}^{n} \mathbf{a}_i\right)/n$, we have

$$\sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{c}) = \sum_{i=1}^{n} \mathbf{a}_i - n\mathbf{c} = 0$$

and we have proved the claim.

Center-based clustering algorithms

- **Theorem:** Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points and $\mathbf{x}$ another point, all in $\mathbb{R}^d$. Let also $\mathbf{c}$ be the centroid of A. Then

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + n\|\mathbf{c} - \mathbf{x}\|^2$$

- **Proof:** Using $\|\mathbf{x}+\mathbf{y}\|^2 = (\mathbf{x}+\mathbf{y}) \cdot (\mathbf{x}+\mathbf{y}) = \|\mathbf{x}\|^2 + 2\,\mathbf{x}\cdot\mathbf{y} + \|\mathbf{y}\|^2$

$$\begin{aligned}\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 &= \sum_{i=1}^{n} \|(\mathbf{a}_i - \mathbf{c}) + (\mathbf{c} - \mathbf{x})\|^2 \\ &= \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + 2(\mathbf{c}-\mathbf{x}) \cdot \sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{c}) + n\|\mathbf{c} - \mathbf{x}\|^2\end{aligned}$$

Since $\mathbf{c}$ is the centroid, $\mathbf{c} = \left(\sum_{i=1}^{n} \mathbf{a}_i\right)/n$, we have

$$\sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{c}) = \sum_{i=1}^{n} \mathbf{a}_i - n\mathbf{c} = 0$$

and we have proved the claim.

Center-based clustering algorithms

- **Theorem:** Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points and $\mathbf{x}$ another point, all in $\mathbb{R}^d$. Let also $\mathbf{c}$ be the centroid of A. Then

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + n\|\mathbf{c} - \mathbf{x}\|^2$$

- **Proof:** Using $\|\mathbf{x} + \mathbf{y}\|^2 = (\mathbf{x} + \mathbf{y}) \cdot (\mathbf{x} + \mathbf{y}) = \|\mathbf{x}\|^2 + 2\,\mathbf{x} \cdot \mathbf{y} + \|\mathbf{y}\|^2$

$$\begin{aligned}\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 &= \sum_{i=1}^{n} \|(\mathbf{a}_i - \mathbf{c}) + (\mathbf{c} - \mathbf{x})\|^2 \\ &= \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + 2(\mathbf{c} - \mathbf{x}) \cdot \sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{c}) + n\|\mathbf{c} - \mathbf{x}\|^2\end{aligned}$$

Since $\mathbf{c}$ is the centroid, $\mathbf{c} = \left(\sum_{i=1}^{n} \mathbf{a}_i\right)/n$, we have

$$\sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{c}) = \sum_{i=1}^{n} \mathbf{a}_i - n\mathbf{c} = 0$$

and we have proved the claim.

Center-based clustering algorithms

- **Theorem:** Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points and $\mathbf{x}$ another point, all in $\mathbb{R}^d$. Let also $\mathbf{c}$ be the centroid of A. Then

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + n\|\mathbf{c} - \mathbf{x}\|^2$$

- **Proof:** Using $\|\mathbf{x}+\mathbf{y}\|^2 = (\mathbf{x}+\mathbf{y}) \cdot (\mathbf{x}+\mathbf{y}) = \|\mathbf{x}\|^2 + 2\,\mathbf{x}\cdot\mathbf{y} + \|\mathbf{y}\|^2$

$$\begin{aligned}\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 &= \sum_{i=1}^{n} \|(\mathbf{a}_i - \mathbf{c}) + (\mathbf{c} - \mathbf{x})\|^2 \\ &= \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + 2(\mathbf{c} - \mathbf{x}) \cdot \sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{c}) + n\|\mathbf{c} - \mathbf{x}\|^2\end{aligned}$$

Since $\mathbf{c}$ is the centroid, $\mathbf{c} = \left(\sum_{i=1}^{n} \mathbf{a}_i\right)/n$, we have

$$\sum_{i=1}^{n} (\mathbf{a}_i - \mathbf{c}) = \sum_{i=1}^{n} \mathbf{a}_i - n\mathbf{c} = 0$$

and we have proved the claim.

Center-based clustering algorithms

- **Theorem:** Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points and $\mathbf{x}$ another point, all in $\mathbb{R}^d$. Let also $\mathbf{c}$ be the centroid of A. Then

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + n\|\mathbf{c} - \mathbf{x}\|^2$$

- **Proof:** Using $\|\mathbf{x} + \mathbf{y}\|^2 = (\mathbf{x} + \mathbf{y}) \cdot (\mathbf{x} + \mathbf{y}) = \|\mathbf{x}\|^2 + 2\,\mathbf{x} \cdot \mathbf{y} + \|\mathbf{y}\|^2$

$$\begin{aligned}\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 &= \sum_{i=1}^{n} \|(\mathbf{a}_i - \mathbf{c}) + (\mathbf{c} - \mathbf{x})\|^2 \\ &= \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + 2(\mathbf{c} - \mathbf{x}) \cdot \sum_{i=1}^{n}(\mathbf{a}_i - \mathbf{c}) + n\|\mathbf{c} - \mathbf{x}\|^2\end{aligned}$$

Since $\mathbf{c}$ is the centroid, $\mathbf{c} = \left(\sum_{i=1}^{n} \mathbf{a}_i\right)/n$, we have

$$\sum_{i=1}^{n}(\mathbf{a}_i - \mathbf{c}) = \sum_{i=1}^{n} \mathbf{a}_i - n\mathbf{c} = 0$$

and we have proved the claim.

Center-based clustering algorithms

- **Corollary:** Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points in $\mathbb{R}^d$. Then

$$D(\mathbf{x}) = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2$$

  is minimised when $\mathbf{x}$ is the centroid $\mathbf{c} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$.
- **Proof:** By the previous theorem,

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + n\|\mathbf{c} - \mathbf{x}\|^2$$

  The first summand does not depend on $\mathbf{x}$ and the second is zero when $\mathbf{x} = \mathbf{c}$. Thus $\mathbf{x} = \mathbf{c}$ minimises $D(\mathbf{x})$.

Center-based clustering algorithms

- **Corollary:** Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points in $\mathbb{R}^d$. Then

$$D(\mathbf{x}) = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2$$

  is minimised when $\mathbf{x}$ is the centroid $\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$.
- **Proof:** By the previous theorem,

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + n\|\mathbf{c} - \mathbf{x}\|^2$$

  The first summand does not depend on $\mathbf{x}$ and the second is zero when $\mathbf{x} = \mathbf{c}$. Thus $\mathbf{x} = \mathbf{c}$ minimises $D(\mathbf{x})$.

Center-based clustering algorithms

- **Corollary:** Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points in $\mathbb{R}^d$. Then

$$D(\mathbf{x}) = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2$$

  is minimised when $\mathbf{x}$ is the centroid $\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$.
- **Proof:** By the previous theorem,

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{x}\|^2 = \sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{c}\|^2 + n\|\mathbf{c} - \mathbf{x}\|^2$$

  The first summand does not depend on $\mathbf{x}$ and the second is zero when $\mathbf{x} = \mathbf{c}$. Thus $\mathbf{x} = \mathbf{c}$ minimises $D(\mathbf{x})$.

Center-based clustering algorithms

- Thus, if we are given a set of points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ in $\mathbb{R}^d$ and the problem is to find a partition of A into k disjoint components $A = \bigcup_{i=1}^{k} A_i$ and k points $\mathbf{x}_1, \ldots, \mathbf{x}_k$ such that the sum

$$\sum_{j=1}^{k} \sum_{\mathbf{a}_i \in A_j} \|\mathbf{a}_i - \mathbf{x}_j\|^2$$

  is as small as possible, then, whatever such an optimal partition $\{A_j : 1 \leq j \leq k\}$ might be, the points $\mathbf{x}_j$ must be the centroids $\mathbf{c}_j$ of sets A_j.
- Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points in $\mathbb{R}^d$ and let $\mathbf{c}$ be the centroid of A. Then

$$\frac{1}{2n} \sum_{i,j=1}^{n} \|\mathbf{a}_i - \mathbf{a}_j\|^2 = \sum_{m=1}^{n} \|\mathbf{a}_m - \mathbf{c}\|^2$$

- This is easy to see just by replacing $\mathbf{c}$ by its definition plus doing some obvious algebra.

Center-based clustering algorithms

- Thus, if we are given a set of points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ in $\mathbb{R}^d$ and the problem is to find a partition of A into k disjoint components $A = \bigcup_{i=1}^{k} A_i$ and k points $\mathbf{x}_1, \ldots, \mathbf{x}_k$ such that the sum

$$\sum_{j=1}^{k} \sum_{\mathbf{a}_i \in A_j} \|\mathbf{a}_i - \mathbf{x}_j\|^2$$

  is as small as possible, then, whatever such an optimal partition $\{A_j : 1 \leq j \leq k\}$ might be, the points $\mathbf{x}_j$ must be the centroids $\mathbf{c}_j$ of sets A_j.
- Let $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ be a set of points in $\mathbb{R}^d$ and let $\mathbf{c}$ be the centroid of A. Then

$$\frac{1}{2n} \sum_{i,j=1}^{n} \|\mathbf{a}_i - \mathbf{a}_j\|^2 = \sum_{m=1}^{n} \|\mathbf{a}_m - \mathbf{c}\|^2$$

- This is easy to see just by replacing $\mathbf{c}$ by its definition plus doing some obvious algebra.

Center-based clustering algorithms

- Thus, if we are given a set of points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ in $\mathbb{R}^d$ and the problem is to find a partition of A into k disjoint components $A = \bigcup_{i=1}^k A_i$ and k points $\mathbf{x}_1, \dots, \mathbf{x}_k$ such that the sum

$$\sum_{j=1}^{k} \sum_{\mathbf{a}_i \in A_j} \|\mathbf{a}_i - \mathbf{x}_j\|^2$$

  is as small as possible, then, whatever such an optimal partition $\{A_j : 1 \leq j \leq k\}$ might be, the points $\mathbf{x}_j$ must be the centroids $\mathbf{c}_j$ of sets A_j.
- Let $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ be a set of points in $\mathbb{R}^d$ and let $\mathbf{c}$ be the centroid of A. Then

$$\frac{1}{2n} \sum_{i,j=1}^{n} \|\mathbf{a}_i - \mathbf{a}_j\|^2 = \sum_{m=1}^{n} \|\mathbf{a}_m - \mathbf{c}\|^2$$

- This is easy to see just by replacing $\mathbf{c}$ by its definition plus doing some obvious algebra.

Center-based clustering algorithms

- Thus, finding disjoint components $A = \bigcup_{m=1}^{k} A_m$ which minimises

$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m} \|\mathbf{a}_j - \mathbf{c}_m\|^2$$

is equivalent to minimising

$$\sum_{m=1}^{k} \frac{1}{2|A_m|} \sum_{\mathbf{a}_i, \mathbf{a}_j \in A_m} \|\mathbf{a}_i - \mathbf{a}_j\|^2$$

Lloyd's Algorithm

- Finding the optimal k-means clustering is an NP hard problem which cannot be solved in polynomial time, so we have to look at approximate solutions.
- The best known approximate k-means clustering algorithm is Lloyd's algorithm.
- **Lloyd's Clustering Algorithm**:
  1. Start with an initial set of cluster centres $\{\mathbf{c}_m^{(0)} : 1 \leq m \leq k\}$ (we will later explain how to obtain such an initial set).
  2. Cluster all points $\mathbf{a} \in A$ into clusters A_m by associating each $\mathbf{a} \in A$ with the nearest cluster centre.
  3. Replace cluster centres with the centroids of thus obtained clusters.
  4. Repeat 2 and 3 until cluster centres (and thus also clusters) stop changing.

Lloyd's Algorithm

- Finding the optimal k-means clustering is an NP hard problem which cannot be solved in polynomial time, so we have to look at approximate solutions.
- The best known approximate k-means clustering algorithm is Lloyd's algorithm.
- **Lloyd's Clustering Algorithm**:
  1. Start with an initial set of cluster centres $\{\mathbf{c}_m^{(0)} : 1 \leq m \leq k\}$ (we will later explain how to obtain such an initial set).
  2. Cluster all points $\mathbf{a} \in A$ into clusters A_m by associating each $\mathbf{a} \in A$ with the nearest cluster centre.
  3. Replace cluster centres with the centroids of thus obtained clusters.
  4. Repeat 2 and 3 until cluster centres (and thus also clusters) stop changing.

Lloyd's Algorithm

- Finding the optimal k-means clustering is an NP hard problem which cannot be solved in polynomial time, so we have to look at approximate solutions.
- The best known approximate k-means clustering algorithm is Lloyd's algorithm.
- **Lloyd's Clustering Algorithm**:
  1. Start with an initial set of cluster centres $\{\mathbf{c}_m^{(0)} : 1 \leq m \leq k\}$ (we will later explain how to obtain such an initial set).
  2. Cluster all points $\mathbf{a} \in A$ into clusters A_m by associating each $\mathbf{a} \in A$ with the nearest cluster centre.
  3. Replace cluster centres with the centroids of thus obtained clusters.
  4. Repeat 2 and 3 until cluster centres (and thus also clusters) stop changing.

Lloyd's Algorithm

- Finding the optimal k-means clustering is an NP hard problem which cannot be solved in polynomial time, so we have to look at approximate solutions.
- The best known approximate k-means clustering algorithm is Lloyd's algorithm.
- **Lloyd's Clustering Algorithm**:
  1. Start with an initial set of cluster centres $\{\mathbf{c}_m^{(0)} : 1 \leq m \leq k\}$ (we will later explain how to obtain such an initial set).
  2. Cluster all points $\mathbf{a} \in A$ into clusters A_m by associating each $\mathbf{a} \in A$ with the nearest cluster centre.
  3. Replace cluster centres with the centroids of thus obtained clusters.
  4. Repeat 2 and 3 until cluster centres (and thus also clusters) stop changing.

Lloyd's Algorithm

- Finding the optimal k-means clustering is an NP hard problem which cannot be solved in polynomial time, so we have to look at approximate solutions.
- The best known approximate k-means clustering algorithm is Lloyd's algorithm.
- **Lloyd's Clustering Algorithm**:
  1. Start with an initial set of cluster centres $\{\mathbf{c}_m^{(0)} : 1 \leq m \leq k\}$ (we will later explain how to obtain such an initial set).
  2. Cluster all points $\mathbf{a} \in A$ into clusters A_m by associating each $\mathbf{a} \in A$ with the nearest cluster centre.
  3. Replace cluster centres with the centroids of thus obtained clusters.
  4. Repeat 2 and 3 until cluster centres (and thus also clusters) stop changing.

Lloyd's Algorithm

- Finding the optimal k-means clustering is an NP hard problem which cannot be solved in polynomial time, so we have to look at approximate solutions.
- The best known approximate k-means clustering algorithm is Lloyd's algorithm.
- **Lloyd's Clustering Algorithm**:
  1. Start with an initial set of cluster centres $\{\mathbf{c}_m^{(0)} : 1 \leq m \leq k\}$ (we will later explain how to obtain such an initial set).
  2. Cluster all points $\mathbf{a} \in A$ into clusters A_m by associating each $\mathbf{a} \in A$ with the nearest cluster centre.
  3. Replace cluster centres with the centroids of thus obtained clusters.
  4. Repeat 2 and 3 until cluster centres (and thus also clusters) stop changing.

Lloyd's Algorithm

- **Claim:** At every round p of its loop, Lloyd's algorithm reduces the size of
$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$
where $A_m^{(p)}$ are the "temporary" clusters and $\mathbf{c}_m^{(p)}$ is the "temporary" centre of cluster $A_m^{(p)}$ at round p of the loop.
- This is obvious, because both steps of the loop have this property: replacing the cluster centres with the centroid of the cluster recuses every summand $\sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$ and so does associating every point with the nearest cluster centre.
- However, the algorithm might stop at a local minimum, not the globally optimal minimum.

Lloyd's Algorithm

- **Claim:** At every round p of its loop, Lloyd's algorithm reduces the size of

$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$

  where $A_m^{(p)}$ are the "temporary" clusters and $\mathbf{c}_m^{(p)}$ is the "temporary" centre of cluster $A_m^{(p)}$ at round p of the loop.
- This is obvious, because both steps of the loop have this property: replacing the cluster centres with the centroid of the cluster recuses every summand $\sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$ and so does associating every point with the nearest cluster centre.
- However, the algorithm might stop at a local minimum, not the globally optimal minimum.

Lloyd's Algorithm

- **Claim:** At every round p of its loop, Lloyd's algorithm reduces the size of
$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$
where $A_m^{(p)}$ are the "temporary" clusters and $\mathbf{c}_m^{(p)}$ is the "temporary" centre of cluster $A_m^{(p)}$ at round p of the loop.
- This is obvious, because both steps of the loop have this property: replacing the cluster centres with the centroid of the cluster recuses every summand $\sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$ and so does associating every point with the nearest cluster centre.
- However, the algorithm might stop at a local minimum, not the globally optimal minimum.

Lloyd's Algorithm

- In lots of applications this local minimum provides a good clustering.
- However, sometimes better results are obtained by running the algorithm several times with different initial set of cluster centres and picking the solution from the run for which the sum

$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$

  is the smallest.
- How can we obtain good starting centres of clusters for the algorithm?
- One good option is to pick a random point $\mathbf{a}_q$ from A as the first centre $\mathbf{c}_1^{(0)} = \mathbf{a}_q$.
- For the second centre pick another point from A which is the farthest away from $\mathbf{a}_q$.
- Continue in this manner to get all k cluster centres, always picking as the next centre a point from A which has the largest minimal distance to all previously picked centres.

Lloyd's Algorithm

- In lots of applications this local minimum provides a good clustering.
- However, sometimes better results are obtained by running the algorithm several times with different initial set of cluster centres and picking the solution from the run for which the sum

$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$

  is the smallest.
- How can we obtain good starting centres of clusters for the algorithm?
- One good option is to pick a random point $\mathbf{a}_q$ from A as the first centre $\mathbf{c}_1^{(0)} = \mathbf{a}_q$.
- For the second centre pick another point from A which is the farthest away from $\mathbf{a}_q$.
- Continue in this manner to get all k cluster centres, always picking as the next centre a point from A which has the largest minimal distance to all previously picked centres.

Lloyd's Algorithm

- In lots of applications this local minimum provides a good clustering.
- However, sometimes better results are obtained by running the algorithm several times with different initial set of cluster centres and picking the solution from the run for which the sum

$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$

  is the smallest.
- How can we obtain good starting centres of clusters for the algorithm?
- One good option is to pick a random point $\mathbf{a}_q$ from A as the first centre $\mathbf{c}_1^{(0)} = \mathbf{a}_q$.
- For the second centre pick another point from A which is the farthest away from $\mathbf{a}_q$.
- Continue in this manner to get all k cluster centres, always picking as the next centre a point from A which has the largest minimal distance to all previously picked centres.

Lloyd's Algorithm

- In lots of applications this local minimum provides a good clustering.
- However, sometimes better results are obtained by running the algorithm several times with different initial set of cluster centres and picking the solution from the run for which the sum

$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$

  is the smallest.
- How can we obtain good starting centres of clusters for the algorithm?
- One good option is to pick a random point $\mathbf{a}_q$ from A as the first centre $\mathbf{c}_1^{(0)} = \mathbf{a}_q$.
- For the second centre pick another point from A which is the farthest away from $\mathbf{a}_q$.
- Continue in this manner to get all k cluster centres, always picking as the next centre a point from A which has the largest minimal distance to all previously picked centres.

Lloyd's Algorithm

- In lots of applications this local minimum provides a good clustering.
- However, sometimes better results are obtained by running the algorithm several times with different initial set of cluster centres and picking the solution from the run for which the sum

$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$

  is the smallest.
- How can we obtain good starting centres of clusters for the algorithm?
- One good option is to pick a random point $\mathbf{a}_q$ from A as the first centre $\mathbf{c}_1^{(0)} = \mathbf{a}_q$.
- For the second centre pick another point from A which is the farthest away from $\mathbf{a}_q$.
- Continue in this manner to get all k cluster centres, always picking as the next centre a point from A which has the largest minimal distance to all previously picked centres.

Lloyd's Algorithm

- In lots of applications this local minimum provides a good clustering.
- However, sometimes better results are obtained by running the algorithm several times with different initial set of cluster centres and picking the solution from the run for which the sum

$$\sum_{m=1}^{k} \sum_{\mathbf{a}_j \in A_m^{(p)}} \|\mathbf{a}_j - \mathbf{c}_m^{(p)}\|^2$$

  is the smallest.
- How can we obtain good starting centres of clusters for the algorithm?
- One good option is to pick a random point $\mathbf{a}_q$ from A as the first centre $\mathbf{c}_1^{(0)} = \mathbf{a}_q$.
- For the second centre pick another point from A which is the farthest away from $\mathbf{a}_q$.
- Continue in this manner to get all k cluster centres, always picking as the next centre a point from A which has the largest minimal distance to all previously picked centres.

Lloyd's Algorithm

- In fact, such a procedure for finding the initial centres $\mathbf{c}_m^{(0)}$ of initial clusters is used as a simple clustering algorithm in itself. You first choose cluster centers as described and then simply clustering points from A according to which is the closest centre $\mathbf{c}_m^{(0)}$ to that point.
- This algorithm is usually called The Farthest Traversal k-clustering algorithm.
- The Farthest Traversal k-clustering algorithm, despite its simplicity, provides a reasonably good approximate clustering in the following sense.
- (Remember that the *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.)

Lloyd's Algorithm

- In fact, such a procedure for finding the initial centres $\mathbf{c}_m^{(0)}$ of initial clusters is used as a simple clustering algorithm in itself. You first choose cluster centers as described and then simply clustering points from A according to which is the closest centre $\mathbf{c}_m^{(0)}$ to that point.
- This algorithm is usually called The Farthest Traversal k-clustering algorithm.
- The Farthest Traversal k-clustering algorithm, despite its simplicity, provides a reasonably good approximate clustering in the following sense.
- (Remember that the *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.)

Lloyd's Algorithm

- In fact, such a procedure for finding the initial centres $\mathbf{c}_m^{(0)}$ of initial clusters is used as a simple clustering algorithm in itself. You first choose cluster centers as described and then simply clustering points from A according to which is the closest centre $\mathbf{c}_m^{(0)}$ to that point.
- This algorithm is usually called The Farthest Traversal k-clustering algorithm.
- The Farthest Traversal k-clustering algorithm, despite its simplicity, provides a reasonably good approximate clustering in the following sense.
- (Remember that the *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.)

Lloyd's Algorithm

- In fact, such a procedure for finding the initial centres $\mathbf{c}_m^{(0)}$ of initial clusters is used as a simple clustering algorithm in itself. You first choose cluster centers as described and then simply clustering points from A according to which is the closest centre $\mathbf{c}_m^{(0)}$ to that point.
- This algorithm is usually called The Farthest Traversal k-clustering algorithm.
- The Farthest Traversal k-clustering algorithm, despite its simplicity, provides a reasonably good approximate clustering in the following sense.
- (Remember that the *radius* of a clustering $A = \bigcup_{m=1}^{k} A_m$ is the largest distance of a point from A to its associated cluster centre.)

Lloyd's Algorithm

- **Theorem:** If A has a k-clustering of radius r, then the Farthest Traversal k-clustering algorithm produces a clustering of radius at most $2r$.
- **Proof:** Suppose opposite, that there is $\mathbf{a} \in A$ at a distance to its cluster larger that $2r$. This would mean that the distance of $\mathbf{a}$ to all cluster centres is larger than $2r$.
- But this implies that also the distances between all pairs of cluster centres must also be larger than $2r$ because otherwise $\mathbf{a}$ would have been chosen as one of the cluster centres.
- Thus, we would have at least $k+1$ points ($\mathbf{a}$ plus the k cluster centres) which are all on pairwise distances larger than $2r$.
- Since we have $k+1$ points in k clusters, two such points must be in the same cluster.
- But no such two points can be in the same cluster of radius r, because their distance would be at most the diameter of the circle which is $2r$.
- Thus there cannot be a k clustering of radius r, which is a contradiction.

Lloyd's Algorithm

- **Theorem:** If A has a k-clustering of radius r, then the Farthest Traversal k-clustering algorithm produces a clustering of radius at most $2r$.
- **Proof:** Suppose opposite, that there is $\mathbf{a} \in A$ at a distance to its cluster larger that $2r$. This would mean that the distance of $\mathbf{a}$ to all cluster centres is larger than $2r$.
- But this implies that also the distances between all pairs of cluster centres must also be larger than $2r$ because otherwise $\mathbf{a}$ would have been chosen as one of the cluster centres.
- Thus, we would have at least $k + 1$ points ($\mathbf{a}$ plus the k cluster centres) which are all on pairwise distances larger than $2r$.
- Since we have $k + 1$ points in k clusters, two such points must be in the same cluster.
- But no such two points can be in the same cluster of radius r, because their distance would be at most the diameter of the circle which is $2r$.
- Thus there cannot be a k clustering of radius r, which is a contradiction.

Lloyd's Algorithm

- **Theorem:** If A has a k-clustering of radius r, then the Farthest Traversal k-clustering algorithm produces a clustering of radius at most $2r$.
- **Proof:** Suppose opposite, that there is $\mathbf{a} \in A$ at a distance to its cluster larger that $2r$. This would mean that the distance of $\mathbf{a}$ to all cluster centres is larger than $2r$.
- But this implies that also the distances between all pairs of cluster centres must also be larger than $2r$ because otherwise $\mathbf{a}$ would have been chosen as one of the cluster centres.
- Thus, we would have at least $k+1$ points ($\mathbf{a}$ plus the k cluster centres) which are all on pairwise distances larger than $2r$.
- Since we have $k+1$ points in k clusters, two such points must be in the same cluster.
- But no such two points can be in the same cluster of radius r, because their distance would be at most the diameter of the circle which is $2r$.
- Thus there cannot be a k clustering of radius r, which is a contradiction.

Lloyd's Algorithm

- **Theorem:** If A has a k-clustering of radius r, then the Farthest Traversal k-clustering algorithm produces a clustering of radius at most $2r$.
- **Proof:** Suppose opposite, that there is $\mathbf{a} \in A$ at a distance to its cluster larger that $2r$. This would mean that the distance of $\mathbf{a}$ to all cluster centres is larger than $2r$.
- But this implies that also the distances between all pairs of cluster centres must also be larger than $2r$ because otherwise $\mathbf{a}$ would have been chosen as one of the cluster centres.
- Thus, we would have at least $k + 1$ points ($\mathbf{a}$ plus the k cluster centres) which are all on pairwise distances larger than $2r$.
- Since we have $k + 1$ points in k clusters, two such points must be in the same cluster.
- But no such two points can be in the same cluster of radius r, because their distance would be at most the diameter of the circle which is $2r$.
- Thus there cannot be a k clustering of radius r, which is a contradiction.

Lloyd's Algorithm

- **Theorem:** If A has a k-clustering of radius r, then the Farthest Traversal k-clustering algorithm produces a clustering of radius at most $2r$.
- **Proof:** Suppose opposite, that there is $\mathbf{a} \in A$ at a distance to its cluster larger that $2r$. This would mean that the distance of $\mathbf{a}$ to all cluster centres is larger than $2r$.
- But this implies that also the distances between all pairs of cluster centres must also be larger than $2r$ because otherwise $\mathbf{a}$ would have been chosen as one of the cluster centres.
- Thus, we would have at least $k+1$ points ($\mathbf{a}$ plus the k cluster centres) which are all on pairwise distances larger than $2r$.
- Since we have $k+1$ points in k clusters, two such points must be in the same cluster.
- But no such two points can be in the same cluster of radius r, because their distance would be at most the diameter of the circle which is $2r$.
- Thus there cannot be a k clustering of radius r, which is a contradiction.

Lloyd's Algorithm

- **Theorem:** If A has a k-clustering of radius r, then the Farthest Traversal k-clustering algorithm produces a clustering of radius at most $2r$.
- **Proof:** Suppose opposite, that there is $\mathbf{a} \in A$ at a distance to its cluster larger that $2r$. This would mean that the distance of $\mathbf{a}$ to all cluster centres is larger than $2r$.
- But this implies that also the distances between all pairs of cluster centres must also be larger than $2r$ because otherwise $\mathbf{a}$ would have been chosen as one of the cluster centres.
- Thus, we would have at least $k+1$ points ($\mathbf{a}$ plus the k cluster centres) which are all on pairwise distances larger than $2r$.
- Since we have $k+1$ points in k clusters, two such points must be in the same cluster.
- But no such two points can be in the same cluster of radius r, because their distance would be at most the diameter of the circle which is $2r$.
- Thus there cannot be a k clustering of radius r, which is a contradiction.

Lloyd's Algorithm

- **Theorem:** If A has a k-clustering of radius r, then the Farthest Traversal k-clustering algorithm produces a clustering of radius at most $2r$.
- **Proof:** Suppose opposite, that there is $\mathbf{a} \in A$ at a distance to its cluster larger that $2r$. This would mean that the distance of $\mathbf{a}$ to all cluster centres is larger than $2r$.
- But this implies that also the distances between all pairs of cluster centres must also be larger than $2r$ because otherwise $\mathbf{a}$ would have been chosen as one of the cluster centres.
- Thus, we would have at least $k+1$ points ($\mathbf{a}$ plus the k cluster centres) which are all on pairwise distances larger than $2r$.
- Since we have $k+1$ points in k clusters, two such points must be in the same cluster.
- But no such two points can be in the same cluster of radius r, because their distance would be at most the diameter of the circle which is $2r$.
- Thus there cannot be a k clustering of radius r, which is a contradiction.

Ward's Algorithm

- Loyd's algorithm has been recently randomized by instead of picking always the furthest point, by picking a point with probability proportional to the shortest distance to one of already picked points.
- Let A_m be a cluster with its centroid $\mathbf{c}_m$ as its centre; let us set

$$\mathrm{cost}(A_m) = \sum_{\mathbf{a}_i \in A_m} \|\mathbf{a}_i - \mathbf{c}_m\|^2$$

- The k-means clustering algorithms are trying to minimise the sum $\sum_{m=1}^{k} \mathrm{cost}(A_m)$.
- Note that if we have two clusters A_m and A_l and take their union $B = A_m \cup A_l$ and the centroid of B as the cluster centre of B, then

$$\mathrm{cost}(B) \geq \mathrm{cost}(A_m) + \mathrm{cost}(A_l),$$

because two centres can get closer to the points in $B = A_m \cup A_l$ then just one point.

Ward's Algorithm

- Loyd's algorithm has been recently randomized by instead of picking always the furthest point, by picking a point with probability proportional to the shortest distance to one of already picked points.
- Let A_m be a cluster with its centroid $\mathbf{c}_m$ as its centre; let us set

$$\mathrm{cost}(A_m) = \sum_{\mathbf{a}_i \in A_m} \|\mathbf{a}_i - \mathbf{c}_m\|^2$$

- The k-means clustering algorithms are trying to minimise the sum $\sum_{m=1}^{k} \mathrm{cost}(A_m)$.
- Note that if we have two clusters A_m and A_l and take their union $B = A_m \cup A_l$ and the centroid of B as the cluster centre of B, then

$$\mathrm{cost}(B) \geq \mathrm{cost}(A_m) + \mathrm{cost}(A_l),$$

because two centres can get closer to the points in $B = A_m \cup A_l$ then just one point.

Ward's Algorithm

- Loyd's algorithm has been recently randomized by instead of picking always the furthest point, by picking a point with probability proportional to the shortest distance to one of already picked points.
- Let A_m be a cluster with its centroid $\mathbf{c}_m$ as its centre; let us set

$$\mathrm{cost}(A_m) = \sum_{\mathbf{a}_i \in A_m} \|\mathbf{a}_i - \mathbf{c}_m\|^2$$

- The k-means clustering algorithms are trying to minimise the sum $\sum_{m=1}^{k} \mathrm{cost}(A_m)$.
- Note that if we have two clusters A_m and A_l and take their union $B = A_m \cup A_l$ and the centroid of B as the cluster centre of B, then

$$\mathrm{cost}(B) \geq \mathrm{cost}(A_m) + \mathrm{cost}(A_l),$$

  because two centres can get closer to the points in $B = A_m \cup A_l$ then just one point.

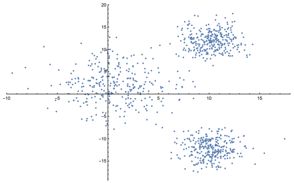
Ward's Algorithm

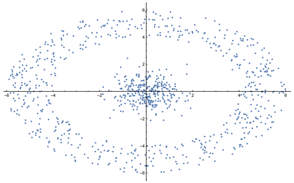
- Ward's algorithm is a greedy k-means algorithm:
  1. Start with every point $\mathbf{a}_i$ in its own cluster.
  2. While the number of clusters is larger than k repeat: find two clusters C and C' such that

$$\text{cost}(C \cup C') - \text{cost}(C) - \text{cost}(C')$$

     is as small as possible and replace them with a single merged cluster $C \cup C'$ with its centroid as its centre.

- How do we cluster data when clusters are not centre based??

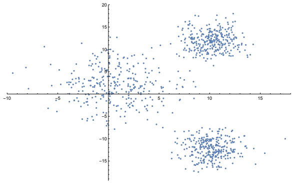
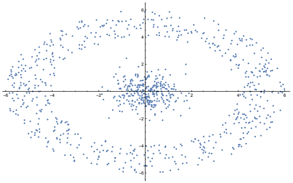




- This done using the *spectral clustering.*

Ward's Algorithm

- Ward's algorithm is a greedy k-means algorithm:
  1. Start with every point $\mathbf{a}_i$ in its own cluster.
  2. While the number of clusters is larger than k repeat: find two clusters C and C' such that
  $$\text{cost}(C \cup C') - \text{cost}(C) - \text{cost}(C')$$
  is as small as possible and replace them with a single merged cluster $C \cup C'$ with its centroid as its centre.
- How do we cluster data when clusters are not centre based??

- This done using the *spectral clustering*.

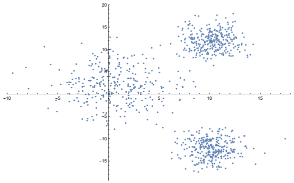
Ward's Algorithm

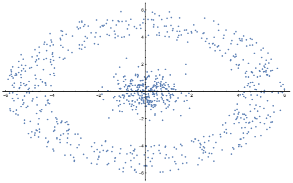
- Ward's algorithm is a greedy k-means algorithm:
  1. Start with every point $\mathbf{a}_i$ in its own cluster.
  2. While the number of clusters is larger than k repeat: find two clusters C and C' such that

$$\mathrm{cost}(C \cup C') - \mathrm{cost}(C) - \mathrm{cost}(C')$$

is as small as possible and replace them with a single merged cluster $C \cup C'$ with its centroid as its centre.

- How do we cluster data when clusters are not centre based??





- This done using the *spectral clustering*.

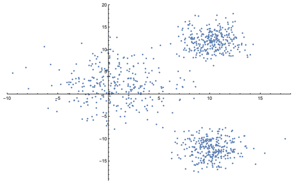
Ward's Algorithm

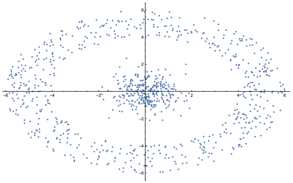
- Ward's algorithm is a greedy k-means algorithm:
  1. Start with every point $\mathbf{a}_i$ in its own cluster.
  2. While the number of clusters is larger than k repeat: find two clusters C and C' such that

$$\text{cost}(C \cup C') - \text{cost}(C) - \text{cost}(C')$$

  is as small as possible and replace them with a single merged cluster $C \cup C'$ with its centroid as its centre.

- How do we cluster data when clusters are not centre based??



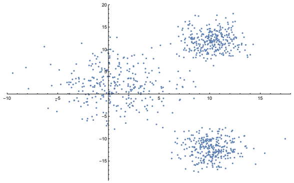


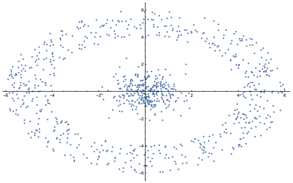
- This done using the *spectral clustering.*

Ward's Algorithm

- Ward's algorithm is a greedy k-means algorithm:
  1. Start with every point $\mathbf{a}_i$ in its own cluster.
  2. While the number of clusters is larger than k repeat: find two clusters C and C' such that
  $$\text{cost}(C \cup C') - \text{cost}(C) - \text{cost}(C')$$
  is as small as possible and replace them with a single merged cluster $C \cup C'$ with its centroid as its centre.

- How do we cluster data when clusters are not centre based??





- This done using the *spectral clustering*.

Similarity Graphs

- We represent a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ as the set of vertices $\{v_1, \dots, v_n\}$ of an undirected weighted graph $G = (V, E)$.
- The weight $w_{ij} \geq 0$ of an edge $e = (v_i, v_j)$ is equal to some form of similarity measure of the data points $\mathbf{a}_i, \mathbf{a}_j$ which correspond to vertices v_i, v_j.
- If $w_{ij} = 0$ this means that vertices v_i and v_j correspond to completely dissimilar data points $\mathbf{a}_i, \mathbf{a}_j$ and in this case the graph does not include an edge of the form $e = (v_i, v_j)$.
- The similarity of vertices v_i and v_j can depend, for example, on a decreasing function of the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\|$ between the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$, such as $e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2}}$.
- Since the graph is undirected, $w_{ij} = w_{ji}$.
- We put all the weights w_{ij} into a symmetric *adjacency matrix* $W = (w_{ij})_{i,j=1}^n$.
- Recall that $w_{ij} = 0$ means that there is no edge between vertices v_i and v_j.
- Since the graph is weighted, the *degree* d_i of a vertex v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Similarity Graphs

- We represent a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ as the set of vertices $\{v_1, \ldots, v_n\}$ of an undirected weighted graph $G = (V, E)$.
- The weight $w_{ij} \geq 0$ of an edge $e = (v_i, v_j)$ is equal to some form of similarity measure of the data points $\mathbf{a}_i, \mathbf{a}_j$ which correspond to vertices v_i, v_j.
- If $w_{ij} = 0$ this means that vertices v_i and v_j correspond to completely dissimilar data points $\mathbf{a}_i, \mathbf{a}_j$ and in this case the graph does not include an edge of the form $e = (v_i, v_j)$.
- The similarity of vertices v_i and v_j can depend, for example, on a decreasing function of the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\|$ between the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$, such as $e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2}}$.
- Since the graph is undirected, $w_{ij} = w_{ji}$.
- We put all the weights w_{ij} into a symmetric *adjacency matrix* $W = (w_{ij})_{i,j=1}^n$.
- Recall that $w_{ij} = 0$ means that there is no edge between vertices v_i and v_j.
- Since the graph is weighted, the *degree* d_i of a vertex v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Similarity Graphs

- We represent a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ as the set of vertices $\{v_1, \dots, v_n\}$ of an undirected weighted graph $G = (V, E)$.
- The weight $w_{ij} \geq 0$ of an edge $e = (v_i, v_j)$ is equal to some form of similarity measure of the data points $\mathbf{a}_i, \mathbf{a}_j$ which correspond to vertices v_i, v_j.
- If $w_{ij} = 0$ this means that vertices v_i and v_j correspond to completely dissimilar data points $\mathbf{a}_i, \mathbf{a}_j$ and in this case the graph does not include an edge of the form $e = (v_i, v_j)$.
- The similarity of vertices v_i and v_j can depend, for example, on a decreasing function of the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\|$ between the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$, such as $e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2}}$.
- Since the graph is undirected, $w_{ij} = w_{ji}$.
- We put all the weights w_{ij} into a symmetric *adjacency matrix* $W = (w_{ij})_{i,j=1}^{n}$.
- Recall that $w_{ij} = 0$ means that there is no edge between vertices v_i and v_j.
- Since the graph is weighted, the *degree* d_i of a vertex v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Similarity Graphs

- We represent a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ as the set of vertices $\{v_1, \ldots, v_n\}$ of an undirected weighted graph $G = (V, E)$.
- The weight $w_{ij} \geq 0$ of an edge $e = (v_i, v_j)$ is equal to some form of similarity measure of the data points $\mathbf{a}_i, \mathbf{a}_j$ which correspond to vertices v_i, v_j.
- If $w_{ij} = 0$ this means that vertices v_i and v_j correspond to completely dissimilar data points $\mathbf{a}_i, \mathbf{a}_j$ and in this case the graph does not include an edge of the form $e = (v_i, v_j)$.
- The similarity of vertices v_i and v_j can depend, for example, on a decreasing function of the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\|$ between the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$, such as $e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2}}$.
- Since the graph is undirected, $w_{ij} = w_{ji}$.
- We put all the weights w_{ij} into a symmetric *adjacency matrix* $W = (w_{ij})_{i,j=1}^n$.
- Recall that $w_{ij} = 0$ means that there is no edge between vertices v_i and v_j.
- Since the graph is weighted, the *degree* d_i of a vertex v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Similarity Graphs

- We represent a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ as the set of vertices $\{v_1, \ldots, v_n\}$ of an undirected weighted graph $G = (V, E)$.
- The weight $w_{ij} \geq 0$ of an edge $e = (v_i, v_j)$ is equal to some form of similarity measure of the data points $\mathbf{a}_i, \mathbf{a}_j$ which correspond to vertices v_i, v_j.
- If $w_{ij} = 0$ this means that vertices v_i and v_j correspond to completely dissimilar data points $\mathbf{a}_i, \mathbf{a}_j$ and in this case the graph does not include an edge of the form $e = (v_i, v_j)$.
- The similarity of vertices v_i and v_j can depend, for example, on a decreasing function of the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\|$ between the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$, such as $e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2}}$.
- Since the graph is undirected, $w_{ij} = w_{ji}$.
- We put all the weights w_{ij} into a symmetric *adjacency matrix* $W = (w_{ij})_{i,j=1}^n$.
- Recall that $w_{ij} = 0$ means that there is no edge between vertices v_i and v_j.
- Since the graph is weighted, the *degree* d_i of a vertex v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Similarity Graphs

- We represent a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ as the set of vertices $\{v_1, \ldots, v_n\}$ of an undirected weighted graph $G = (V, E)$.
- The weight $w_{ij} \geq 0$ of an edge $e = (v_i, v_j)$ is equal to some form of similarity measure of the data points $\mathbf{a}_i, \mathbf{a}_j$ which correspond to vertices v_i, v_j.
- If $w_{ij} = 0$ this means that vertices v_i and v_j correspond to completely dissimilar data points $\mathbf{a}_i, \mathbf{a}_j$ and in this case the graph does not include an edge of the form $e = (v_i, v_j)$.
- The similarity of vertices v_i and v_j can depend, for example, on a decreasing function of the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\|$ between the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$, such as $e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2}}$.
- Since the graph is undirected, $w_{ij} = w_{ji}$.
- We put all the weights w_{ij} into a symmetric *adjacency matrix* $W = (w_{ij})_{i,j=1}^n$.
- Recall that $w_{ij} = 0$ means that there is no edge between vertices v_i and v_j.
- Since the graph is weighted, the *degree* d_i of a vertex v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Similarity Graphs

- We represent a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ as the set of vertices $\{v_1, \ldots, v_n\}$ of an undirected weighted graph $G = (V, E)$.
- The weight $w_{ij} \geq 0$ of an edge $e = (v_i, v_j)$ is equal to some form of similarity measure of the data points $\mathbf{a}_i, \mathbf{a}_j$ which correspond to vertices v_i, v_j.
- If $w_{ij} = 0$ this means that vertices v_i and v_j correspond to completely dissimilar data points $\mathbf{a}_i, \mathbf{a}_j$ and in this case the graph does not include an edge of the form $e = (v_i, v_j)$.
- The similarity of vertices v_i and v_j can depend, for example, on a decreasing function of the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\|$ between the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$, such as $e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2}}$.
- Since the graph is undirected, $w_{ij} = w_{ji}$.
- We put all the weights w_{ij} into a symmetric *adjacency matrix* $W = (w_{ij})_{i,j=1}^{n}$.
- Recall that $w_{ij} = 0$ means that there is no edge between vertices v_i and v_j.
- Since the graph is weighted, the *degree* d_i of a vertex v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Similarity Graphs

- We represent a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ as the set of vertices $\{v_1, \ldots, v_n\}$ of an undirected weighted graph $G = (V, E)$.
- The weight $w_{ij} \geq 0$ of an edge $e = (v_i, v_j)$ is equal to some form of similarity measure of the data points $\mathbf{a}_i, \mathbf{a}_j$ which correspond to vertices v_i, v_j.
- If $w_{ij} = 0$ this means that vertices v_i and v_j correspond to completely dissimilar data points $\mathbf{a}_i, \mathbf{a}_j$ and in this case the graph does not include an edge of the form $e = (v_i, v_j)$.
- The similarity of vertices v_i and v_j can depend, for example, on a decreasing function of the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\|$ between the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$, such as $e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2}}$.
- Since the graph is undirected, $w_{ij} = w_{ji}$.
- We put all the weights w_{ij} into a symmetric *adjacency matrix* $W = (w_{ij})_{i,j=1}^n$.
- Recall that $w_{ij} = 0$ means that there is no edge between vertices v_i and v_j.
- Since the graph is weighted, the *degree* d_i of a vertex v_i is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

Similarity Graphs

- The *degree* matrix D is defined as a diagonal matrix with degree d_i of vertex v_i on the i^{th} entry of the diagonal of D and zeros everywhere off diagonal.
- So graph G is a compact summary of the set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$.
- The geometry of A is completely lost, and only pairwise similarities between the data points are preserved.
- This is actually good because it will allow us to handle clustering of points which is not centre based, but is based on "local similarity" of data points, as we will see later.
- Given a subset $S \subset V$ of vertices, we denote by $\overline{S}$ the complement $V \setminus S$ of S in V.
- Given a subset $S \subset V$ of vertices, we denote by $\mathbb{1}_S$ the indicator vector $\mathbb{1}_S = (f_1, \dots, f_n) \in \mathbb{R}^n$ where

$$f_i = \begin{cases} 1 & \text{if} \quad v_i \in S \\ 0 & \text{if} \quad v_i \notin S \end{cases}$$

- For simplicity, we abbreviate $v_i \in S$ as $i \in S$.

Similarity Graphs

- The *degree* matrix D is defined as a diagonal matrix with degree d_i of vertex v_i on the i^{th} entry of the diagonal of D and zeros everywhere off diagonal.
- So graph G is a compact summary of the set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$.
- The geometry of A is completely lost, and only pairwise similarities between the data points are preserved.
- This is actually good because it will allow us to handle clustering of points which is not centre based, but is based on "local similarity" of data points, as we will see later.
- Given a subset $S \subset V$ of vertices, we denote by $\overline{S}$ the complement $V \setminus S$ of S in V.
- Given a subset $S \subset V$ of vertices, we denote by $\mathbb{1}_S$ the indicator vector $\mathbb{1}_S = (f_1, \dots, f_n) \in \mathbb{R}^n$ where

$$f_i = \begin{cases} 1 & \text{if} \quad v_i \in S \\ 0 & \text{if} \quad v_i \notin S \end{cases}$$

- For simplicity, we abbreviate $v_i \in S$ as $i \in S$.

Similarity Graphs

- The *degree* matrix D is defined as a diagonal matrix with degree d_i of vertex v_i on the i^{th} entry of the diagonal of D and zeros everywhere off diagonal.
- So graph G is a compact summary of the set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.
- The geometry of A is completely lost, and only pairwise similarities between the data points are preserved.
- This is actually good because it will allow us to handle clustering of points which is not centre based, but is based on "local similarity" of data points, as we will see later.
- Given a subset $S \subset V$ of vertices, we denote by $\overline{S}$ the complement $V \setminus S$ of S in V.
- Given a subset $S \subset V$ of vertices, we denote by $\mathbb{1}_S$ the indicator vector $\mathbb{1}_S = (f_1, \ldots, f_n) \in \mathbb{R}^n$ where

$$f_i = \begin{cases} 1 & \text{if } \; v_i \in S \\ 0 & \text{if } \; v_i \notin S \end{cases}$$

- For simplicity, we abbreviate $v_i \in S$ as $i \in S$.

Similarity Graphs

- The *degree* matrix D is defined as a diagonal matrix with degree d_i of vertex v_i on the i^{th} entry of the diagonal of D and zeros everywhere off diagonal.
- So graph G is a compact summary of the set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$.
- The geometry of A is completely lost, and only pairwise similarities between the data points are preserved.
- This is actually good because it will allow us to handle clustering of points which is not centre based, but is based on "local similarity" of data points, as we will see later.
- Given a subset $S \subset V$ of vertices, we denote by $\overline{S}$ the complement $V \setminus S$ of S in V.
- Given a subset $S \subset V$ of vertices, we denote by $\mathbb{1}_S$ the indicator vector $\mathbb{1}_S = (f_1, \dots, f_n) \in \mathbb{R}^n$ where

$$f_i = \begin{cases} 1 & \text{if} \quad v_i \in S \\ 0 & \text{if} \quad v_i \notin S \end{cases}$$

- For simplicity, we abbreviate $v_i \in S$ as $i \in S$.

Similarity Graphs

- The *degree* matrix D is defined as a diagonal matrix with degree d_i of vertex v_i on the i^{th} entry of the diagonal of D and zeros everywhere off diagonal.
- So graph G is a compact summary of the set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.
- The geometry of A is completely lost, and only pairwise similarities between the data points are preserved.
- This is actually good because it will allow us to handle clustering of points which is not centre based, but is based on "local similarity" of data points, as we will see later.
- Given a subset $S \subset V$ of vertices, we denote by $\overline{S}$ the complement $V \setminus S$ of S in V.
- Given a subset $S \subset V$ of vertices, we denote by $\mathbb{1}_S$ the indicator vector $\mathbb{1}_S = (f_1, \ldots, f_n) \in \mathbb{R}^n$ where

$$f_i = \begin{cases} 1 & \text{if } \ v_i \in S \\ 0 & \text{if } \ v_i \notin S \end{cases}$$

- For simplicity, we abbreviate $v_i \in S$ as $i \in S$.

Similarity Graphs

- The *degree* matrix D is defined as a diagonal matrix with degree d_i of vertex v_i on the i^{th} entry of the diagonal of D and zeros everywhere off diagonal.
- So graph G is a compact summary of the set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.
- The geometry of A is completely lost, and only pairwise similarities between the data points are preserved.
- This is actually good because it will allow us to handle clustering of points which is not centre based, but is based on "local similarity" of data points, as we will see later.
- Given a subset $S \subset V$ of vertices, we denote by $\overline{S}$ the complement $V \setminus S$ of S in V.
- Given a subset $S \subset V$ of vertices, we denote by $\mathbb{1}_S$ the indicator vector $\mathbb{1}_S = (f_1, \ldots, f_n) \in \mathbb{R}^n$ where

$$f_i = \begin{cases} 1 & \text{if} \quad v_i \in S \\ 0 & \text{if} \quad v_i \notin S \end{cases}$$

- For simplicity, we abbreviate $v_i \in S$ as $i \in S$.

Similarity Graphs

- The *degree* matrix D is defined as a diagonal matrix with degree d_i of vertex v_i on the i^{th} entry of the diagonal of D and zeros everywhere off diagonal.
- So graph G is a compact summary of the set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$.
- The geometry of A is completely lost, and only pairwise similarities between the data points are preserved.
- This is actually good because it will allow us to handle clustering of points which is not centre based, but is based on "local similarity" of data points, as we will see later.
- Given a subset $S \subset V$ of vertices, we denote by $\overline{S}$ the complement $V \setminus S$ of S in V.
- Given a subset $S \subset V$ of vertices, we denote by $\mathbb{1}_S$ the indicator vector $\mathbb{1}_S = (f_1, \ldots, f_n) \in \mathbb{R}^n$ where

$$f_i = \begin{cases} 1 & \text{if} \quad v_i \in S \\ 0 & \text{if} \quad v_i \notin S \end{cases}$$

- For simplicity, we abbreviate $v_i \in S$ as $i \in S$.

Similarity Graphs

- For any two subsets $S, B \subset V$ we define

$$W(S,B) = \sum_{i \in S, j \in B} w_{ij}$$

  and for any set S we define two types of measurements of the "size" of S:

  1. $|S|$ is the number of elements in S;
  2. $\text{vol}(S) = \sum_{i \in S} d_i$

- Recall that $d_i = \sum_{j=1}^{n} w_{ij}$ is the degree of vertex v_i.
- A natural partition of vertices of a graph $G = (V, E)$ is into its connected components.

Similarity Graphs

- For any two subsets $S, B \subset V$ we define

$$W(S,B) = \sum_{i \in S, j \in B} w_{ij}$$

and for any set S we define two types of measurements of the "size" of S:

1. $|S|$ is the number of elements in S;
2. $\text{vol}(S) = \sum_{i \in S} d_i$

- Recall that $d_i = \sum_{j=1}^{n} w_{ij}$ is the degree of vertex v_i.
- A natural partition of vertices of a graph $G = (V, E)$ is into its connected components.

Similarity Graphs

- For any two subsets $S, B \subset V$ we define

$$W(S,B) = \sum_{i \in S, j \in B} w_{ij}$$

  and for any set S we define two types of measurements of the “size” of S:

  1. $|S|$ is the number of elements in S;
  2. $\mathrm{vol}(S) = \sum_{i \in S} d_i$

- Recall that $d_i = \sum_{j=1}^{n} w_{ij}$ is the degree of vertex v_i.
- A natural partition of vertices of a graph $G = (V, E)$ is into its connected components.

Similarity Graphs

- For any two subsets $S, B \subset V$ we define

$$W(S,B) = \sum_{i \in S, j \in B} w_{ij}$$

  and for any set S we define two types of measurements of the “size” of S:

  1. $|S|$ is the number of elements in S;
  2. $\mathrm{vol}(S) = \sum_{i \in S} d_i$

- Recall that $d_i = \sum_{j=1}^{n} w_{ij}$ is the degree of vertex v_i.
- A natural partition of vertices of a graph $G = (V, E)$ is into its connected components.

Similarity Graphs

- For any two subsets $S, B \subset V$ we define

$$W(S,B) = \sum_{i \in S, j \in B} w_{ij}$$

  and for any set S we define two types of measurements of the "size" of S:

  1. $|S|$ is the number of elements in S;
  2. $\text{vol}(S) = \sum_{i \in S} d_i$

- Recall that $d_i = \sum_{j=1}^{n} w_{ij}$ is the degree of vertex v_i.
- A natural partition of vertices of a graph $G = (V, E)$ is into its connected components.

Similarity Graphs

- Given a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ we associate vertices $\{v_1, \ldots, v_n\}$ of a similarity graph G, but there are many ways how we can associate weights w_{ij} which measure the similarity of data points $\mathbf{a}_i$ and $\mathbf{a}_j$ that correspond to vertices v_i and v_j.

  (1) **The ε-neighbourhood graph:**

  - We connect all pairs of vertices v_i, v_j such that the distances between the associated data points $\mathbf{a}_i, \mathbf{a}_j$ are smaller than ε.
  - The distance is usually the Euclidean distance $\|\mathbf{a}_i - \mathbf{a}_j\| = \sqrt{\sum_{p=1}^{d}(a_{ip} - a_{jp})^2}$, where $\mathbf{a}_i = (a_{i1}, \ldots, a_{id}) \in \mathbb{R}^d$.

Similarity Graphs

(2) **The k-nearest neighbour graphs:**

- There are two flavours of k-nearest neighbour graphs:
  1. *Unidirectional k-nearest neighbour graph.* We connect v_i with v_j if either v_j is among k nearest neighbours of v_i or vice versa, v_i is among k nearest neighbours of v_j.
  2. *Mutual k-nearest neighbour graph.* We connect v_i with v_j if both v_j is among k nearest neighbours of v_i and v_i is also among k closest neighbours of v_j.
- In both cases the edge is then weighted with the degree of similarity of the vertices v_i and v_j.

Similarity Graphs

(2) **The k-nearest neighbour graphs:**

- There are two flavours of k-nearest neighbour graphs:
    1. *Unidirectional k-nearest neighbour graph.* We connect v_i with v_j if either v_j is among k nearest neighbours of v_i or vice versa, v_i is among k nearest neighbours of v_j.
    2. *Mutual k-nearest neighbour graph.* We connect v_i with v_j if both v_j is among k nearest neighbours of v_i and v_i is also among k closest neighbours of v_j.
- In both cases the edge is then weighted with the degree of similarity of the vertices v_i and v_j.

Similarity Graphs

(2) **The k-nearest neighbour graphs:**

- There are two flavours of k-nearest neighbour graphs:
  1. *Unidirectional k-nearest neighbour graph.* We connect v_i with v_j if either v_j is among k nearest neighbours of v_i or vice versa, v_i is among k nearest neighbours of v_j.
  2. *Mutual k-nearest neighbour graph.* We connect v_i with v_j if both v_j is among k nearest neighbours of v_i and v_i is also among k closest neighbours of v_j.
- In both cases the edge is then weighted with the degree of similarity of the vertices v_i and v_j.

Similarity Graphs

(2) **The k-nearest neighbour graphs:**

- There are two flavours of k-nearest neighbour graphs:
    1. *Unidirectional k-nearest neighbour graph.* We connect v_i with v_j if either v_j is among k nearest neighbours of v_i or vice versa, v_i is among k nearest neighbours of v_j.
    2. *Mutual k-nearest neighbour graph.* We connect v_i with v_j if both v_j is among k nearest neighbours of v_i and v_i is also among k closest neighbours of v_j.
- In both cases the edge is then weighted with the degree of similarity of the vertices v_i and v_j.

Similarity Graphs

(3) **The fully connected graphs:**

- We simply connect all pairs of vertices v_i and v_j for which the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$ have a strictly positive similarity, or similarity higher than some prescribed threshold ε.
- To ensure that such a graph represents local neighbourhood relationships, the similarity measure must be chosen to respect such localisation condition.
- Often we take weights which reflect such local similarities by the following formula:

$$\mathrm{w}_{ij} = e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma^2}}$$

- Here σ is a parameter which determines "the size" of the neighbourhood, namely how fast the similarity decreases as distance increases.

▶ Unfortunately, there is not a simple way how to choose a similarity graph.
▶ The best is just to try several and pick the one which eventually produces the most informative clustering.

Similarity Graphs

(3) **The fully connected graphs:**

- We simply connect all pairs of vertices v_i and v_j for which the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$ have a strictly positive similarity, or similarity higher than some prescribed threshold ε.
- To ensure that such a graph represents local neighbourhood relationships, the similarity measure must be chosen to respect such localisation condition.
- Often we take weights which reflect such local similarities by the following formula:

$$\mathrm{w}_{ij} = e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma^2}}$$

- Here σ is a parameter which determines "the size" of the neighbourhood, namely how fast the similarity decreases as distance increases.

▶ Unfortunately, there is not a simple way how to choose a similarity graph.
▶ The best is just to try several and pick the one which eventually produces the most informative clustering.

Similarity Graphs

(3) **The fully connected graphs:**

- We simply connect all pairs of vertices v_i and v_j for which the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$ have a strictly positive similarity, or similarity higher than some prescribed threshold ε.
- To ensure that such a graph represents local neighbourhood relationships, the similarity measure must be chosen to respect such localisation condition.
- Often we take weights which reflect such local similarities by the following formula:

$$\mathrm{w}_{ij} = e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma^2}}$$

- Here σ is a parameter which determines "the size" of the neighbourhood, namely how fast the similarity decreases as distance increases.

▶ Unfortunately, there is not a simple way how to choose a similarity graph.
▶ The best is just to try several and pick the one which eventually produces the most informative clustering.

Similarity Graphs

(3) **The fully connected graphs:**

- We simply connect all pairs of vertices v_i and v_j for which the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$ have a strictly positive similarity, or similarity higher than some prescribed threshold ε.
- To ensure that such a graph represents local neighbourhood relationships, the similarity measure must be chosen to respect such localisation condition.
- Often we take weights which reflect such local similarities by the following formula:

$$\mathrm{w}_{ij} = e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma^2}}$$

- Here σ is a parameter which determines "the size" of the neighbourhood, namely how fast the similarity decreases as distance increases.

▶ Unfortunately, there is not a simple way how to choose a similarity graph.

▶ The best is just to try several and pick the one which eventually produces the most informative clustering.

Similarity Graphs

(3) **The fully connected graphs:**

- We simply connect all pairs of vertices v_i and v_j for which the corresponding data points $\mathbf{a}_i$ and $\mathbf{a}_j$ have a strictly positive similarity, or similarity higher than some prescribed threshold ε.
- To ensure that such a graph represents local neighbourhood relationships, the similarity measure must be chosen to respect such localisation condition.
- Often we take weights which reflect such local similarities by the following formula:

$$\mathrm{w}_{ij} = e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma^2}}$$

- Here σ is a parameter which determines "the size" of the neighbourhood, namely how fast the similarity decreases as distance increases.

▶ Unfortunately, there is not a simple way how to choose a similarity graph.
▶ The best is just to try several and pick the one which eventually produces the most informative clustering.

Spectral Graph Theory

- Recall that the $n \times n$ diagonal matrix D has the degrees d_i of vertices v_i on its diagonal, where $d_i = \sum_{j=1}^{n} w_{ij}$.
- The (unnormalised) graph Laplacian matrix L is defined as

$$L = D - W$$

  where $W = (w_{ij})_{i,j=1}^{n}$.
- Clearly, L is symmetric and it does not depend on w_{ii}, $1 \leq i \leq n$.
- Graph Laplacians are crucial for spectral clustering.
- A matrix M of size $n \times n$ is *positive semi-definite* if for all vectors $f \in \mathbb{R}^n$ we have

$$f^\top M f \geq 0$$

- From linear algebra we know that a symmetric matrix is positive semi-definite iff all of its eigenvalues are real and non-negative.
- The next theorem summarises their main properties important for spectral clustering.

Spectral Graph Theory

- Recall that the $n \times n$ diagonal matrix D has the degrees d_i of vertices v_i on its diagonal, where $d_i = \sum_{j=1}^{n} w_{ij}$.
- The (unnormalised) graph Laplacian matrix L is defined as

$$L = D - W$$

  where $W = (w_{ij})_{i,j=1}^{n}$.
- Clearly, L is symmetric and it does not depend on w_{ii}, $1 \leq i \leq n$.
- Graph Laplacians are crucial for spectral clustering.
- A matrix M of size $n \times n$ is *positive semi-definite* if for all vectors $f \in \mathbb{R}^n$ we have

$$f^\top M f \geq 0$$

- From linear algebra we know that a symmetric matrix is positive semi-definite iff all of its eigenvalues are real and non-negative.
- The next theorem summarises their main properties important for spectral clustering.

Spectral Graph Theory

- Recall that the $n \times n$ diagonal matrix D has the degrees d_i of vertices v_i on its diagonal, where $d_i = \sum_{j=1}^{n} w_{ij}$.
- The (unnormalised) graph Laplacian matrix L is defined as

$$L = D - W$$

  where $W = (w_{ij})_{i,j=1}^{n}$.
- Clearly, L is symmetric and it does not depend on $w_{ii},\ 1 \leq i \leq n$.
- Graph Laplacians are crucial for spectral clustering.
- A matrix M of size $n \times n$ is *positive semi-definite* if for all vectors $f \in \mathbb{R}^n$ we have

$$f^\top M f \geq 0$$

- From linear algebra we know that a symmetric matrix is positive semi-definite iff all of its eigenvalues are real and non-negative.
- The next theorem summarises their main properties important for spectral clustering.

Spectral Graph Theory

- Recall that the $n \times n$ diagonal matrix D has the degrees d_i of vertices v_i on its diagonal, where $d_i = \sum_{j=1}^{n} w_{ij}$.
- The (unnormalised) graph Laplacian matrix L is defined as

$$L = D - W$$

  where $W = (w_{ij})_{i,j=1}^{n}$.
- Clearly, L is symmetric and it does not depend on w_{ii}, $1 \leq i \leq n$.
- Graph Laplacians are crucial for spectral clustering.
- A matrix M of size $n \times n$ is *positive semi-definite* if for all vectors $f \in \mathbb{R}^n$ we have

$$f^{\top} M f \geq 0$$

- From linear algebra we know that a symmetric matrix is positive semi-definite iff all of its eigenvalues are real and non-negative.
- The next theorem summarises their main properties important for spectral clustering.

Spectral Graph Theory

- Recall that the $n \times n$ diagonal matrix D has the degrees d_i of vertices v_i on its diagonal, where $d_i = \sum_{j=1}^{n} w_{ij}$.
- The (unnormalised) graph Laplacian matrix L is defined as

$$L = D - W$$

  where $W = (w_{ij})_{i,j=1}^{n}$.
- Clearly, L is symmetric and it does not depend on w_{ii}, $1 \le i \le n$.
- Graph Laplacians are crucial for spectral clustering.
- A matrix M of size $n \times n$ is *positive semi-definite* if for all vectors $f \in \mathbb{R}^n$ we have

$$f^\mathsf{T} M f \ge 0$$

- From linear algebra we know that a symmetric matrix is positive semi-definite iff all of its eigenvalues are real and non-negative.
- The next theorem summarises their main properties important for spectral clustering.

Spectral Graph Theory

- Recall that the $n \times n$ diagonal matrix D has the degrees d_i of vertices v_i on its diagonal, where $d_i = \sum_{j=1}^{n} w_{ij}$.
- The (unnormalised) graph Laplacian matrix L is defined as

$$L = D - W$$

  where $W = (w_{ij})_{i,j=1}^{n}$.
- Clearly, L is symmetric and it does not depend on $w_{ii},\ 1 \leq i \leq n$.
- Graph Laplacians are crucial for spectral clustering.
- A matrix M of size $n \times n$ is *positive semi-definite* if for all vectors $f \in \mathbb{R}^n$ we have

$$f^{\mathsf{T}} M f \geq 0$$

- From linear algebra we know that a symmetric matrix is positive semi-definite iff all of its eigenvalues are real and non-negative.
- The next theorem summarises their main properties important for spectral clustering.

Spectral Graph Theory

- Recall that the $n \times n$ diagonal matrix D has the degrees d_i of vertices v_i on its diagonal, where $d_i = \sum_{j=1}^{n} w_{ij}$.
- The (unnormalised) graph Laplacian matrix L is defined as

$$L = D - W$$

  where $W = (w_{ij})_{i,j=1}^{n}$.
- Clearly, L is symmetric and it does not depend on w_{ii}, $1 \leq i \leq n$.
- Graph Laplacians are crucial for spectral clustering.
- A matrix M of size $n \times n$ is *positive semi-definite* if for all vectors $f \in \mathbb{R}^n$ we have

$$f^{\mathsf{T}} M f \geq 0$$

- From linear algebra we know that a symmetric matrix is positive semi-definite iff all of its eigenvalues are real and non-negative.
- The next theorem summarises their main properties important for spectral clustering.

Spectral Graph Theory

- **Theorem:** The matrix $L = D - W$ has the following properties:

  (1) For every vector $f \in \mathbb{R}^n$,

  $$f^\top L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

  (2) L is a symmetric positive semi-definite matrix.

  (3) The smallest eigenvalue of L is 0 and its corresponding eigenvector is $\mathbb{1} = (1, 1, \ldots, 1)$.

- **Proof:**

  (1) $$f^\top L f = f^\top D f - f^\top W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j$$

  $$= \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{n} w_{ij}\right) f_i^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n}\left(\sum_{i=1}^{n} w_{ij}\right) f_j^2\right)$$

  $$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Spectral Graph Theory

- **Theorem:** The matrix $L = D - W$ has the following properties:
  (1) For every vector $f \in \mathbb{R}^n$,
  $$f^\top L f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$
  (2) L is a symmetric positive semi-definite matrix.
  (3) The smallest eigenvalue of L is 0 and its corresponding eigenvector is $\mathbb{1} = (1, 1, \ldots, 1)$.
- **Proof:**

$$(1) \quad f^\top L f = f^\top D f - f^\top W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{n} w_{ij}\right) f_i^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n}\left(\sum_{i=1}^{n} w_{ij}\right) f_j^2\right)$$

$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Spectral Graph Theory

- **Theorem:** The matrix $L = D - W$ has the following properties:
  (1) For every vector $f \in \mathbb{R}^n$,
$$f^\top L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$
  (2) L is a symmetric positive semi-definite matrix.
  (3) The smallest eigenvalue of L is 0 and its corresponding eigenvector is $\mathbb{1} = (1, 1, \ldots, 1)$.
- **Proof:**

(1) $$f^\top L f = f^\top D f - f^\top W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j$$
$$= \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{n} w_{ij}\right) f_i^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n}\left(\sum_{i=1}^{n} w_{ij}\right) f_j^2\right)$$
$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Spectral Graph Theory

- **Theorem:** The matrix $L = D - W$ has the following properties:
  (1) For every vector $f \in \mathbb{R}^n$,
  $$f^\top L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$
  (2) L is a symmetric positive semi-definite matrix.
  (3) The smallest eigenvalue of L is 0 and its corresponding eigenvector is $\mathbb{1} = (1, 1, \ldots, 1)$.
- **Proof:**
  (1) $$f^\top L f = f^\top D f - f^\top W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j$$
  $$= \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{n} w_{ij}\right) f_i^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n}\left(\sum_{i=1}^{n} w_{ij}\right) f_j^2\right)$$
  $$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Spectral Graph Theory

- **Theorem:** The matrix $L = D - W$ has the following properties:
  (1) For every vector $f \in \mathbb{R}^n$,
  $$f^\top L f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$
  (2) L is a symmetric positive semi-definite matrix.
  (3) The smallest eigenvalue of L is 0 and its corresponding eigenvector is $\mathbb{1} = (1, 1, \ldots, 1)$.
- **Proof:**

(1)
$$f^\top L f = f^\top D f - f^\top W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j$$
$$= \frac{1}{2}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{n} w_{ij}\right) f_i^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n}\left(\sum_{i=1}^{n} w_{ij}\right) f_j^2\right)$$
$$= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Spectral Graph Theory

(2) Since we have shown that $f^\top Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$ and since $w_{ij} \geq 0$, L satisfies $f^\top Lf \geq 0$ for all vectors f and is thus positive semi-definite.

(3) Note that $L\mathbb{1} = D\mathbb{1} - W\mathbb{1}$. However, it is easy to see that both $D\mathbb{1}$ and $W\mathbb{1}$ produce the same vector with the i^{th} coordinate equal to the degree d_i of vertex v_i. Thus, $L\mathbb{1} = D\mathbb{1} - W\mathbb{1} = \mathbf{0} = 0 \cdot \mathbb{1}$. So, since $L\mathbb{1} = 0 \cdot \mathbb{1}$, 0 is the smallest eigenvalue of L (because all eigenvalues are non-negative) and $\mathbb{1}$ is the corresponding eigenvector.

- Let M be an $n \times n$ matrix. Then an eigenvalue λ of this matrix has:
  1. An algebraic multiplicity k if the characteristic polynomial $P_n(x)$ has a form $P_n(x) = (x - \lambda)^k Q(x)$ where $Q(x)$ is a polynomial of degree $n - k$.
  2. A geometric multiplicity k if λ has k linearly independent eigenvectors.

Spectral Graph Theory

(2) Since we have shown that $f^\top L f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$ and since $w_{ij} \geq 0$, L satisfies $f^\top L f \geq 0$ for all vectors f and is thus positive semi-definite.

(3) Note that $L\mathbb{1} = D\mathbb{1} - W\mathbb{1}$. However, it is easy to see that both $D\mathbb{1}$ and $W\mathbb{1}$ produce the same vector with the i^{th} coordinate equal to the degree d_i of vertex v_i. Thus, $L\mathbb{1} = D\mathbb{1} - W\mathbb{1} = \mathbf{0} = 0 \cdot \mathbb{1}$. So, since $L\mathbb{1} = 0 \cdot \mathbb{1}$, 0 is the smallest eigenvalue of L (because all eigenvalues are non-negative) and $\mathbb{1}$ is the corresponding eigenvector.

- Let M be an $n \times n$ matrix. Then an eigenvalue λ of this matrix has:
  1. An algebraic multiplicity k if the characteristic polynomial $P_n(x)$ has a form $P_n(x) = (x-\lambda)^k Q(x)$ where $Q(x)$ is a polynomial of degree $n-k$.
  2. A geometric multiplicity k if λ has k linearly independent eigenvectors.

Spectral Graph Theory

(2) Since we have shown that $f^{\mathsf{T}} L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2$ and since $w_{ij} \geq 0$, L satisfies $f^{\mathsf{T}} L f \geq 0$ for all vectors f and is thus positive semi-definite.

(3) Note that $L\mathbb{1} = D\mathbb{1} - W\mathbb{1}$. However, it is easy to see that both $D\mathbb{1}$ and $W\mathbb{1}$ produce the same vector with the i^{th} coordinate equal to the degree d_i of vertex v_i. Thus, $L\mathbb{1} = D\mathbb{1} - W\mathbb{1} = \mathbf{0} = 0 \cdot \mathbb{1}$. So, since $L\mathbb{1} = 0 \cdot \mathbb{1}$, 0 is the smallest eigenvalue of L (because all eigenvalues are non-negative) and $\mathbb{1}$ is the corresponding eigenvector.

- Let M be an $n \times n$ matrix. Then an eigenvalue λ of this matrix has:
  1. An algebraic multiplicity k if the characteristic polynomial $P_n(x)$ has a form $P_n(x) = (x - \lambda)^k Q(x)$ where $Q(x)$ is a polynomial of degree $n - k$.
  2. A geometric multiplicity k if λ has k linearly independent eigenvectors.

Spectral Graph Theory

(2) Since we have shown that $f^\top L f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$ and since $w_{ij} \geq 0$, L satisfies $f^\top L f \geq 0$ for all vectors f and is thus positive semi-definite.

(3) Note that $L\mathbb{1} = D\mathbb{1} - W\mathbb{1}$. However, it is easy to see that both $D\mathbb{1}$ and $W\mathbb{1}$ produce the same vector with the i^{th} coordinate equal to the degree d_i of vertex v_i. Thus, $L\mathbb{1} = D\mathbb{1} - W\mathbb{1} = \mathbf{0} = 0 \cdot \mathbb{1}$. So, since $L\mathbb{1} = 0 \cdot \mathbb{1}$, 0 is the smallest eigenvalue of L (because all eigenvalues are non-negative) and $\mathbb{1}$ is the corresponding eigenvector.

- Let M be an $n \times n$ matrix. Then an eigenvalue λ of this matrix has:
  1. An algebraic multiplicity k if the characteristic polynomial $P_n(x)$ has a form $P_n(x) = (x - \lambda)^k Q(x)$ where $Q(x)$ is a polynomial of degree $n - k$.
  2. A geometric multiplicity k if λ has k linearly independent eigenvectors.

Spectral Graph Theory

(2) Since we have shown that $f^\top Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$ and since $w_{ij} \geq 0$, L satisfies $f^\top Lf \geq 0$ for all vectors f and is thus positive semi-definite.

(3) Note that $L\mathbb{1} = D\mathbb{1} - W\mathbb{1}$. However, it is easy to see that both $D\mathbb{1}$ and $W\mathbb{1}$ produce the same vector with the i^{th} coordinate equal to the degree d_i of vertex v_i. Thus, $L\mathbb{1} = D\mathbb{1} - W\mathbb{1} = \mathbf{0} = 0 \cdot \mathbb{1}$. So, since $L\mathbb{1} = 0 \cdot \mathbb{1}$, 0 is the smallest eigenvalue of L (because all eigenvalues are non-negative) and $\mathbb{1}$ is the corresponding eigenvector.

- Let M be an $n \times n$ matrix. Then an eigenvalue λ of this matrix has:
  1. An algebraic multiplicity k if the characteristic polynomial $P_n(x)$ has a form $P_n(x) = (x - \lambda)^k Q(x)$ where $Q(x)$ is a polynomial of degree $n - k$.
  2. A geometric multiplicity k if λ has k linearly independent eigenvectors.

Spectral Graph Theory

- **Theorem:** Let G be an undirected weighted graph with n vertices and non-negative weights which has exactly k connected components $A_1, \ldots, A_k$. Then the algebraic and geometric multiplicities of the eigenvalue 0 of L are both equal to k and the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.
- **Proof:** Assume first that the graph is connected, i.e., that it has exactly one connected component and let $f \in \mathbb{R}^n$ be an eigenvector with eigenvalue 0. Then, $Lf = 0 \cdot f = \mathbf{0}$ and thus also $f^\top Lf = 0$. Consequently, by the previous theorem we have

$$0 = f^\top Lf = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

Thus, if two vertices v_i and v_j are connected by an edge, then $w_{ij} > 0$ and consequently $f_i = f_j$. Going along any path we get that the coordinates of f must be constant at all vertices along that path. Since G is connected, obviously all coordinates of f must be constant. Thus, $\mathbb{1}_V = (1, 1, \ldots, 1)$ is an eigenvector. (Note that we do not normalise eigenvectors).

Spectral Graph Theory

- **Theorem:** Let G be an undirected weighted graph with n vertices and non-negative weights which has exactly k connected components $A_1, \dots, A_k$. Then the algebraic and geometric multiplicities of the eigenvalue 0 of L are both equal to k and the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \dots, \mathbb{1}_{A_k}$ of those components.
- **Proof:** Assume first that the graph is connected, i.e., that it has exactly one connected component and let $f \in \mathbb{R}^n$ be an eigenvector with eigenvalue 0. Then, $Lf = 0 \cdot f = \mathbf{0}$ and thus also $f^{\mathsf{T}} L f = 0$. Consequently, by the previous theorem we have

$$0 = f^{\mathsf{T}} L f = \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2$$

Thus, if two vertices v_i and v_j are connected by an edge, then $w_{ij} > 0$ and consequently $f_i = f_j$. Going along any path we get that the coordinates of f must be constant at all vertices along that path. Since G is connected, obviously all coordinates of f must be constant. Thus, $\mathbb{1}_V = (1, 1, \dots, 1)$ is an eigenvector. (Note that we do not normalise eigenvectors).

Spectral Graph Theory

- **Theorem:** Let G be an undirected weighted graph with n vertices and non-negative weights which has exactly k connected components $A_1, \ldots, A_k$. Then the algebraic and geometric multiplicities of the eigenvalue 0 of L are both equal to k and the eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.
- **Proof:** Assume first that the graph is connected, i.e., that it has exactly one connected component and let $f \in \mathbb{R}^n$ be an eigenvector with eigenvalue 0. Then, $Lf = 0 \cdot f = \mathbf{0}$ and thus also $f^{\mathsf{T}} L f = 0$. Consequently, by the previous theorem we have

$$0 = f^{\mathsf{T}} L f = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

Thus, if two vertices v_i and v_j are connected by an edge, then $w_{ij} > 0$ and consequently $f_i = f_j$. Going along any path we get that the coordinates of f must be constant at all vertices along that path. Since G is connected, obviously all coordinates of f must be constant. Thus, $\mathbb{1}_V = (1, 1, \ldots, 1)$ is an eigenvector. (Note that we do not normalise eigenvectors).

Spectral Graph Theory

- **Proof (continued):** Assume now that G has k connected components. We can assume that the vertices of the connected components are listed in order of the component they belong to. Thus, matrix W is a block matrix of the form

$$W = \begin{pmatrix} W_1 & & & 0 \\ & W_2 & & \\ & & \ddots & \\ 0 & & & W_k \end{pmatrix}$$

  where the block W_i corresponds to the connected component A_i and with 0's outside the blocks W_i.

- But then $L = D - W$ also has the same structure:

$$L = \begin{pmatrix} L_1 & & & 0 \\ & L_2 & & \\ & & \ddots & \\ 0 & & & L_k \end{pmatrix}$$

Spectral Graph Theory

- **Proof (continued):** Assume now that G has k connected components. We can assume that the vertices of the connected components are listed in order of the component they belong to. Thus, matrix W is a block matrix of the form

$$W = \begin{pmatrix} W_1 & & & 0 \\ & W_2 & & \\ & & \ddots & \\ 0 & & & W_k \end{pmatrix}$$

  where the block W_i corresponds to the connected component A_i and with 0's outside the blocks W_i.

- But then $L = D - W$ also has the same structure:

$$L = \begin{pmatrix} L_1 & & & 0 \\ & L_2 & & \\ & & \ddots & \\ 0 & & & L_k \end{pmatrix}$$

Spectral Graph Theory

- **Proof (continued):** The eigenvalues of block matrices are the union of the eigenvalues of each block as a separate matrix, and the eigenvectors of a block matrix are the eigenvectors of the blocks with zeros appended outside each block.
- Thus, since each L_i corresponds to a single connected component, it has 0 as the smallest eigenvalue with an eigenvector $\mathbb{1}_{A_i}$.
- Consequently, the matrix L has as many eigenvalues 0 as there are connected components, and the corresponding eigenvectors are the indicator vectors of the connected components.

Spectral Graph Theory

- **Proof (continued):** The eigenvalues of block matrices are the union of the eigenvalues of each block as a separate matrix, and the eigenvectors of a block matrix are the eigenvectors of the blocks with zeros appended outside each block.
- Thus, since each L_i corresponds to a single connected component, it has 0 as the smallest eigenvalue with an eigenvector $\mathbb{1}_{A_i}$.
- Consequently, the matrix L has as many eigenvalues 0 as there are connected components, and the corresponding eigenvectors are the indicator vectors of the connected components.

Spectral Graph Theory

- **Proof (continued):** The eigenvalues of block matrices are the union of the eigenvalues of each block as a separate matrix, and the eigenvectors of a block matrix are the eigenvectors of the blocks with zeros appended outside each block.
- Thus, since each L_i corresponds to a single connected component, it has 0 as the smallest eigenvalue with an eigenvector $\mathbb{1}_{A_i}$.
- Consequently, the matrix L has as many eigenvalues 0 as there are connected components, and the corresponding eigenvectors are the indicator vectors of the connected components.

Spectral Graph Theory

- Assume now that a graph G has k connected components which we would like to find.
- After forming the Laplacian matrix L we would use a standard software to find its eigenvalues.
- Such a software would output that 0 is an eigenvalue of L with multiplicity k and it would output k eigenvectors corresponding to eigenvalue 0.
- However, generally, these vectors would not be the indicators $\mathbb{1}_{A_i}$ of the connected components but k (mutually orthogonal) linear combinations of these indicator vectors, because any k orthogonal vectors in the eigensubspace corresponding to the eigenvalue 0 are equally good candidates.
- Thus, we will obtain k vectors from $\mathbb{R}^n$ which look like this:

$$\begin{aligned} \mathbf{e}_1 &= \alpha_1^1 \mathbb{1}_{A_1} + \alpha_2^1 \mathbb{1}_{A_2} + \ldots + \alpha_k^1 \mathbb{1}_{A_k} \\ &\ldots \qquad \ldots \qquad \ldots \\ \mathbf{e}_k &= \alpha_1^k \mathbb{1}_{A_1} + \alpha_2^k \mathbb{1}_{A_2} + \ldots + \alpha_k^k \mathbb{1}_{A_k} \end{aligned}$$

Spectral Graph Theory

- Assume now that a graph G has k connected components which we would like to find.
- After forming the Laplacian matrix L we would use a standard software to find its eigenvalues.
- Such a software would output that 0 is an eigenvalue of L with multiplicity k and it would output k eigenvectors corresponding to eigenvalue 0.
- However, generally, these vectors would not be the indicators $\mathbb{1}_{A_i}$ of the connected components but k (mutually orthogonal) linear combinations of these indicator vectors, because any k orthogonal vectors in the eigensubspace corresponding to the eigenvalue 0 are equally good candidates.
- Thus, we will obtain k vectors from $\mathbb{R}^n$ which look like this:

$$\begin{aligned} \mathbf{e}_1 &= \alpha_1^1 \mathbb{1}_{A_1} + \alpha_2^1 \mathbb{1}_{A_2} + \ldots + \alpha_k^1 \mathbb{1}_{A_k} \\ &\ldots \qquad \ldots \qquad \ldots \\ \mathbf{e}_k &= \alpha_1^k \mathbb{1}_{A_1} + \alpha_2^k \mathbb{1}_{A_2} + \ldots + \alpha_k^k \mathbb{1}_{A_k} \end{aligned}$$

Spectral Graph Theory

- Assume now that a graph G has k connected components which we would like to find.
- After forming the Laplacian matrix L we would use a standard software to find its eigenvalues.
- Such a software would output that 0 is an eigenvalue of L with multiplicity k and it would output k eigenvectors corresponding to eigenvalue 0.
- However, generally, these vectors would not be the indicators $\mathbb{1}_{A_i}$ of the connected components but k (mutually orthogonal) linear combinations of these indicator vectors, because any k orthogonal vectors in the eigensubspace corresponding to the eigenvalue 0 are equally good candidates.
- Thus, we will obtain k vectors from $\mathbb{R}^n$ which look like this:

$$\begin{aligned} \mathbf{e}_1 &= \alpha_1^1 \mathbb{1}_{A_1} + \alpha_2^1 \mathbb{1}_{A_2} + \ldots + \alpha_k^1 \mathbb{1}_{A_k} \\ \ldots & \qquad \ldots \qquad \ldots \\ \mathbf{e}_k &= \alpha_1^k \mathbb{1}_{A_1} + \alpha_2^k \mathbb{1}_{A_2} + \ldots + \alpha_k^k \mathbb{1}_{A_k} \end{aligned}$$

Spectral Graph Theory

- Assume now that a graph G has k connected components which we would like to find.
- After forming the Laplacian matrix L we would use a standard software to find its eigenvalues.
- Such a software would output that 0 is an eigenvalue of L with multiplicity k and it would output k eigenvectors corresponding to eigenvalue 0.
- However, generally, these vectors would not be the indicators $\mathbb{1}_{A_i}$ of the connected components but k (mutually orthogonal) linear combinations of these indicator vectors, because any k orthogonal vectors in the eigensubspace corresponding to the eigenvalue 0 are equally good candidates.
- Thus, we will obtain k vectors from $\mathbb{R}^n$ which look like this:

$$\begin{aligned}\mathbf{e}_1 &= \alpha_1^1 \mathbb{1}_{A_1} + \alpha_2^1 \mathbb{1}_{A_2} + \ldots + \alpha_k^1 \mathbb{1}_{A_k} \\ \ldots & \qquad \ldots \qquad \ldots \\ \mathbf{e}_k &= \alpha_1^k \mathbb{1}_{A_1} + \alpha_2^k \mathbb{1}_{A_2} + \ldots + \alpha_k^k \mathbb{1}_{A_k}\end{aligned}$$

Spectral Graph Theory

- Assume now that a graph G has k connected components which we would like to find.
- After forming the Laplacian matrix L we would use a standard software to find its eigenvalues.
- Such a software would output that 0 is an eigenvalue of L with multiplicity k and it would output k eigenvectors corresponding to eigenvalue 0.
- However, generally, these vectors would not be the indicators $\mathbb{1}_{A_i}$ of the connected components but k (mutually orthogonal) linear combinations of these indicator vectors, because any k orthogonal vectors in the eigensubspace corresponding to the eigenvalue 0 are equally good candidates.
- Thus, we will obtain k vectors from $\mathbb{R}^n$ which look like this:

$$\begin{aligned}
\mathbf{e}_1 &= \alpha_1^1 \mathbb{1}_{A_1} + \alpha_2^1 \mathbb{1}_{A_2} + \ldots + \alpha_k^1 \mathbb{1}_{A_k} \\
&\ldots \qquad \ldots \qquad \ldots \\
\mathbf{e}_k &= \alpha_1^k \mathbb{1}_{A_1} + \alpha_2^k \mathbb{1}_{A_2} + \ldots + \alpha_k^k \mathbb{1}_{A_k}
\end{aligned}$$

Spectral Graph Theory

- We can now form a matrix E of size $n \times k$ whose k columns are the eigenvectors $\mathbf{e}_i \in \mathbb{R}^n$. We now consider n vectors $\mathbf{y}_j \in \mathbb{R}^k$ which are the rows of E.
- Note that for every two vertices v_i, v_j which belongs to the same connected component A_m the corresponding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ are identical and equal $(\alpha_m^1, \alpha_m^2, \ldots, \alpha_m^k)$ because only $\mathbb{1}_{A_m}$ has 1's at positions i and j; all other $\mathbb{1}_{A_l}$ for $l \neq m$ have zeros at these positions, given that v_i, v_j belong to the connected component A_m and that the components are disjoint.
- This allows us to easily identify the connected components, by selecting points i with identical corresponding vectors v_i.

Spectral Graph Theory

- We can now form a matrix E of size $n \times k$ whose k columns are the eigenvectors $\mathbf{e}_i \in \mathbb{R}^n$. We now consider n vectors $\mathbf{y}_j \in \mathbb{R}^k$ which are the rows of E.
- Note that for every two vertices v_i, v_j which belongs to the same connected component A_m the corresponding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ are identical and equal $(\alpha_m^1, \alpha_m^2, \dots, \alpha_m^k)$ because only $\mathbb{1}_{A_m}$ has 1's at positions i and j; all other $\mathbb{1}_{A_l}$ for $l \neq m$ have zeros at these positions, given that v_i, v_j belong to the connected component A_m and that the components are disjoint.
- This allows us to easily identify the connected components, by selecting points i with identical corresponding vectors v_i.

Spectral Graph Theory

- We can now form a matrix E of size $n \times k$ whose k columns are the eigenvectors $\mathbf{e}_i \in \mathbb{R}^n$. We now consider n vectors $\mathbf{y}_j \in \mathbb{R}^k$ which are the rows of E.
- Note that for every two vertices v_i, v_j which belongs to the same connected component A_m the corresponding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ are identical and equal $(\alpha_m^1, \alpha_m^2, \ldots, \alpha_m^k)$ because only $\mathbb{1}_{A_m}$ has 1's at positions i and j; all other $\mathbb{1}_{A_l}$ for $l \neq m$ have zeros at these positions, given that v_i, v_j belong to the connected component A_m and that the components are disjoint.
- This allows us to easily identify the connected components, by selecting points i with identical corresponding vectors v_i.

Spectral Clustering

- When we cluster points into disjoint clusters we want any two vertices v_i, v_j with high similarity (i.e., with high weight w_{ij} of the corresponding edge (v_i, v_j)) to be in the same cluster, and any two vertices from different clusters either not to be connected with an edge (i.e., $w_{ij} = 0$) to be connected with an edge with a weight w_{ij} as small as possible.
- Thus, the clusters should be, in a sense, "approximate connected components" of tightly connected vertices with weak edges between such "approximate components".
- We can take k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ corresponding to the k smallest eigenvalues (in place of eigenvalue 0 of multiplicity k) and form the corresponding matrix E with these eigenvectors as columns.
- We again consider the row vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$. If the graph G had k connected components, then for every two points v_i and v_j from the same components we saw that the corresponding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ would be identical.
- For clustering this is no longer the case, but since within each optimal cluster the weights of edges are high and the weights of edges between different clusters are low, we can hope that the points v_i and v_j from the same optimal cluster will have similar vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ and points v_i and v_j from different optimal clusters will have substantially different vectors $\mathbf{y}_i$ and $\mathbf{y}_j$.

Spectral Clustering

- When we cluster points into disjoint clusters we want any two vertices v_i, v_j with high similarity (i.e., with high weight w_{ij} of the corresponding edge (v_i, v_j)) to be in the same cluster, and any two vertices from different clusters either not to be connected with an edge (i.e., $w_{ij} = 0$) to be connected with an edge with a weight w_{ij} as small as possible.
- Thus, the clusters should be, in a sense, "approximate connected components" of tightly connected vertices with weak edges between such "approximate components".
- We can take k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ corresponding to the k smallest eigenvalues (in place of eigenvalue 0 of multiplicity k) and form the corresponding matrix E with these eigenvectors as columns.
- We again consider the row vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$. If the graph G had k connected components, then for every two points v_i and v_j from the same components we saw that the corresponding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ would be identical.
- For clustering this is no longer the case, but since within each optimal cluster the weights of edges are high and the weights of edges between different clusters are low, we can hope that the points v_i and v_j from the same optimal cluster will have similar vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ and points v_i and v_j from different optimal clusters will have substantially different vectors $\mathbf{y}_i$ and $\mathbf{y}_j$.

Spectral Clustering

- When we cluster points into disjoint clusters we want any two vertices v_i, v_j with high similarity (i.e., with high weight w_{ij} of the corresponding edge (v_i, v_j)) to be in the same cluster, and any two vertices from different clusters either not to be connected with an edge (i.e., $w_{ij} = 0$) to be connected with an edge with a weight w_{ij} as small as possible.
- Thus, the clusters should be, in a sense, "approximate connected components" of tightly connected vertices with weak edges between such "approximate components".
- We can take k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ corresponding to the k smallest eigenvalues (in place of eigenvalue 0 of multiplicity k) and form the corresponding matrix E with these eigenvectors as columns.
- We again consider the row vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$. If the graph G had k connected components, then for every two points v_i and v_j from the same components we saw that the corresponding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ would be identical.
- For clustering this is no longer the case, but since within each optimal cluster the weights of edges are high and the weights of edges between different clusters are low, we can hope that the points v_i and v_j from the same optimal cluster will have similar vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ and points v_i and v_j from different optimal clusters will have substantially different vectors $\mathbf{y}_i$ and $\mathbf{y}_j$.

Spectral Clustering

- When we cluster points into disjoint clusters we want any two vertices v_i, v_j with high similarity (i.e., with high weight w_{ij} of the corresponding edge (v_i, v_j)) to be in the same cluster, and any two vertices from different clusters either not to be connected with an edge (i.e., $w_{ij} = 0$) to be connected with an edge with a weight w_{ij} as small as possible.
- Thus, the clusters should be, in a sense, "approximate connected components" of tightly connected vertices with weak edges between such "approximate components".
- We can take k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ corresponding to the k smallest eigenvalues (in place of eigenvalue 0 of multiplicity k) and form the corresponding matrix E with these eigenvectors as columns.
- We again consider the row vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$. If the graph G had k connected components, then for every two points v_i and v_j from the same components we saw that the corresponding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ would be identical.
- For clustering this is no longer the case, but since within each optimal cluster the weights of edges are high and the weights of edges between different clusters are low, we can hope that the points v_i and v_j from the same optimal cluster will have similar vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ and points v_i and v_j from different optimal clusters will have substantially different vectors $\mathbf{y}_i$ and $\mathbf{y}_j$.

Spectral Clustering

- When we cluster points into disjoint clusters we want any two vertices v_i, v_j with high similarity (i.e., with high weight w_{ij} of the corresponding edge (v_i, v_j)) to be in the same cluster, and any two vertices from different clusters either not to be connected with an edge (i.e., $w_{ij} = 0$) to be connected with an edge with a weight w_{ij} as small as possible.
- Thus, the clusters should be, in a sense, "approximate connected components" of tightly connected vertices with weak edges between such "approximate components".
- We can take k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ corresponding to the k smallest eigenvalues (in place of eigenvalue 0 of multiplicity k) and form the corresponding matrix E with these eigenvectors as columns.
- We again consider the row vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$. If the graph G had k connected components, then for every two points v_i and v_j from the same components we saw that the corresponding vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ would be identical.
- For clustering this is no longer the case, but since within each optimal cluster the weights of edges are high and the weights of edges between different clusters are low, we can hope that the points v_i and v_j from the same optimal cluster will have similar vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ and points v_i and v_j from different optimal clusters will have substantially different vectors $\mathbf{y}_i$ and $\mathbf{y}_j$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \leq i, j \leq n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \ldots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \ldots, C_k$.

  Output: Clusters $A_1, \ldots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \leq i, j \leq n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \ldots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \ldots, C_k$.

  Output: Clusters $A_1, \ldots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \leq i, j \leq n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \ldots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \ldots, C_k$.

  Output: Clusters $A_1, \ldots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \leq i, j \leq n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \ldots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \ldots, C_k$.

  Output: Clusters $A_1, \ldots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \leq i, j \leq n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \dots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \dots, C_k$.

  Output: Clusters $A_1, \dots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \leq i, j \leq n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \dots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \dots, C_k$.

  Output: Clusters $A_1, \dots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \le i, j \le n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \dots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \dots, C_k$.

  Output: Clusters $A_1, \dots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \leq i, j \leq n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \dots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \dots, C_k$.

  Output: Clusters $A_1, \dots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering

- The previous heuristic analysis suggests the following clustering algorithm which can produce clusters of similar points that are not centre based.
- **Spectral Clustering Algorithm**.

  Input: a set of data points $A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$, number k of clusters to construct.

  1. Construct a similarity graph G by one of the ways described; let $W = \{w_{ij} : 1 \leq i, j \leq n\}$ be its weighted adjacency matrix.
  2. Compute the Laplacian $L = D - W$.
  3. Compute the k eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ of L which correspond to k smallest eigenvalues.
  4. Let E be the matrix of size $n \times k$ containing the eigenvectors $\mathbf{e}_1, \dots, \mathbf{e}_k$ as columns.
  5. For $i = 1, \dots, n$, let $\mathbf{y}_i$ be the vector corresponding to the i^{th} row of E.
  6. Cluster points $\{\mathbf{y}_1, \dots, \mathbf{y}_n\}$ using the k-means algorithm into clusters $C_1, \dots, C_k$.

  Output: Clusters $A_1, \dots, A_k$ with A_i defined as $A_i = \{v_j : \mathbf{y}_j \in C_i\}$.

Spectral Clustering as graph partitioning

- For a given number k of subsets, the MinCut approach to graph partitioning simply consists in choosing a partition $A_1, \ldots, A_k$ which minimises

$$\mathrm{cut}(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{n} W(A_i, \overline{A_i}) \tag{1}$$

- (Note that if $k = 2$ this is just the standard MinCut problem like we had fot Karger's algorithm).
- Recall that for any two sets A and B, we have $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$, and that $\overline{A}$ denotes the complement of set A.
- Note that the factor $\frac{1}{2}$ in (2) is present because every edge is counted twice.
- Unfortunately, minimising (2) is not a good idea.
- The reason is that minimising (2) often produces clusters some of which contain only a single vertex or just a few vertices.

Spectral Clustering as graph partitioning

- A better idea is to find a partition $A_1, \ldots, A_k$ which minimises

$$\mathrm{RatioCut}(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{n} \frac{W(A_i, \overline{A_i})}{|A_i|} \tag{2}$$

- Having $|A_i|'s$ in the denominator encourages the algorithm to find clusters A_i which all have reasonably large number of points, rather than just a few.
- For every partition $A_1, \ldots, A_k$ we can define the corresponding set of orthonormal indicator vectors $\mathbf{h}_j = (h_{1j}, h_{2j}, \ldots, h_{nj})^\mathsf{T}$ by setting $\mathbf{h}_j = 1/|A_j| \cdot \mathbb{1}_{A_j}$, i.e., by letting

$$h_{ij} = \begin{cases} \frac{1}{\sqrt{|A_j|}} & \text{if} \quad v_i \in A_j \\ 0 & \text{if} \quad v_i \notin A_j \end{cases}$$

- Note that

$$\|\mathbf{h}_j\|^2 = \sum_{v_i \in A_j} \left(\frac{1}{\sqrt{|A_j|}} \right)^2 = |A_j| \frac{1}{|A_j|} = 1$$

and, since $A_j's$ are pairwise disjoint, $\mathbf{h}_m \cdot \mathbf{h}_l = 0$ if $m \neq l$.

Spectral Clustering as graph partitioning

- Also, note that we have shown that

$$\begin{aligned}
\mathbf{h}_j^\mathsf{T} L \mathbf{h}_j &= \frac{1}{2}\sum_{m,l=1}^{n} w_{ml}(h_{mj}-h_{lj})^2 \\
&= \frac{1}{2}\left(\sum_{m\in A_j;\, l\notin A_j} w_{ml}(h_{mj}-h_{lj})^2 + \sum_{l\in A_j;\, m\notin A_j} w_{ml}(h_{mj}-h_{lj})^2\right) \\
&= \frac{1}{2}\left(\sum_{m\in A_j;\, l\notin A_j} w_{ml}\frac{1}{|A_j|} + \sum_{l\in A_j;\, m\notin A_j} w_{ml}\frac{1}{|A_j|}\right) \\
&= \frac{\mathrm{cut}(A_j,\overline{A_j})}{|A_j|}
\end{aligned}$$

- Let H be the matrix of size $n \times k$ with vectors $\mathbf{h}_j$, $1 \le j \le k$ as columns.
- Then the above equality implies that the sum of the diagonal elements of $H^\mathsf{T} L H$, i.e. the trace of $H^\mathsf{T} L H$ satisfies

$$\mathrm{Tr}(H^\mathsf{T} L H) = \sum_{j=1}^{k} \frac{\mathrm{cut}(A_j,\overline{A_j})}{|A_j|}$$

Spectral Clustering as graph partitioning

- Since we have proved that

$$\mathrm{RatioCut}(A_1, \dots A_k) = \sum_{j=1}^{k} \frac{\mathrm{cut}(A_j, \overline{A_j})}{|A_j|} = \mathrm{Tr}(H^\intercal L H)$$

  we can conclude that to minimise $\mathrm{RatioCut}(A_1, \dots A_k)$ we have to find disjoint sets $A_1, \dots A_k$ which minimise $\mathrm{Tr}(H^\intercal L H)$ where the columns $\mathbf{h}_j$ of H are of the form $\mathbf{h}_j = 1/|A_j| \cdot \mathbb{1}_{A_j}$.
- This is an NP hard problem, so we find only an approximate solution by solving the following relaxation of it:

$$\begin{aligned} &\text{Find} \quad H \in \mathbb{R}^{n \times k} \quad \text{which minimises} \quad \mathrm{Tr}(H^\intercal L H) \\ &\qquad\qquad\qquad\qquad\quad \text{subject to the constraint} \quad H^\intercal H = I \end{aligned}$$

- The Rayleigh-Ritz theorem from linear algebra tells us that such an $H \in \mathbb{R}^{n \times k}$ is obtained as the matrix with k eigenvectors corresponding to the k smallest eigenvalues of L as the k columns of H.

Spectral Clustering as graph partitioning

- Just as before, these eigenvectors might be approximations of linear combinations of the indicator functions $\mathbb{1}_{A_j}$, so we again cluster the n rows of H into k clusters $C_1, \dots C_k$ to obtain the partition $A_1, \dots A_k$ defined by

$$v_j \in A_m \quad \text{if an only if the } j^{th} \text{ row of } H \text{ belongs to } C_m$$

- But notice that this is precisely what our spectral clustering algorithm does.
- Thus, besides the original heuristics with the connected components, we now see that the spectral clustering algorithm finds an approximate solution to the problem of finding $A_1, \dots, A_k$ which minimise the ratio cut

$$\text{RatioCut}(A_1, \dots A_k) = \sum_{j=1}^{k} \frac{\text{cut}(A_j, \overline{A_j})}{|A_j|}$$

Spectral Clustering as graph partitioning

- It is possible to normalise the Laplacian so that the solution approximately minimises the so called Ncut defined as

$$\mathrm{Ncut}(A_1, \dots A_k) = \sum_{j=1}^{k} \frac{\mathrm{cut}(A_j, \overline{A_j})}{\mathrm{vol}(A_j)}$$

  where $\mathrm{vol}(A)$ is the sum of the degrees of all vertices in A:

$$\mathrm{vol}(A) = \sum_{v_i \in A} d_i = \sum_{v_i \in A} \left(\sum_{j=1}^{n} w_{ij} \right)$$

- This sometimes produces better clustering, which also has a nice interpretation via random walk on graphs (a random walk seldom switches between different clusters).
- You can find all the details in a wonderfully written tutorial by Ulrike von Luxburg from the Max Planck Institute for Biological Cybernetics, available at `http://www.tml.cs.uni-tuebingen.de/team/luxburg/publications/Luxburg07_tutorial.pdf`, which we have followed closely in a part of our presentation.