Week 10

Data Mining and Knowledge Discovery

Dr John Evans
j.evans8@herts.ac.uk

University of Hertfordshire UH

Plan for today

K-means Clustering

SSE

Problems with K-means

Recap

- We saw the idea of ensemble methods which combine several base classifiers and then use each classifier's prediction to make a final prediction (often by majority voting).
- We saw methods for constructing ensembles:
  - Manipulate the training set;
  - Manipulate the input features;
  - Manipulate the class labels;
  - Manipulate the learning algorithm.
- We saw the trade-off between bias and variance and the effects of underfitting and overfitting on generalisation performance.
- We saw bagging which repeatedly samples (with replacement) a data set according to a uniform probability distribution in such a way that each bootstrap sample has the same size as the original data.
- We saw random forests whose base classifiers are decision trees. They manipulate training instances and the input attributes.

Partition-based clustering

- The simplest and most fundamental type of cluster analysis is partitioning.
- This organises the objects into several exclusive clusters, the number of which is usually assumed to be given.

Partition-based clustering

- The simplest and most fundamental type of cluster analysis is partitioning.
- This organises the objects into several exclusive clusters, the number of which is usually assumed to be given.
- In general, we have a data set D with n objects for which we wish to partition into k (where $k \leq n$) clusters.
- The partitioning algorithm then organises these n objects into the k partitions and, in principle, does so in a way which optimises the objective partitioning criterion.
- This could be a dissimilarity function based on distance, say.

Partition-based clustering

- The simplest and most fundamental type of cluster analysis is partitioning.
- This organises the objects into several exclusive clusters, the number of which is usually assumed to be given.
- In general, we have a data set D with n objects for which we wish to partition into k (where $k \leq n$) clusters.
- The partitioning algorithm then organises these n objects into the k partitions and, in principle, does so in a way which optimises the objective partitioning criterion.
- This could be a dissimilarity function based on distance, say.
- In this way, we have k clusters $C_1, \ldots, C_k$, each of which are contained in D and for which $C_i \cap C_j = \emptyset$ when $i \neq j$. If our objective function is well-chosen, we have high intracluster similarity and low intercluster similarity.

Types of partition-based clustering

- There are several algorithms we can use. Two of the most prominent are K-means and K-medoid.

Types of partition-based clustering

- There are several algorithms we can use. Two of the most prominent are K-means and K-medoid.
- Both are prototype-based clustering algorithms with the former defining a prototype in terms of a centroid. This is usually the mean of a group of points and is typically applied to objects in a continuous n-dimensional space.

Types of partition-based clustering

- There are several algorithms we can use. Two of the most prominent are K-means and K-medoid.
- Both are prototype-based clustering algorithms with the former defining a prototype in terms of a centroid. This is usually the mean of a group of points and is typically applied to objects in a continuous n-dimensional space.
- In contrast, K-medoid clustering defined a prototype in terms of a medoid, which is the most representative point for a group of points. This can be applied to a wide range of data since it requires only a proximity measure for a pair of points.
- While a centroid almost never corresponds to an actual data point, a medoid, by its definition, must always be an actual data point.

The K-means method

Our focus will be the K-means algorithm, whose basic algorithm is as follows:

The K-means method

Our focus will be the K-means algorithm, whose basic algorithm is as follows:

1. First, we choose k initial centroids, where k is our desired number of clusters.

The K-means method

Our focus will be the K-means algorithm, whose basic algorithm is as follows:

1. First, we choose k initial centroids, where k is our desired number of clusters.
2. Each point is then assigned to the closest centroid, and each collection of points assigned to a centroid is a cluster.

The K-means method

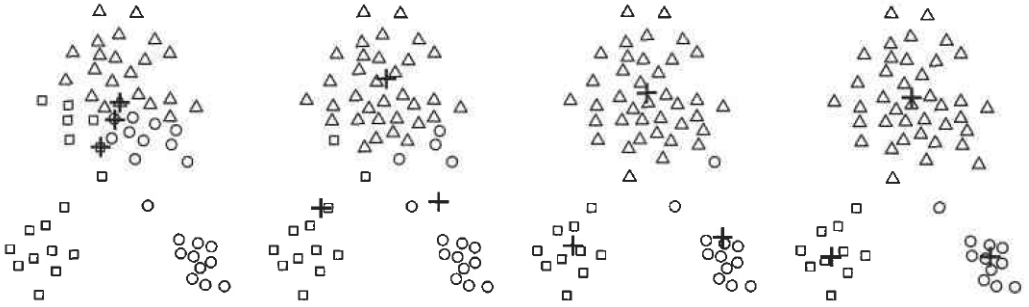
Our focus will be the K-means algorithm, whose basic algorithm is as follows:

1. First, we choose k initial centroids, where k is our desired number of clusters.
2. Each point is then assigned to the closest centroid, and each collection of points assigned to a centroid is a cluster.
3. The centroid of each cluster is then updated based on the points assigned to the cluster.
4. We repeat the assignment and update steps until no point changes clusters, or equivalently, until the centroids remain the same.

Basic K-means algorithm

Input:	k: the number of clusters, D: a data set containing n objects
Output:	A set of k clusters
Method:	
Step 1	Arbitrarily choose k objects from D as the initial cluster centres;
Step 2	**repeat**
Step 3	(re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
Step 4	update the cluster means; that is, calculate the mean value of the objects for each cluster;
Step 5	**until** no change;

Example



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Example (Explanation)

- We start with three centroids and proceed in four steps.
- Note that the centroids are indicated by '$+$', and all points belonging to the same cluster have the same marker shape.
- In the first step, points are assigned to the initial centroids, which are all in the larger group of points. After the points are assigned to the centroid, the centroid is then updated.
- In the second step, the points are assigned to the updated centroids before the centroids are updated again.
- We note that in Steps 2, 3 and 4, two of the centroids move to the two small groups of points at the bottom of the figures.
- When the k-means algorithm terminates in Step 5 (because no changes occur), the centroids have identified the natural groupings of points.

K-means 'always' settles down

- For a number of combinations of proximity functions and types of centroids, k-means always converges to a solution (i.e. K-means reaches a state in which no points are shifting from one cluster to another, and hence the centroids don't change).
- However, because most of the convergence occurs in the early steps, we often replace the condition in Step 5 of the basic algorithm by a weaker condition, e.g. repeat until only 1% of the points change clusters.

More detail (Step 3)

Step 3: Assign each object to the nearest cluster.

More detail (Step 3)

Step 3: Assign each object to the nearest cluster.

- First, we need a proximity measure that quantifies the notion of ‘closest’ for the specific data under consideration.
- The typical choice when working in Euclidean space is the Euclidean L_2 distance.
- For documents we often use the cosine similarity.

More detail (Step 3)

Step 3: Assign each object to the nearest cluster.

- First, we need a proximity measure that quantifies the notion of ‘closest’ for the specific data under consideration.
- The typical choice when working in Euclidean space is the Euclidean L_2 distance.
- For documents we often use the cosine similarity.
- Of course, we can also use any of the other proximity measures discussed in earlier in the module, such as the Manhattan L_1 distance or Jaccard similarity measure. The former would be used for Euclidean data, while the latter for documents.

More detail (Step 3 continued)

- As the algorithm repeatedly calculates similarity, it is typical to use relatively simple similarity measures.

More detail (Step 3 continued)

- As the algorithm repeatedly calculates similarity, it is typical to use relatively simple similarity measures.
- However, in some cases, such as in low-dimensional Euclidean space, it is possible to avoid computing many of the similarities, thereby significantly speeding up the K-means algorithm.
- Another approach to speed up the algorithm is the bisecting K-means strategy which we will discuss next week. This speeds things up by reducing the number of similarities computed.

More detail (Step 4)

Step 4: Update the cluster means.

- Note that the centroid can vary depending on the proximity measure for the data and the goal of the clustering.

More detail (Step 4)

Step 4: Update the cluster means.

- Note that the centroid can vary depending on the proximity measure for the data and the goal of the clustering.
- The goal of the clustering is typically expressed by an objective function which depends on the proximities of the points to one another, or to the cluster centroids.
- For instance, we may wish to minimise the squared distance of each point to its closest centroid.

More detail (Step 4)

Step 4: Update the cluster means.

- Note that the centroid can vary depending on the proximity measure for the data and the goal of the clustering.
- The goal of the clustering is typically expressed by an objective function which depends on the proximities of the points to one another, or to the cluster centroids.
- For instance, we may wish to minimise the squared distance of each point to its closest centroid.
- We will now demonstrate this with two examples, but in both the key point is that once we have specified a proximity measure and an objective function, the centroid that we should choose can be determined mathematically.

Example (Data in Euclidean space)

- Consider data whose proximity measure is Euclidean distance.

Example (Data in Euclidean space)

- Consider data whose proximity measure is Euclidean distance.
- For our objective function, which measures the quality of a clustering, we use the *sum of the squared error (SSE)*.
- Here, we calculate the error of each data point (its Euclidean distance to the closest centroid) and then compute the total sum of the squared errors.

Example (Data in Euclidean space)

- Consider data whose proximity measure is Euclidean distance.
- For our objective function, which measures the quality of a clustering, we use the *sum of the squared error (SSE)*.
- Here, we calculate the error of each data point (its Euclidean distance to the closest centroid) and then compute the total sum of the squared errors.
- Given two different sets of clusters that are produced by two different runs of K-means, we choose the one with smallest squared error. This results in the prototypes (centroids) of this clustering being a better representation of the points in their cluster. In this way, clustering becomes an optimisation problem.

Example (Data in Euclidean space continued)

We now make the following notational choices:

- x, an object
- C_i, the i^{th} cluster,
- c_i, the centroid of cluster C_i,
- m_i, the number of objects in the i^{th} cluster,
- m, the number of objects in the data set,
- k, the number of clusters.

Example (Data in Euclidean space continued)

We now make the following notational choices:

- x, an object
- C_i, the i^{th} cluster,
- c_i, the centroid of cluster C_i,
- m_i, the number of objects in the i^{th} cluster,
- m, the number of objects in the data set,
- k, the number of clusters.

Given this notation, the SSE is formally defined as follows:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist(c_i, x)^2,$$

where $dist$ is the standard Euclidean L_2 distance between two objects.

Example (Data in Euclidean space continued)

Goal: Minimise the total SSE.[1]

[1] Approaches such as gradient descent are often used for this.

Example (Data in Euclidean space continued)

Goal: Minimise the total SSE.[1]

- To keep things simple, we consider the case when the proximity function is Euclidean distance and the objective is to minimise the SSE in the case of one-dimensional data.
- In other words, we want to minimise,

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} (c_i - x)^2,$$

where C_i is the i^{th} cluster, x is a point in C_i and c_i is the mean of the i^{th} cluster.

[1] Approaches such as gradient descent are often used for this.

Example (Data in Euclidean space continued)

Goal: Minimise the total SSE.[1]

- To keep things simple, we consider the case when the proximity function is Euclidean distance and the objective is to minimise the SSE in the case of one-dimensional data.
- In other words, we want to minimise,

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} (c_i - x)^2,$$

  where C_i is the i^{th} cluster, x is a point in C_i and c_i is the mean of the i^{th} cluster.
- We solve for the j^{th} centroid c_j, which minimises this equation, by differentiating the SSE and equating to 0.

[1] Approaches such as gradient descent are often used for this.

Example (Data in Euclidean space continued)

- First,

$$\frac{\partial}{\partial c_j} SSE = \quad \frac{\partial}{\partial c_j} \sum_{i=1}^{k} \sum_{x \in C_i} (c_i - x)^2$$

Example (Data in Euclidean space continued)

- First,

$$\begin{aligned}\frac{\partial}{\partial c_j} SSE = & \frac{\partial}{\partial c_j} \sum_{i=1}^{k} \sum_{x \in C_i} (c_i - x)^2 \\ = & \sum_{i=1}^{k} \sum_{x \in C_i} \frac{\partial}{\partial c_j} (c_i - x)^2\end{aligned}$$

Example (Data in Euclidean space continued)

- First,

$$\begin{aligned}
\frac{\partial}{\partial c_j} SSE &= \frac{\partial}{\partial c_j} \sum_{i=1}^{k} \sum_{x \in C_i} (c_i - x)^2 \\
&= \sum_{i=1}^{k} \sum_{x \in C_i} \frac{\partial}{\partial c_j} (c_i - x)^2 \\
&= \sum_{x \in C_j} \frac{\partial}{\partial c_j} (c_j - x)^2 \\
&= \sum_{x \in C_j} 2(c_j - x) \qquad = 0.
\end{aligned}$$

Example (Data in Euclidean space continued)

- First,

$$\begin{aligned}\frac{\partial}{\partial c_j} SSE &= \tfrac{\partial}{\partial c_j} \textstyle\sum_{i=1}^{k} \sum_{x \in C_i} (c_i - x)^2 \\ &= \textstyle\sum_{i=1}^{k} \sum_{x \in C_i} \tfrac{\partial}{\partial c_j} (c_i - x)^2 \\ &= \textstyle\sum_{x \in C_j} \tfrac{\partial}{\partial c_j} (c_j - x)^2 \\ &= \textstyle\sum_{x \in C_j} 2(c_j - x) \qquad = 0.\end{aligned}$$

- We want to solve this for c_j and recall that we said m_j is the number of objects in the j^{th} cluster (i.e. we are summing m_j times). Rearranging therefore gives,

$$m_j c_j = \sum_{x \in C_k} x \;\; \Rightarrow \;\; c_j = \frac{1}{m_j} \sum_{x \in C_j} x.$$

Example (Data in Euclidean space continued)

- In other words, the best centroid for minimising the SSE of a cluster is the mean of the points in the cluster.
- More generally, using our notation above, the centroid of the i^{th} cluster is given by,

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}.$$

Example (Data in Euclidean space continued)

- In other words, the best centroid for minimising the SSE of a cluster is the mean of the points in the cluster.
- More generally, using our notation above, the centroid of the i^{th} cluster is given by,

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}.$$

- To illustrate this, suppose our cluster has three points:

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 2 \\ -3 \end{pmatrix}, \begin{pmatrix} 3 \\ -5 \end{pmatrix}.$$

Then, $c_i = \frac{1}{3}\left(\begin{pmatrix} 1 \\ -1 \end{pmatrix} + \begin{pmatrix} 2 \\ -3 \end{pmatrix} + \begin{pmatrix} 3 \\ -5 \end{pmatrix}\right) = \frac{1}{3}\begin{pmatrix} 1+2+3 \\ (-1)+(-3)+(-5) \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \end{pmatrix}$

Example (Data in Euclidean space concluded)

- Steps 3 and 4 of the K-means algorithm directly attempt to minimise the objective function (in our case, the SSE).
- Step 3 then forms clusters by assigning points to their nearest centroid, which minimises the SSE for the given set of centroids.

Example (Data in Euclidean space concluded)

- Steps 3 and 4 of the K-means algorithm directly attempt to minimise the objective function (in our case, the SSE).
- Step 3 then forms clusters by assigning points to their nearest centroid, which minimises the SSE for the given set of centroids.
- Finally, Step 4 recomputes the centroids so as to further minimise the SSE.
- However, the actions of K-means in Steps 3 and 4 are guaranteed to only find a local minimum with respect to the SSE because they are based on optimising the SSE for specific choices of the centroids and clusters, rather than for all possible choices.

Example (Document data)

- The K-means algorithm is in no means restricted only to data in Euclidean space.

Example (Document data)

- The K-means algorithm is in no means restricted only to data in Euclidean space.
- For example, if we have document data then we can use the cosine similarity measure.
- We assume that the document data is represented as a document-term matrix.
- Our objective is to maximise the similarity of the documents in a cluster to the cluster centroid.

Example (Document data)

- The K-means algorithm is in no means restricted only to data in Euclidean space.
- For example, if we have document data then we can use the cosine similarity measure.
- We assume that the document data is represented as a document-term matrix.
- Our objective is to maximise the similarity of the documents in a cluster to the cluster centroid.
- This quantity is known as the *cohesion* of the cluster and, as with the Euclidean data, the cluster centroid is the mean.
- The analogous quantity to the total SSE is the total cohesion, given by,

$$Total\ Cohesion = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} cosine(\mathbf{x},\ \mathbf{c}_i).$$

Other choices for proxmity

Common choices for proximity, centroids and objective functions		
Proximity Function	Centroid	Objective Function
Manhattan (L_1)	Median	Minimise sum of the L_1 distance of an object to its cluster centroid
Squared Euclidean (L_2^2)	Mean	Minimise sum of the squared L_2 distance of an object to its cluster centroid
Cosine Similarity	Mean	Maximise sum of the cosine similarity of an object to its cluster centroid
Bregman Divergence	Mean	Minimise sum of the Bregman divergence of an object to its cluster centroid

More detail (Step 1)

Moving on, we will focus simply on two-dimensional data. This is for simplicity and does not suggest any greater importance of this case.

Step 1: Randomly choose starting centroids.

More detail (Step 1)

Moving on, we will focus simply on two-dimensional data. This is for simplicity and does not suggest any greater importance of this case.

Step 1: Randomly choose starting centroids.

- When random initialisation of centroids is used, different runs of K-means typically produce different total SSEs.
- As such, the K-means algorithm is not guaranteed to converge to the global optimum, and often terminates at a local optimum.

More detail (Step 1)

Moving on, we will focus simply on two-dimensional data. This is for simplicity and does not suggest any greater importance of this case.

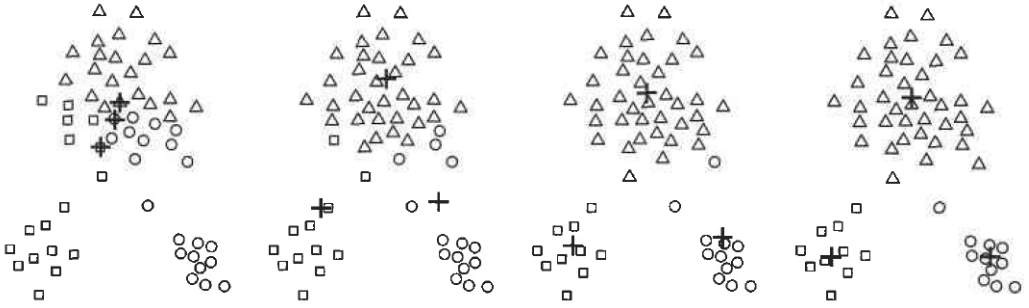
Step 1: Randomly choose starting centroids.

- When random initialisation of centroids is used, different runs of K-means typically produce different total SSEs.
- As such, the K-means algorithm is not guaranteed to converge to the global optimum, and often terminates at a local optimum.
- Such results inevitably depend on the initial random selection and so it follows that choosing the proper initial centroids is the key step of the basic K-means procedure.
- While common approach is to choose the initial centroids randomly, the resulting clusters are often poor.

Example (Poor initial centroids)

- Using the same data set seen earlier, we can demonstrate how randomly selected initial centroids can be poor.
- Contrast the figures on the following two slides.
- Both show clusters that result from two particular choices of initial centroids.
- In the first figure, even though all the initial centroids are from one natural cluster, the minimum SSE clustering is still found.
- In the second, however, even though the initial centroids appear better distributed, we obtain a suboptimal clustering with higher squared error.

Example (Poor initial centroids - Figure 1)



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Example (Poor initial centroids - Figure 2)

(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

A solution

- One approach to overcoming the previous example's problems is to perform multiple runs of the algorithm, each time with a different set of randomly chosen initial centroids.
- Then, we select the set of clusters with the minimum SSE.

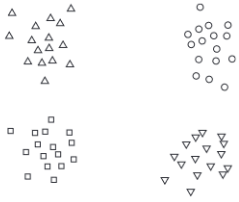
A solution

- One approach to overcoming the previous example's problems is to perform multiple runs of the algorithm, each time with a different set of randomly chosen initial centroids.
- Then, we select the set of clusters with the minimum SSE.
- While simple, this strategy can struggle depending on the data set and/or the number of clusters sought.

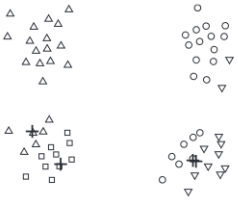
Example (Multiple run throughs)

- Consider the data set shown in Figure (a) of the following slide.
- This consists of two pairs of clusters. The clusters in each (top-bottom) pair are closer to each other than to the clusters in the other pair.
- Figures (b)-(d) then show that if we start with two initial centroids per pair of clusters, then even when both centroids are in a single cluster, the centroids will redistribute themselves so that the ‘true’ clusters are found.

Example (Multiple run throughs)



(a) Initial points.



(b) Iteration 1.

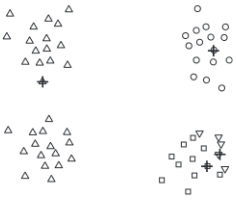
(c) Iteration 2.

(d) Iteration 3.

Example (Multiple run throughs)

- However, if we next look at figure on the following slide, then we see that if a pair of clusters has only one initial centroid and the other pair has three, then two of the true clusters will be combined and one true cluster will be split.

Example (Multiple run throughs)



(a) Iteration 1.

(b) Iteration 2.

(c) Iteration 3.

(d) Iteration 4.

Example Multiple run throughs concluded)

- Note that an optimal clustering will only be obtained as long as two initial centroids fall anywhere in a pair of clusters, since the centroids will redistribute themselves, one to each cluster.
- Unfortunately, as the number of clusters becomes larger, it is increasingly likely that at least one pair of clusters will have only one initial centroid.

Techniques for initialisation

Due to the problems highlighted above, other techniques are often employed for initialisation. We now present two approaches that can be effective.

Techniques for initialisation

Due to the problems highlighted above, other techniques are often employed for initialisation. We now present two approaches that can be effective.

1. Take a sample of points and cluster them using a hierarchical clustering technique. Next, k clusters are extracted from the hierarchical clustering and the centroids of those clusters are used as the initial centroids. This approach often works well but is practical only if
   - **1.1** the sample is relatively small, e.g. a few hundred/thousand (hierarchical clustering is expensive), and
   - **1.2** k is relatively small compared to the sample size.

Techniques for initialisation

2. Select the first point at random or take the centroid of all points. Then, for each successive initial centroid, select the point that is farthest from any of the initial centroids already selected. In this way, we obtain a set of initial centroids that is guaranteed to be not only randomly selected, but also well separated.

Techniques for initialisation

2. Select the first point at random or take the centroid of all points. Then, for each successive initial centroid, select the point that is farthest from any of the initial centroids already selected. In this way, we obtain a set of initial centroids that is guaranteed to be not only randomly selected, but also well separated.

- Such an approach can select outliers, rather than points in dense regions (clusters).
- This can lead to a situation where many clusters have just one point (an outlier) which reduces the number of centroids for forming clusters for the majority of points.
- As might be expected, this is also expensive.

Techniques for initialisation

2. Select the first point at random or take the centroid of all points. Then, for each successive initial centroid, select the point that is farthest from any of the initial centroids already selected. In this way, we obtain a set of initial centroids that is guaranteed to be not only randomly selected, but also well separated.

- Such an approach can select outliers, rather than points in dense regions (clusters).
- This can lead to a situation where many clusters have just one point (an outlier) which reduces the number of centroids for forming clusters for the majority of points.
- As might be expected, this is also expensive.
- To overcome these problems, this approach is often applied to a sample of the points. Because outliers are comparatively rare, they tend to not show up in a random sample.

Additional issues with K-means (Handling empty clusters)

Exercise: Read pp. 123-124 to see a new variation of the K-means approach, known as the K-means$++$ method.

Additional issues with K-means (Handling empty clusters)

Exercise: Read pp. 123-124 to see a new variation of the K-means approach, known as the K-means$++$ method.

One potential problem occurs if no points are allocated to a cluster during the assignment step. This is possible and if/when it does, a strategy is needed to choose a replacement centroid. If not, the squared error will be larger than necessary.

Additional issues with K-means (Handling empty clusters)

Exercise: Read pp. 123-124 to see a new variation of the K-means approach, known as the K-means$++$ method.

One potential problem occurs if no points are allocated to a cluster during the assignment step. This is possible and if/when it does, a strategy is needed to choose a replacement centroid. If not, the squared error will be larger than necessary.

- One approach is to choose the point that is farthest away from any current centroid. At the very least, this eliminates the point that currently contributes most to the total squared error.
- Another approach is to use a K-means$++$ approach.
- A third approach is to choose the replacement centroid at random from the cluster that has the highest SSE.

Additional issues with K-means (Outliers)

- When the squared error criterion is used, outliers can unduly influence the clusters that are found.
- In particular, when outliers are present, the resulting cluster centroids are typically not as representative as they otherwise would be, thereby leading to higher SSE.
- Because of this, it is often useful to discover outliers and eliminate them beforehand.

Additional issues with K-means (Outliers)

- When the squared error criterion is used, outliers can unduly influence the clusters that are found.
- In particular, when outliers are present, the resulting cluster centroids are typically not as representative as they otherwise would be, thereby leading to higher SSE.
- Because of this, it is often useful to discover outliers and eliminate them beforehand.
- There are times when we do not wish to eliminate outliers (e.g. when using clustering for data compression, financial analysis etc.).
- In such cases, the issue becomes one of identifying outliers. There are several techniques used for this, but we will not discuss them here.

Additional issues with K-means (Reducing the SSE with postprocessing)

- An obvious way to reduce the SSE is to find more clusters.

Additional issues with K-means (Reducing the SSE with postprocessing)

- An obvious way to reduce the SSE is to find more clusters.
- In many cases we would like to improve the SSE *without* increasing k.
- As K-means typically converges to a local minimum, this is often possible.
- There are various techniques used and the usual strategy is to focus on individual clusters since the total SSE is simply the sum of the SSE contributed by each cluster (for this reason, the terms *total* SSE and *cluster* SSE are often used).

Additional issues with K-means (Reducing the SSE with postprocessing)

- An obvious way to reduce the SSE is to find more clusters.
- In many cases we would like to improve the SSE *without* increasing k.
- As K-means typically converges to a local minimum, this is often possible.
- There are various techniques used and the usual strategy is to focus on individual clusters since the total SSE is simply the sum of the SSE contributed by each cluster (for this reason, the terms *total* SSE and *cluster* SSE are often used).
- We can change the total SSE by performing various operations on the clusters, such as splitting or merging clusters.

Additional issues with K-means (Reducing the SSE with postprocessing)

- A commonly used approach is to employ alternate cluster splitting and merging phases. During a splitting phase, clusters are divided, while during a merging phase the are combined.
- In this way, it is often possible to escape local SSE minima and still produce a clustering solution with the desired number of clusters.

Additional issues with K-means (Reducing the SSE with postprocessing)

- A commonly used approach is to employ alternate cluster splitting and merging phases. During a splitting phase, clusters are divided, while during a merging phase the are combined.
- In this way, it is often possible to escape local SSE minima and still produce a clustering solution with the desired number of clusters.
- The following are some techniques used in the splitting and merging phases.
  1. Decrease the total SSE by increasing the number of clusters
     - Split a cluster
       The cluster with the largest SSE is usually chosen, but we could also split the cluster with the largest standard deviation for a particular attribute.
     - Introduce a new cluster centroid
       Often, the point that is farthest from any cluster centre is chosen. We can easily determine this if we keep track of the SSE contributed by each point. Alternatively, we could choose randomly from all points or from the points with the highest SSE with respect to their closest centroids.

Summary

- We have discussed the K-means clustering method
- We have seen how, in Euclidean space, we want to minimise the total SSE.
- We have seen how other proximity measures can be used for centroids, proximity and objective functions.
- We have seen problems with choosing initial centroids.
- We have seen pther issues with the K-means algorithm, such as:
  - Handling empty clusters.
  - Outliers.
  - Reducing the SSE with postprocessing.