Clustering and Visualisation of Data

Hiroshi Shimodaira*

January-March 2020

Cluster analysis aims to partition a data set into meaningful or useful groups, based on distances between data points. In some cases the aim of cluster analysis is to obtain greater understanding of the data, and it is hoped that the clusters capture the natural structure of the data. In other cases cluster analysis does not necessarily add to understanding of the data, but enables it to be processed more efficiently.

Cluster analysis can be contrasted with *classification*. Classification is a *supervised* learning process: there is a training set in which each data item has a label. For example, in handwriting recognition the training set may correspond to a set of images of handwritten digits, together with a label ('zero' to 'nine') for each digit. In the test set the label for each image is unknown: it is the job of the classifier to predict a label for each test item.

Clustering, on the other hand, is an *unsupervised* procedure in which the training set does not contain any labels. The aim of a clustering algorithm is to group such a data set into clusters, based on the unlabelled data alone. In many situations there is no 'true' set of clusters. For example consider the twenty data points shown in Figure 3.1 (a). It is reasonable to divide this set into two clusters (Figure 3.1 (b)), four clusters (Figure 3.1 (c)) or five clusters (Figure 3.1 (d)).

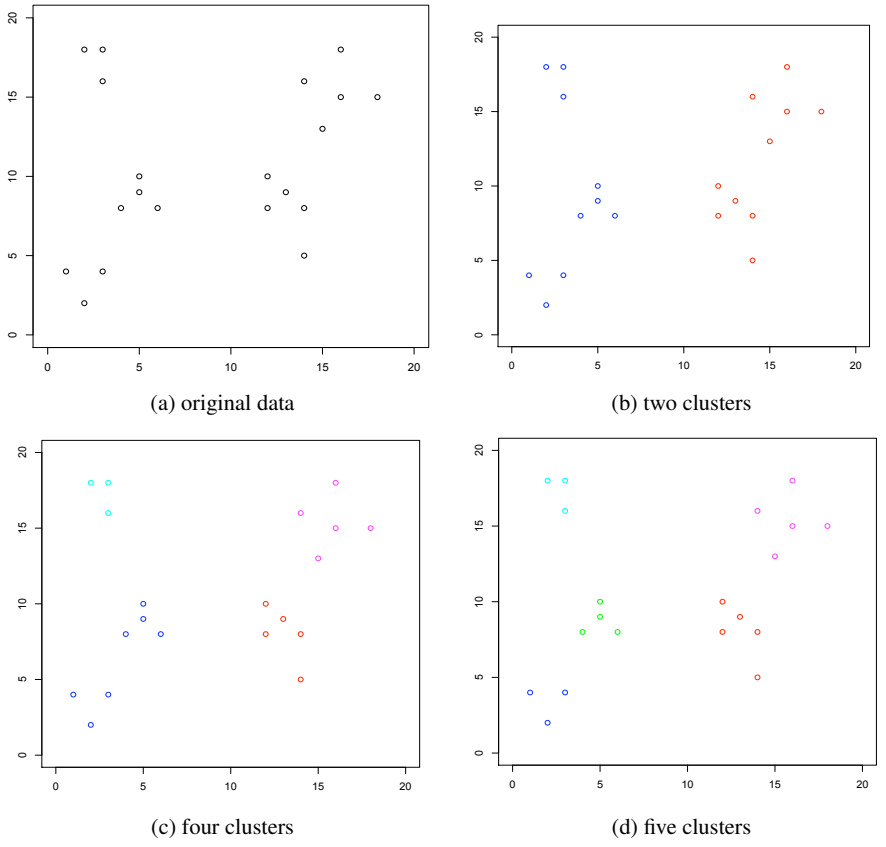
There are many reasons to perform clustering. Most commonly it is done to better understand the data (*data interpretation*), or to efficiently code the data set (*data compression*).

Data interpretation: Automatically dividing a set of data items into groups is an important way to analyse and describe the world. Automatic clustering has been used to cluster documents (such as web pages), user preference data (of the type discussed in the previous chapter), and many forms of scientific observational data in fields ranging from astronomy to psychology to biology.

Data compression: Clustering may be used to compress data by representing each data item in a cluster by a single cluster *prototype*, typically at the centre of the cluster. Consider D-dimensional data which has been clustered into K clusters. Rather than representing a data item as a D-dimensional vector, we could store just its cluster index (an integer from 1 to K). This representation, known as *vector quantisation*, reduces the required storage for a large data set at the cost of some information loss. Vector quantisation is used in image, video and audio compression.

3.1 Types of clustering

There are two main approaches to clustering: hierarchical and partitional. *Hierarchical clustering* forms a tree of nested clusters in which, at each level in the tree, a cluster is the union of its children.



(a) original data (b) two clusters

(c) four clusters (d) five clusters

Figure 3.1: Clustering a set of 20 two-dimensional data points.

Partitional clustering does not have a nested or hierarchical structure, it simply divides the data set into a fixed number of non-overlapping clusters, with each data point assigned to exactly one cluster. The most commonly employed partitional clustering algorithm, K-means clustering, is discussed below. Whatever approach to clustering is employed, the core operations are distance computations: computing the distance between two data points, between a data point and a cluster prototype, or between two cluster prototypes.

There are two main approaches to hierarchical clustering. In *top-down clustering* algorithms, all the data points are initially collected in a single top-level cluster. This cluster is then split into two (or more) sub-clusters, and each these sub-clusters is further split. The algorithms continues to build a tree structure in a top down fashion, until the leaves of the tree contain individual data points. An alternative approach is *agglomerative hierarchical clustering*, which acts in a bottom-up way. An agglomerative clustering algorithm starts with each data point defining a one-element cluster. Such an algorithm operates by repeatedly merging the two closest clusters until a single cluster is obtained.

3.2 K-means clustering

K-means clustering aims to divide a set of D-dimensional data points into K clusters. The number of clusters, K, must be specified, it is not determined by the clustering: thus it will always attempt to find K clusters in the data, whether they really exist or not.

Each cluster is defined by its cluster centre, and clustering proceeds by assigning each of the input data points to the cluster with the closest centre, using a Euclidean distance metric. The centre of each cluster is then re-estimated as the *centroid* of the points assigned to it. The process is then iterated. The algorithm is:

- *Initialise* K cluster centres, $\{\mathbf{m}_k\}_1^K$
- While not *converged*:
  - Assign each data vector $\mathbf{x}_n$ $(1 \le n \le N)$ to the closest cluster centre;
  - Recompute each cluster mean as the mean of the vectors assigned to that cluster

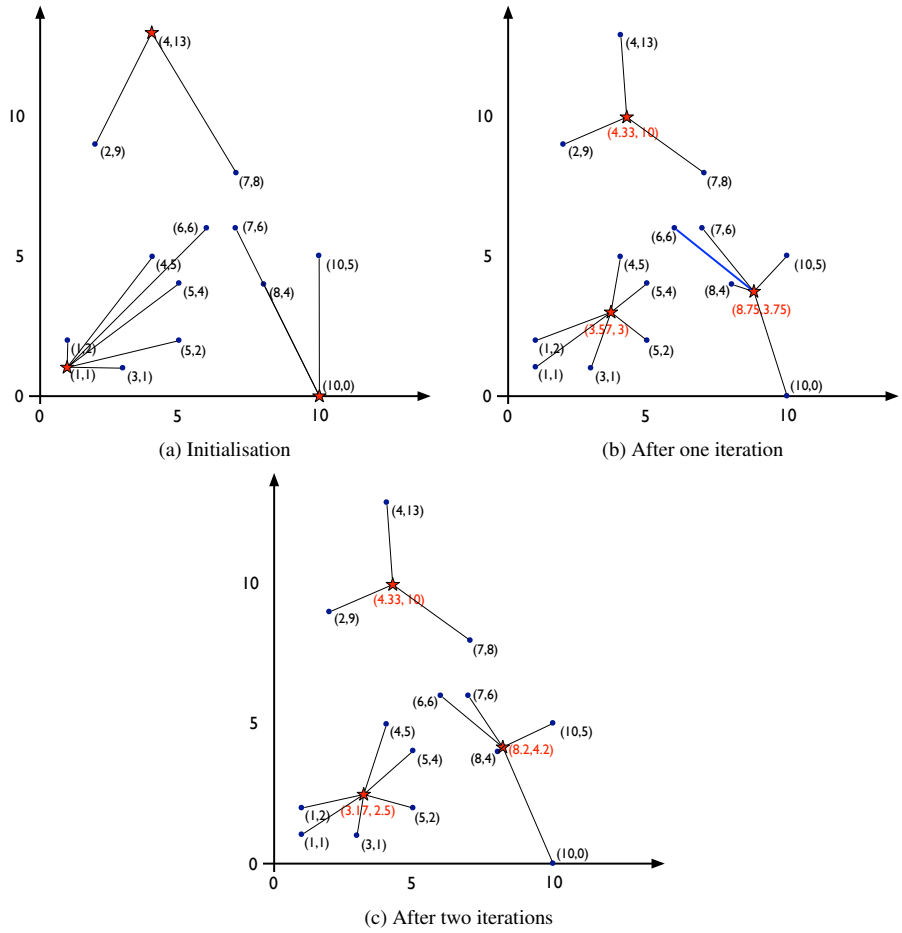
The algorithm requires a distance measure to be defined in the data space, and the Euclidean distance is often used.

The initialisation method needs to be further specified. There are several possible ways to initialise the cluster centres, including:

- Choose random data points as cluster centres
- Randomly assign data points to K clusters and compute means as initial centres
- Choose data points with extreme values
- Find the mean for the whole data set then perturb into K means

All of these work reasonably, and there is no 'best' way. However, as discussed below, the initialisation has an effect on the final clustering: different initialisations lead to different cluster solutions.

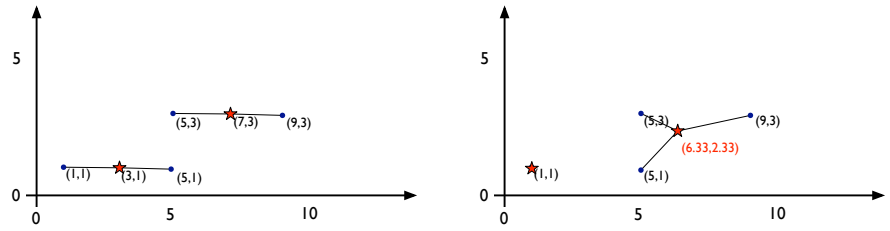
The algorithm iterates until it converges. Convergence is reached when the assignment of points to clusters does not change after an iteration. An attractive feature of K-means is that convergence is



(a) Initialisation (b) After one iteration

(c) After two iterations

Figure 3.2: Example of K-means algorithm applied to 14 data points, $K = 3$. The lines indicate the distances from each point to the centre of the cluster to which it is assigned. Here only one point (6,6) moves cluster after updating the means. In general, multiple points can be reassigned after each update of the centre positions.



(a) Within-cluster sum-squared error = 4 (b) Within-cluster sum-squared error = 3.33

Figure 3.3: Two different converged clusterings for the same data set, but starting from different initialisations.

guaranteed. However, the number of iterations required to reach convergence is not guaranteed. For large datasets it is often sensible to specify a maximum number of iterations, especially since a good clustering solution is often reached after a few iterations. Figure 3.2 illustrates the K-means clustering process. Figure 3.3 illustrates how different initialisations can lead to different solutions.

3.3 Mean squared error function

K-means clustering is an intuitively sensible algorithm, but is it possible to get a better mathematical description of what is going on? To compare two different clusterings into K clusters we can use the *mean squared error* function, which can be thought of as measuring the *scatter* of the data points relative to their cluster centres. Thus if we have two sets of K clusters, it makes sense to prefer the one with the smallest mean squared error: the clustering with the lowest scatter of data points relative to their cluster centres.

Let us define an indicator variable z_{nk}, such that $z_{nk} = 1$ if the n th data point $\mathbf{x}_n$ belongs to cluster k and $z_{nk} = 0$ otherwise. Then we can write the mean squared error as

$$E = \frac{1}{N} \sum_{k=1}^{K} \sum_{n=1}^{N} z_{nk} \|\mathbf{x}_n - \mathbf{m}_k\|^2, \tag{3.1}$$

where $\mathbf{m}_k$ is the centre of cluster k, N is the number of data points in total, and $\|\cdot\|$ denotes the Euclidean norm (i.e. L^2-norm) of a vector. Another way of viewing this error function is in terms of the squared deviation of the data points belonging to a cluster from the cluster centre, summed over all centres. This may be regarded as a variance measure for each cluster, and hence K-means is sometimes referred to as *minimum variance* clustering. Like all variance-based approaches this criterion is dependent on the scale of the data. In the introduction we said that the aim of clustering was to discover a structure of clusters such that points in the same group are close to each other, and far from points in other clusters. The mean squared error only addresses the first of these: it does not include a between-clusters term.

Depending on the initialisation of the cluster centres, K-means can converge to different solutions; this is illustrated in Figure 3.3. The same data set of 4 points, can have two different clusterings, depending on where the initial cluster centres are placed. Both these solutions are *local minima* of the error function, but they have different error values. For the solution in Figure 3.3a, the error is:

$$E = \frac{((4+4)+(4+4))}{4} = 4\,.$$

The second solution (Figure 3.3b) has a lower error:

$$E = \frac{0 + (32/9 + 20/9 + 68/9)}{4} = \frac{10}{3} < 4\,.$$

We have discussed the *batch* version of K-means. The online (sample-by-sample) version in which a point is assigned to a cluster and the cluster centre is immediately updated is less vulnerable to local minima.

There are many variants of K-means clustering that have been designed to improve the efficiency and to find lower error solutions.

3.4 Dimensionality reduction and data visualisation

Another way of obtaining better understanding of the data is to visualise it. For example, we could easily see the shapes of data clusters or spot outliers if they are plotted in a two or three dimensional space. It is, however, not straightforward to visualise high-dimensional data.

A simple solution would be to pick up only two components out of D ones and plot them in the same manner as we did for Figure 1 in Note 2, in which we picked up two films, 'Hancock' and 'Revolutionary Road' out of 6 films to plot critics. We would, however, need to draw more number of plots with different combinations of films to grasp the distribution of data.

A more general way of visualising high-dimensional data is to transform the data to the one in a two-dimensional space. For example, we can apply a linear transformation or mapping using a unit vector $\mathbf{u} = (u_1, \ldots, u_D)^T$ in the original D-dimensional vector space.[1] Calculating a dot-product (Euclidean inner-product) between $\mathbf{u}$ and $\mathbf{x}_n$ gives a scalar y_n:

$$y_n = \mathbf{u} \cdot \mathbf{x}_n = \mathbf{u}^T \mathbf{x}_n = u_1 x_{n1} + \cdots + u_D x_{nD} \tag{3.2}$$

which can be regarded as the orthogonal projection of $\mathbf{x}_n$ on the axis defined by $\mathbf{u}$.

We now consider another unit vector $\mathbf{v}$ that is orthogonal to $\mathbf{u}$, and project $\mathbf{x}_n$ orthogonally on it to get another scalar z_n:

$$z_n = \mathbf{v} \cdot \mathbf{x}_n = \mathbf{v}^T \mathbf{x}_n = v_1 x_{n1} + \cdots + v_D x_{nD}\,. \tag{3.3}$$

You will see that $\mathbf{x}_n$ is mapped to a point $(y_n, z_n)^T$ in a two-dimensional space, and the whole $\{\mathbf{x}_n\}_1^N$ can be mapped to $\left\{(y_n, z_n)^T\right\}_1^N$ in the same manner. It is easy to see that the resultant plots depend on the choice of $\mathbf{u}$ and $\mathbf{v}$. One option is to choose the pair of vectors that maximise the total variance of the projected data:

$$\begin{aligned} &\max_{\mathbf{u},\mathbf{v}} \ \mathrm{Var}\,(y) + \mathrm{Var}\,(z) \\ &\text{subject to} \ \ \|\mathbf{u}\| = 1, \|\mathbf{v}\| = 1, \mathbf{u} \perp \mathbf{v}\,. \end{aligned} \tag{3.4}$$

This means that we try to find a two dimensional space, i.e., a plane, such that the projected data on the plane spread as wide as possible rather than a plane on which the data are concentrated in a small area. It is known that the optimal projection vectors are given by the eigenvectors of the sample *covariance*

[1] NB : $\|\mathbf{u}\| = 1$ by definition.

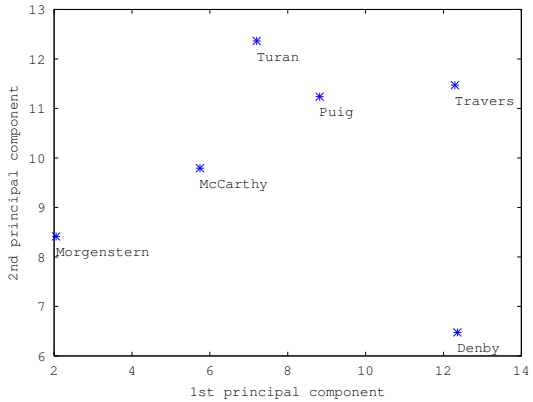


Figure 3.4: Plot of the critics in the 2-dimensional space defined by the first two principal components. (see Note 2)

matrix[2], $\mathbf{S}$, defined as

$$\mathbf{S} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T \tag{3.5}$$

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\,. \tag{3.6}$$

If scalar representation is preferred to the matrix one above, the element s_{ij} at i'th row and j'th column of $\mathbf{S}$, is given as:

$$s_{ij} = \frac{1}{N-1}\sum_{n=1}^{N}(x_{ni} - m_i)(x_{nj} - m_j) \tag{3.7}$$

$$m_i = \frac{1}{N}\sum_{n=1}^{N}x_{ni}\,. \tag{3.8}$$

To be explicit, let $\{\lambda_1, \ldots, \lambda_D\}$ be the eigenvalues sorted in decreasing order so that λ_1 is the largest and λ_D is the smallest, the optimal projection vectors $\mathbf{u}^*$ and $\mathbf{v}^*$ are the eigenvectors that correspond to the two largest eigenvalues λ_1 and λ_2. Figure 3.4 depicts the scatter plot with the method for the example shown in Note 2.[3]

Generally speaking, if we would like to effectively transform $\mathbf{x}_n$ in a D-dimensional space to $\mathbf{y}_n$ in a lower ℓ-dimensional space ($\ell < D$), it can be done with the eigenvectors, $\mathbf{p}_1, \ldots, \mathbf{p}_\ell$, that correspond to

[2]Covariance matrix is an extension of the variance for univariate case to the multivariate (multi-dimensional) case. We will see more details about covariance matrices in Note 8, where we consider Gaussian distributions.

[3]Someone might wonder how each film (review scores) contributes to the principal components. This can be confirmed with 'factor loading(s)', which is the correlation coefficient between the principal component and the review scores of the film.

the ℓ largest eigenvalues $\lambda_1, \ldots, \lambda_\ell$:

$$\mathbf{y}_n = \begin{pmatrix} \mathbf{p}_1^T \\ \vdots \\ \mathbf{p}_\ell^T \end{pmatrix}\mathbf{x}_n = \begin{pmatrix} \mathbf{p}_1^T\mathbf{x}_n \\ \vdots \\ \mathbf{p}_\ell^T\mathbf{x}_n \end{pmatrix}. \tag{3.9}$$

This technique is called *principal component analysis (PCA)* and widely used for data visualisation, *dimensionality reduction*, data compression, and feature extraction. [4]

3.5 Summary

In this chapter we have:

1. introduced the notion of clustering a data set into non-overlapping groups using hierarchical or partitional approaches;
2. described the most important partitional algorithm, K-means clustering;
3. defined a within-cluster mean squared error function for K-means clustering;
4. introduced the notion of dimensionality reduction for data visualisation;
5. described the technique, principal component analysis (PCA) for dimensionality reduction.

The key properties of K-means are that it:

- Is an automatic procedure for clustering unlabelled data.
- Requires a pre-specified number of clusters.
- Chooses a set of clusters with the minimum within-cluster variance.
- Is guaranteed to converge (eventually, in a finite number of steps).
- Provides a locally optimal solution, dependent on the initialisation.

3.6 Reading

Further reading

- Bishop, section 9.1 (on clustering)
- Bishop, section 12.1 (on principal component analysis)
- Segaran, chapter 3 (on clustering)

[4]It is almost always the case that reducing dimensionality involves degradation, i.e., loss of information. The ratio, $\sum_{i=1}^{\ell}\lambda_i / \sum_{i=1}^{D}\lambda_i$, indicates how much amount of information retains after the conversion.

Exercises

1. Consider a data set of one-dimensional samples $\{1, 3, 5, 6, 8, 9\}$.

   (a) Cluster the data set into two clusters using the k-means clustering, with the cluster initialised to 2 and 7.

   (b) What if the initial cluster centres are 6 and 8?

2. For k-means clustering, give an intuitive proof that the mean squared error does not increase after each iteration.

3. Assume we have calculated a covariance matrix $\mathbf{S}$ for a set of samples in a 3D space, and obtained eigenvectors and eigenvalues of S, which are given as follows:

Eigenvalues	Eigenvectors
$\lambda_1 = 1.0$	$\mathbf{p}_1 = (0.492404,\ -0.809758,\ 0.319109)^T$
$\lambda_2 = 0.9$	$\mathbf{p}_2 = (-0.086824,\ 0.319109,\ 0.943732)^T$
$\lambda_3 = 0.001$	$\mathbf{p}_3 = (-0.866025,\ -0.492404,\ 0.086824)^T$

   Using dimensionality reduction with PCA, plot the following four samples on a 2D plane.

$$\mathbf{x}_1 = (0.88932,\ -1.30533,\ 1.58282)^T$$
$$\mathbf{x}_2 = (1.07097,\ -0.83358,\ 2.49964)^T$$
$$\mathbf{x}_3 = (0.14555,\ -0.27002,\ 2.22394)^T$$
$$\mathbf{x}_4 = (0.49218,\ -0.44141,\ 1.25416)^T$$