

Chapter 14

Subspace Clustering

14.1 Motivation and overview

All the clustering methods that we have looked at in the previous chapters assume that we are clustering the points in the full d -dimensional space. However, as we saw in Chapter 3, as the dimensionality increases the points tend to get scattered away from the center towards the boundaries of the enclosing hypersphere or hypercube. Furthermore, as dimensionality increases, there may not be much difference between the closest and farthest neighbor for a given point. This “curse of dimensionality” makes clustering more challenging in high dimensional spaces.

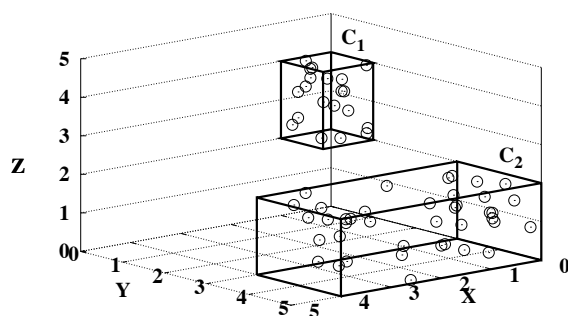


Figure 14.1: Subspace Clusters

In this chapter we will look at methods that try to find subspaces where the points may cluster well. That is, instead of trying to cluster the points in all the d dimensions, our goal is to find lower dimensional projections with a dense set of points, i.e., the subspace clusters. For example, consider the clusters shown in Figure 14.1. C_1 can be considered to be a cluster in the full three dimensional space, since all the points are within some range threshold, say 2 units, in each dimension. However, when we consider the cluster C_2 , we note that the points appear scattered in the full three dimensional space spanned by the cluster. In particular, the X axis contributes the most to the scattering, whereas if we consider the projection of the points on only the dimensions Y and Z , we find that they cluster well, as shown in the projection shown in

Figure 14.2(c).

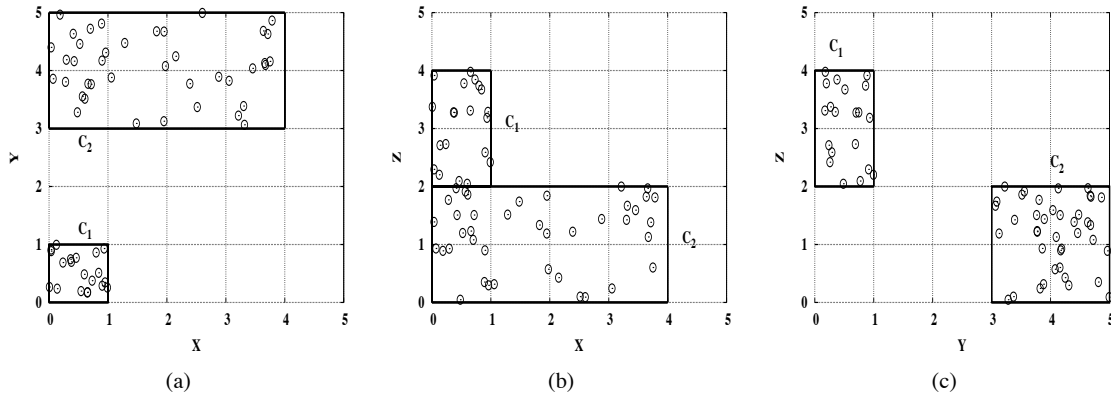


Figure 14.2: Cluster Projections

Note that unlike PCA or SVD methods, which identify *one* lower-dimensional subspace (a set of dimensions) that best represents the *entire* dataset, our goal is to find *several* lower-dimensional subspaces that represent, or cluster, some *subset* of the points in the dataset.

We may represent subspace clustering using an extended version of the graph model we have been employing throughout the cluster chapters. In this case, we will have a multigraph, where the vertices may represent either clusters or points, and each edge represents a relationship in a set of dimensions. Thus, a clustering is the graph induced by a set of dimensions, that is, a sub-graph of the multi-graph where all edges have the same label, meaning that they belong to the same set of dimensions.

There are several issues while determining a subspace cluster. The first concerns the determination of the proper subspace, that is, the set of dimensions that yields a good cluster in terms of the quality or semantic meaning. The second issue deals with the shape of the cluster. We may assume that the cluster is generated from a multivariate uniform or normal distribution, which yield a hyper-rectangle or hyper-ellipsoid shape, respectively. The third issue is whether we assume the dimensions are independent or not. If we assume independence, then the clusters will be *axis-aligned*. For example, in Figure 14.1, each of rectangle has sides parallel to (or aligned with) the major axes. If the clusters were normal-shaped, then the axes of the cluster would also be aligned with the principal axes. On the other hand, if the dimensions are not independent, and we allow linear dependence, then the clusters will appear rotated in space, and will not necessarily be axis-aligned. For the former case, we can find subspaces that are essentially subsets of the original set of dimensions, whereas for the latter, we have to find subspaces with dimensions which are linear combinations of the original set of dimensions. Non-axis aligned clusters will thus have some relationship to methods like PCA/SVD. The key difference here is that instead of determining only one subspace, we have to identify several promising subspaces where points cluster well. Another issue while implementing subspace clustering techniques is to devise metrics that allow one to compare clusters defined in different dimensioned spaces. We should be careful about the impact of differences in terms of the magnitude of the attributes as well as the effects of different number of dimensions.

14.2 Algorithmic description

In this section we describe the two main strategies for determining subspace clusterings: axis parallel clusters and arbitrary subspace clusters. We start by discussing the exhaustive strategy for determining the best set of dimensions, where the cluster shape is inherent to the clustering technique.

14.2.1 Exhaustive

For simplicity, let us focus on axis parallel clusters and let us assume that the dimensions are independent. The subspace clustering task is to find all dense groups of points in lower dimensional spaces formed by selecting some subsets of the original dimensions. If the data space has d dimensions then there are $2^d - 1$ potential non-empty subspaces or subsets of dimensions we may have to consider, making it extremely expensive to perform a brute-force search of all subspaces. The complexity is $O(C \times 2^d)$ in the worst case, where d is the number of dimensions and C is the cost of the clustering technique employed (e.g., K-means) for each dimension combination. In order to apply such a strategy we need just a metric that allows us to compare the clusterings obtained for the different subspaces. If we allow non-axis parallel clusters, but restricted to linear relationships, we may employ PCA/SVD like methods to enumerate different low-dimensional spaces with dimensions that are linear combinations of the original d dimensions. In this case, we potentially have an infinite number of choice available in terms of the subspaces.

14.2.2 Axis Parallel Dense Subspace Clusters

In this section we discuss two strategies for generating axis parallel clusters: level-wise and randomized projection. They are discussed in the next subsections.

Level-wise Approach

Let \mathcal{D} be a dataset of n points in d -dimensions. As before, we use $\mathbf{x}_i = (x_i^1, x_i^2, \dots, x_i^d)$ to denote a the i -th point, and $\mathbf{x}^j = (x_1^j, x_2^j, \dots, x_n^j)$ to denote the j -th dimension. Assume that each dimension is partitioned into b intervals or bins of equal width. This gives us a partitioning of the d -dimensional space into a total of b^d hypercubes or cells. Each cell \mathbf{c} , can be defined as the cross-product of the intervals/bins in each dimension, given as $[l^1, u^1] \times [l^2, u^2] \times \dots \times [l^d, u^d]$. A point \mathbf{x}_i is said to belong to cell \mathbf{c} , written as $\mathbf{x}_i \in \mathbf{c}$, provided $x_i^j \in [l^j, u^j]$ in each dimension j . Likewise, given any subset of the dimensions A , we can find the set of points contained in the cells spanned by those dimensions. We simply ignore the irrelevant dimensions, and check if $x_i^j \in [l^j, u^j]$ in each relevant dimension $j \in A$. In this case we write $\mathbf{x}_i \in \mathbf{c}_A$.

Let A be a set of relevant dimensions. Define the density or support of a cell \mathbf{c} in the set of dimensions A as follows:

$$sup_A(\mathbf{c}) = |\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbf{c}_A\}| \quad (14.1)$$

In other words, the density of the cell \mathbf{c} restricted to the dimensions in A , is simply the number of points that belong to the cell \mathbf{c} (considering only the dimensions A). A cell is said to be *dense* or frequent in A , if $sup_A(\mathbf{c}) \geq minsup$, where $minsup$ is some user-defined density threshold.

Define a *subspace cluster* as a maximal set of dense cells in k dimensions, that share at least a face in common. It is important to note that, as shown in Figure 14.2, if the cluster is dense in a higher dimensional subspace A , it is dense in any of its subspaces $A' \subset A$. Notice also the similarity between dense subspaces and itemsets. For example, if we treat each bin $[l, u]$ in a given dimension as an item, then each k -dimensional

cell is essentially a k -itemset, and the main step in computing the maximal dense clusters is to compute the dense/frequent cells.

Levelwise Algorithm: A simple level-wise search for dense cluster proceeds as follows. First we project all the points on each of the dimensions and make note of the dense intervals (single dimensional cells). In the next step, we consider all possible two-dimensional cells obtained as combinations of the dense intervals, and compute their density. In general, in iteration k , we combine the $(k - 1)$ dense cells to create candidate k -dimensional cells. We compute their support and retain only the dense k -dimensional cells. The iterative process continues until there are no new dense cells found. At the end, we check if there are any two k -dimensional cells that share a face, and merge them into a larger cluster to obtain the final maximal subspace clusters.

Consider the example shown in Figure 14.1. We first count the dense intervals in each dimensions as shown in Table 14.1, with $minsup = 10$.

Dimension	Interval	Support
X	$[0, 1)$	37
X	$[3, 4)$	11
Y	$[0, 1)$	20
Y	$[3, 4)$	17
Y	$[4, 5)$	23
Z	$[0, 1)$	14
Z	$[1, 2)$	26
Z	$[3, 4)$	12

Table 14.1: Dense Single Dimensional Cells

Next, we count the possible two dimensional candidate cells. The ones that are dense are listed in Table 14.2.

Dimensions	Cell	Support
XY	$[0, 1) \times [0, 1)$	20
XY	$[0, 1) \times [4, 5)$	10
XZ	$[0, 1) \times [1, 2)$	10
XZ	$[0, 1) \times [3, 4)$	12
YZ	$[0, 1) \times [3, 4)$	12
YZ	$[3, 4) \times [1, 2)$	12
YZ	$[4, 5) \times [1, 2)$	14

Table 14.2: Dense Two Dimensional Cells

Finally, we create the dense three dimensional candidates and compute their support. We find that the only dense three dimensional dense cell is: $[0, 1) \times [0, 1) \times [3, 4)$ with support 12.

Finally, we will merge adjacent cells that share a face (a point in 1 dimension, an edge in 2 dimensions, a side in 3 dimensions, and so on), and report all the maximal clusters in the various subspaces. For instance, looking at Table 14.1, we obtain the following one-dimensional clusters: $X[0, 1)$, $X[3, 4)$, $Y[0, 1)$, $Y[3, 5)$, $Z[0, 2)$, $Z[3, 4)$. Notice how $Y[3, 4)$ and $Y[4, 5)$ have been merged into the longer interval $Y[3, 5)$. Likewise for $Z[0, 2)$. Among the 2-dimensional cells, $YZ : [3, 4) \times [1, 2)$ and $YZ : [4, 5) \times [1, 2)$, will be merged into the

larger cluster $YZ : [3, 5) \times [1, 2)$. The final set of maximal subspace clusters are then given as: $X : [3, 4)$ and $Z : [0, 2)$ as one-dimensional maximal clusters, $XY : [0, 1) \times [4, 5)$, $XZ : [0, 1) \times [1, 2)$, and $YZ : [3, 5) \times [1, 2)$ as two-dimensional maximal clusters, and $XYZ : [0, 1) \times [0, 1) \times [3, 4)$ as the three-dimensional maximal cluster. Notice how the level-wise grid-based approach is able to capture only the dense subsets of the true clusters.

There are a number of limitations of the simple level-wise approach for finding dense subspace clusters. First is the complexity itself; given b bins in each dimension, there are $O(b^d)$ cells. Second, the method is quite sensitive to the choice of the number of bins b , and also the bin boundaries. Third, the curse of dimensionality is still a factor, since out of the b^d cells many of them will be empty, and furthermore, it is quite likely that a cell contains only a single point, which will make it hard to find the dense cells. The fixed *minsup* threshold is also a problem, since it will have to be rather low to capture higher-dimensional subspaces. Unfortunately this means that many smaller subspaces will be very dense, blowing up the complexity of the method. If the threshold is too high, then many true subspace clusters will be missed. The example above brought some of these limitations to light.

Randomized Projections

Instead of the level-wise search approach which is a complete method, we consider an alternative randomized approach to find the dense subspace clusters. This formulation does not rely on binning each dimension. Instead it defines a dense subspace cluster as a pair (C, A) , where $C \subseteq \mathcal{D}$ is the set of points, and A is a subset of relevant dimensions, provided that the cluster satisfies the following conditions:

1. Density criteria: C should be dense, i.e., $sup_A(C) \geq minsup$
2. Clustering criteria: For all dimensions $j \in A$, $\max(x_i^j) - \min(x_i^j) \leq w$ where \mathbf{x}_i is a point in the cluster C , and w is a user-defined maximum width or range of values. That is, for those dimensions relevant to the cluster, the points are within a bound or range w .
3. Scattering criteria: For all dimensions $j \notin A$, $\max(x_i^j) - \min(x_i^j) > w$. That is, for the irrelevant dimensions the points are scattered wider than the bound w .

The randomized projection-based algorithm given in Algorithm 14.1. The method initially picks a seed point \mathbf{x}_p , and then selects a random sample of points, S_j , of size $|S_j| = r$. For each point in the random sample S_j , we check if the point $\mathbf{x}_q \in S_j$ is within a distance w of \mathbf{x}_p . If so, this dimension is added to the set of relevant dimensions A . We next check the density of the region $R_A(\mathbf{x}_p, w)$, which is defined as the rectangle of width w in each dimension in A , centered at \mathbf{x}_p . If it is dense, the cluster (C, A) is stored for final processing. Since the method is randomized, it picks multiple random samples S_j and multiple seed points \mathbf{x}_p , and finally reports only the most promising clusters, such as those with the large support and having many dimensions.

14.2.3 Skewed Subspace Clusters

The problem with all of the above discussed approaches is that they all assume that subspace-clusters are axis-parallel hyper-rectangles. A skewed or non-axis parallel subspace cluster as shown in Figure 14.3 cannot be found by both of the subspace methods above. Such subspace clusters are characterized by a new set of dimensions, which are linear combinations of the original dimensions. We will now look at methods that can find such skewed subspace clusters.

```

RANDOMPROJECTIONS ( $\mathcal{D}$ ,  $w$ ,  $minsup$ ):
1 foreach ( $i = 1$  to  $maxiter$ ) do
2   Pick a random seed point  $\mathbf{x}_p \in \mathcal{D}$ ;
3   foreach ( $j = 1$  to  $max\ samples$ ) do
4     Pick a sample  $S_j \subseteq \mathcal{D}$  of size  $r$ ;
5     foreach ( $\mathbf{x}_q \in S_j$ ) do
6       foreach Dimension  $k \in [1 : d]$  do
7         if ( $|\mathbf{x}_p^k - \mathbf{x}_q^k| \leq w$ ) then
8           Add  $k$  to the list of valid dimensions  $A$ ;
9            $C = \mathcal{D} \cap R(\mathbf{x}_p, w)$ ;
10          if ( $sup_A(C) \geq minsup$ ) then
11            Store  $(C, A)$  as a potential subspace cluster;
12 Rank all clusters  $(C, A)$ , and output best clusters;

```

Algorithm 14.1: Random Projection Algorithm

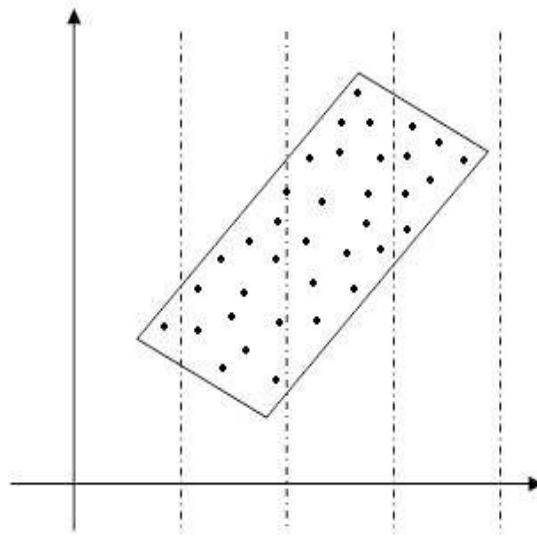


Figure 14.3: Example of a non-axis parallel cluster

Kmeans Projective Clustering

In Projective Kmeans the idea is to discover clusters in subspaces spanned by different combinations of dimensions.

We define a r -subspace as the subspace spanned by r linearly independent vectors. For example, a point is a 0-subspace, a line is a 1-subspace, a plane is a 2-subspace and so on. Given a r -subspace S , we define

the distance of a point \mathbf{x}_p to the subspace S as follows:

$$\delta(\mathbf{x}_p, S) = \min_{\mathbf{x}_q \in S} \delta(\mathbf{x}_p, \mathbf{x}_q) \quad (14.2)$$

That is, the distance between \mathbf{x}_p and S is defined to be the distance between \mathbf{x}_p and its closest point \mathbf{x}_q in S . Note that geometrically, the closest point in S is really the projection of \mathbf{x}_p onto the space S , i.e., $\mathbf{v} = \text{proj}_S(\mathbf{x}_p)$. The distance between the space S and \mathbf{x}_p is then simply the (perpendicular) distance between \mathbf{v} and \mathbf{x}_p .

Define the root-mean-square (rms) distance between a point \mathbf{x} and a set of points P as follows:

$$\text{rms}(P, \mathbf{x}) = \sum_{\mathbf{y} \in P} \delta^2(\mathbf{x}, \mathbf{y}) \quad (14.3)$$

where δ^2 is the squared (Euclidean) distance. Likewise, the *rms* between two points sets P and C can be defined as:

$$\text{rms}(P, C) = \sum_{\mathbf{x} \in P} \min_{\mathbf{y} \in C} \delta^2(\mathbf{x}, \mathbf{y}) \quad (14.4)$$

Given a set of points P , the single point \mathbf{c} that best approximates P is the mean point, or the *centroid*. Also recall that given a value k for the number of clusters, in k -means clustering, we need to find the set $C = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k\}$ of k centroids that best approximates the collection of points P , given as:

$$C^* = \arg \min_C \text{rms}(P, C) \quad (14.5)$$

We now want to generalize the above optimization criteria of k -means. Note that k -means clustering is interested only in the 0-subspace approximations, i.e., points, which are the centroids of the clusters. What we now desire is to find k subspaces of different dimensionalities that best represent the point set P .

Instead of single points, we want to find k different subspaces $\mathcal{S} = \{S_1, S_2, \dots, S_k\}$ of differing dimensionalities, so that we minimize the rms between P and \mathcal{S} :

$$\mathcal{S}^* = \arg \min_{\mathcal{S}} \left\{ \sum_{\mathbf{x} \in P} \min_{S_i \in \mathcal{S}} \delta^2(\mathbf{x}, S_i) \right\} \quad (14.6)$$

PROJECTIVEKMEANS (\mathcal{D}, k):

- 1 Partition \mathcal{D} into k parts randomly: $\{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_k\}$
- 2 **while** (*convergence condition not met*) **do**
- 3 **for** ($j=1$ to k) **do**
- 4 SVD(\mathcal{D}_j) = $U_j \Delta_j V_j^T$
- 5 $S_j = (\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_{r_j})$
- 6 **foreach** (*point* $\mathbf{x}_i \in \mathcal{D}$) **do**
- 7 Assign point \mathbf{x}_i to the closest subspace in \mathcal{S}

Algorithm 14.2: Projective k -means Algorithm

The projective Kmeans algorithm is shown in Figure 14.2. It has three main steps. Initially in step 1) we randomly partition the dataset \mathcal{D} into k parts. Then we iteratively refine these partitions, detecting the proper subspace of dimensionality r_j for each partition via the singular value decomposition (SVD). For example,

we choose the r_j largest singular values and the subspace spanned by their corresponding right singular vectors as S_j . In other words, these r_j vectors $v_1^T, \dots, v_{r_j}^T$ serve as a basis for points in the subspace S_j . The subspace dimensionality r_j can be chosen so as to keep the reconstruction error below some threshold. Once the new subspaces \mathcal{S} are found, we reassign each point in the dataset to the closest subspace in \mathcal{S} . These steps are repeated until convergence.

Note that if we always choose dimensionality $r_j = 0$ for all subspaces S_j (in which case there is no need for the SVD), the best subspace that approximates each partition D_j will be its centroid, and thus the projective Kmeans essentially becomes the same as the Kmeans algorithm we studied earlier.