Data Mining: Clustering/Unsupervised Learning k-Means Clustering

Unsupervised Learning

Unsupervised Learning occurs when we need to predict **an unknown categorical function** that partitions a set of data points based on the similarity between the points.

More formally,

> Given a collection of data points $D = \{x_1, \ldots, x_n\}$, where $x_i = (a_1, \ldots, a_M)$ over some set of attributes/dimensions $A = \{A_1, \ldots, A_M\}$, and a distance metric $d : dom(A) \times dom(A) \longrightarrow \mathbb{R}$, infer a categorical function $f : dom(A) \longrightarrow \mathbb{C}$, where $\mathbb{C} = \{c_1, \ldots, c_k\}$ for some k.
>
> The learned function f should minimize
>
> $$\sum_{x,y \in D(f(x)=f(y))} d(x,y)$$
>
> and maximize
>
> $$\sum_{x,y \in D(f(x)\neq f(y))} d(x,y)$$

A set of data points $C_i = \{x \in D | f(x) = c_i \in \mathbb{C}\}$ is typically called a **cluster**. So, a **cluster** is a *collection of data instances deemed to be **similar** to each other and **dissimilar** to other data instances*.

Clustering. **Clustering** a.k.a. the process of organizing data instances into groups whose members are similar in some way (and different from members of other clusters) is another name for the *unsupervised learning* problems/methods/algorithms.

Dataset. The dataset in a **clustering** task is a collection $D = \{x_1, \ldots, x_n\}$ of data points over the set of attributes $A = \{A_1, \ldots, A_M\}$.

Note: The key difference between **clustering** and **classification** tasks is that in **classification** tasks, the dataset includes a **class variable**.

In **clustering** tasks, class variable is not available. The task is **not to predict** the class of each data point, but to **organize data points into groups by their perceived similarity**.

Example. Consider the four pictures in Figure 1.

- Scatterplot (a) shows an example of three easy-to-distinguish clusters with clear boundaries and clear membership.
- Scatterplot (b) shows an example of two clusters that are relatively easy to identify. However, the boundary and the exact membership in the clusters is subject to discussion and not all algorithms may be able to determine correctly the desired cluster for each poiint.
- Scatterplot (c) shows an example of two clusters, where membership (in one of the two clusters) is determined by more than just proximity. Clustering algorithms based solely on assigning points to clusters based on their proximity to each other (or to some other locations) may be unable to identify the "ring" cluster properly.
- Scatterplot (d) shows an examples of two clusters with a "bridge" between them, which makes it hard for both the algorithms and the humans to identify membership of certain data points. Additionally, the diagram contains three *outliers*: data points that lie well outside any cluster. Clustering algorithms that do not properly detect outliers may be unable to detect proper cluster boundaries because of it.

Clustering Algorithm Example: k-Means Clustering Algorithm Family

Historical Note. The **k-Means Clustering algorithm** has been discovered and rediscovered by researchers in different fields many times. It appeared in the works of Lloyd (1957) [1], Forgey (1965)[**?**], Friedman and Rubin (1967) and McQueen (1967).

Applicability. Dataset $D = \{x_1, \ldots, x_n\}$, $x_i = (x_{i1}, \ldots x_{iM}) \in \mathcal{R}^m$.

In order to be able to use **k-means clustering algorithm** the notion of the **mean** must exist for the domain of each attribute A_i.

Algorithm Outline. The algorithm takes as input the dataset $D = \{x_1, \ldots x_n\}$ and an integer k — the number of clusters to build.

The algorithm proceeds as follows.

1. Select k initial cluster centroids.

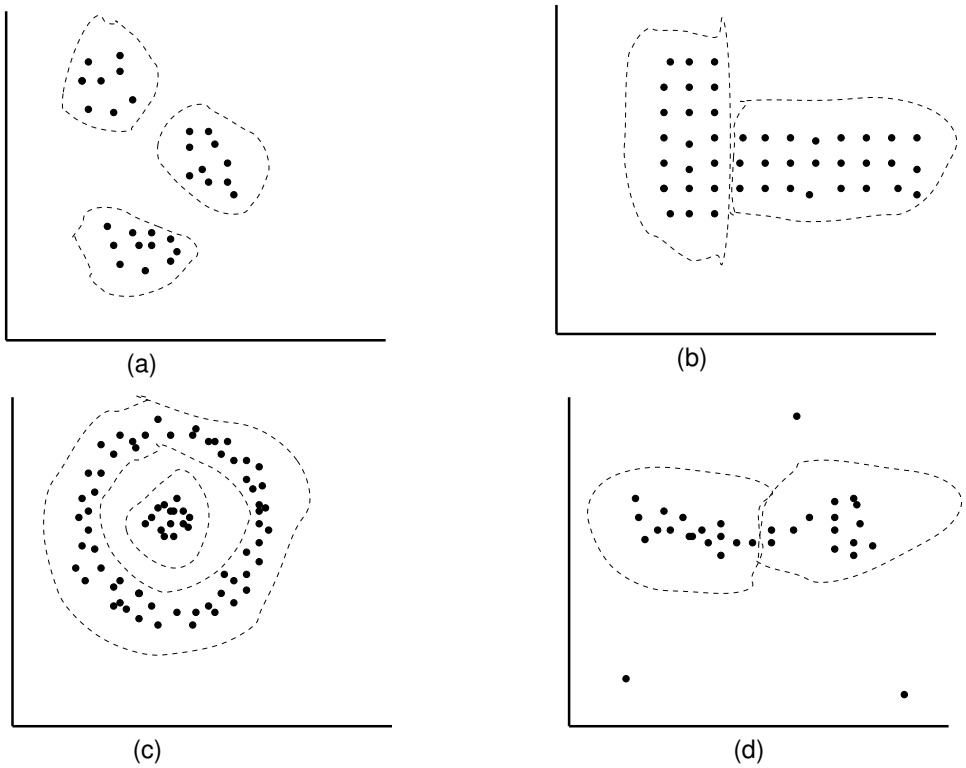

Figure 1: Clusters in data.

2. On each step, for each data point compute its distances from each of the cluster centroids and assign it to the **closest centroid**.
3. Recompute cluster centroids.
4. Steps 2 and 3 are repeated until the process converges.

Details: cluster centroids. Initial centroid selection:

- Pick k random data points from the datasets. (**seeds**).
- Use the following selection procedure SelectCentroids:
  1. Compute the **centroid** c of the entire dataset D.
  2. First centroid, m_1 is the $x \in D$, such that $d(m_1, c) = \max_{x \in D}(d(x, c))$ (the point furtherest away from the centroid).
  3. Pick m_2 such that $d(m_1, m_2) = \max_{x \in D}(d(m_1, x))$.
  4. Pick m_i, such that $\sum_{j=1}^{i-1}(d(m_j, m_i)) = \max_{x \in D} \sum_{j=1}^{i-1}(d(m_j, x))$.
- Select a sample $S \subset D$, $|S| > k$. Perform the procedure described above on S. (This helps fight **outliers**).
- Pre-select the seeds before starting the algorithm (and, for example, locate them as the first k data points in D).

Centroid recomputation:

```
Algorithm diskKMeans(D,k);
begin
  m[] := SelectInitialCentroids(D,k);
  repeat
    for j := 1 to k do
      s[j] := (0,0,0,...,0);  // s[] - family of vectors of size dim(D)
      num[j] := 0;  // num[] - number of points in each cluster
      cl[j] := ∅;  // cl[] - actual clusters
    endfor
    foreach x ∈ D do
      cluster := arg min_{j=1,...,k}(dist(x, m_j));  // assign x to the cluster
      cl[cluster] := cl[cluster] ∪ {x};
      s[j] := s[j] + x;
      num[j] := num[j] + 1;
    endfor
    for j := 1 to k do m[j] := s[j]/num[j];
  until isStoppingCondition(m[],cl[]) = true;
  output cl[];
end
```

Figure 2: Disk version of the k-Means clustering algorithm.

- Let $m_{t,1}, \ldots m_{t,k}$ are the k centroids on step $t > 1$. Let $C_{t,1}, \ldots, C_{t,k}$ be the k clusters assigned on step t. The new centroids $m_{t+1,1}, \ldots, m_{t+1,k}$ are computed as **means** of points in their clusters on previous iteration:

$$m_{t+1,i} = \frac{1}{|C_{t,i}|} \sum_{x \in C_{t,i}} x.$$

Stopping criteria. The k-means algorithm can use any of the following stopping (convergence) criteria:

1. no (or minimum) reassignment of points between clusters;
2. no (or minimum) change in cluster centroids;
3. insignificant decrease in the **sum of squared error**:

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_{t,j}} d(x, m_{t,j}).$$

Disk Version of the k-means clustering algorithm. Figure 2 has the pseudocode for a version of the k-means clustering algorithm that works with data stored in secondary storage (on disk).

Strengths and Weaknesses of the k-Means Clustering Algorithm

Strengths. The key strengths are:

- *Simplicity*.

- *Efficiency*.

  Time Complexity: $O(tkn)$ ($n = |D|$, k – number of clusters, t – number of iterations)

  Data Complexity: $t \cdot B(D)$ ($B(D)$ — number of disk blocks in holding D).

Weaknesses and addressing them.

- **Applicability. Mean** of a set of data points must be computable.

  Note: A variation of k**-means clustering algorithm** called k**-modes clustering** can be used with *categorical data*.

- **Need for** k. The user must "guess" proper k. If k is not the number of true clusters in the dataset, the k**-means clustering algorithm** may perform poorly.

  Example. Figure 3.(a). The dataset has two clusters (circles). If k**-means clustering** is run for $k = 3$, one of the reported clusters may (will) combine points from two "real" clusters.

  Note: Run k**-means clustering** multiple times with increasing values of k. *Typically, the number of clusters is small enough to make this feasible*.

- **Sensitivity to initial centroid choice**. Poorly selected centroids may yield bad results even when clusters are clearly defined.

  Example. Figure 3.(b). If centroids are selected randomly, two data points from the same cluster may be selected (solid boxes). Dashed boxes show the clusters computed by the k**-means clustering algorithm** with this selection of initial centroids. Figure 3.(d) shows the results of the k**-means clustering algorithm** when initial centroids are picked from different real clusters.

  Dealing with initial centroid choice sensitivity.

  - **Random selection** works well when $|D|$ is large.
  - Use procedure SelectCentroids described above. It attempts to place initial centers **as far away from each other as possible**.
  - Manually select seeds prior to running k**-means clustering**.

- **Sensitivity to outliers.**

  > *An **outlier** is a data points located very far away from other data points (i.e., isolated).*

  The k**-means clustering algorithm** does not detect outliers. Outliers are assigned to clusters on each step, and may affect undue influence on the location of the cluster's centroid. This may yield incorrect cluster boundaries.

  Example. Figure 3.(c). The data set consists of two clusters and a single outlier. If the outlier is detected, two clusters can be properly separated. Otherwise, the outlier gets assigned to one of the clusters, "pulls" the cluster centroid away from the true position, and allows the second cluster to "annex" some of the points of the first cluster.

  Dealing with outlier sensitivity.

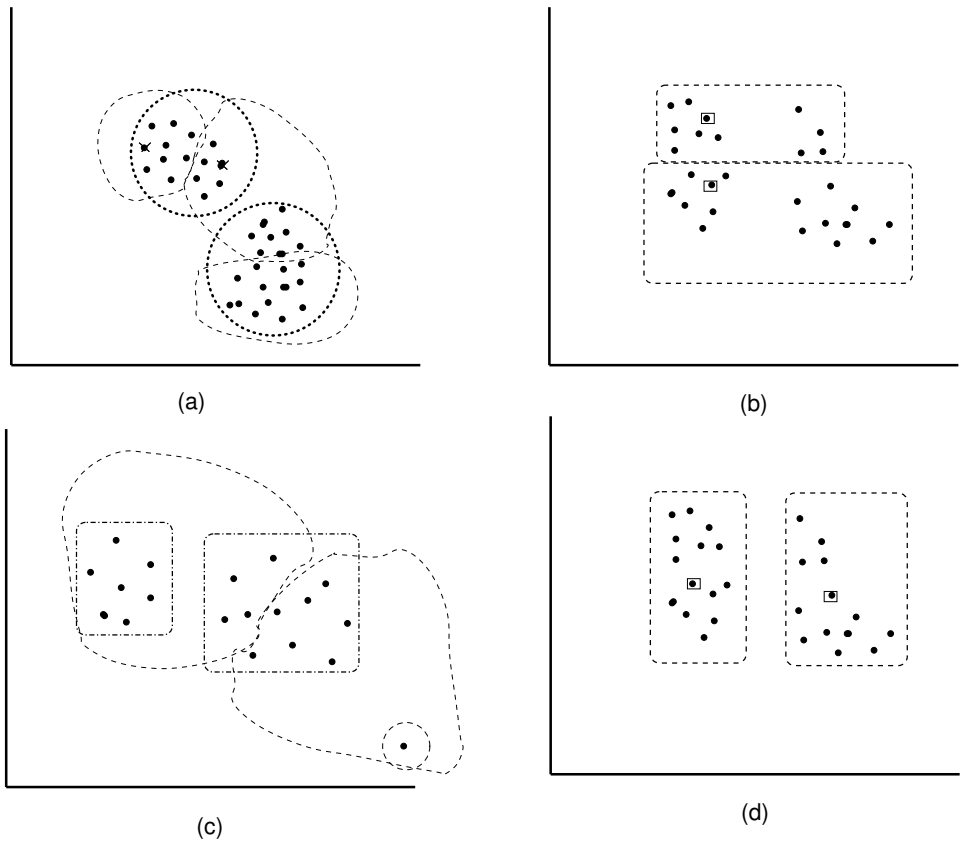

Figure 3: Weaknesses of k-means clustering illustrated.

- **Outlier detection during k-means clustering.** Discover points that are too far away from the centroid of their cluster and remove these points from consideration. Use a specific threshold to determine if a point is *too far away from the cluster centroid*.
- **Random sampling**. Randomly sample the data, and cluster the sample. *The probability of selecting an outlier into a sample is* ***very small***.

- **Cannot properly detect clusters with close centers.** If two clusters have centers near each other, the k**-means clustering algorithm** will note be able to properly detect them.

  Example. Figure 1.(c).The two clusters: the outer ring and the inner kernel have centers that are located close to each other. Therefore, instead of recognizing the *inside/outside* clusters, k**-means clustering** will yield a *left/right* or *top/bottom* or similar split (based on the initial centroid assingments).

  Essentially, the k**-means clustering algorithm** *replaces the problem of separating points in different clusters with the problem of separating cluster centroids*. Thus, in this example it will move towards increasing the distance between cluster centers.

  Dealing with complex cluster shapes/co-centered clusters.

  - *Sometimes, it is not a problem*. (due to application/dataset specifics)
  - Outside of the latter observation, *there is really no way to compensate for this*.

Representation of Clusters

Cluster = data points. Represent each cluster as the collection of points in it.

- **Good** for applications interested in properties of individual data points.
- **Bad** for applications interested in *simple*, *intuitive* descriptions of discovered clusters.

Cluster = centroid + radius. Each cluster is represented by a centroid and a radius.

- **Good**, because ***succinct*** (and simple).
- Information is available from the k**-means clustering algorithm**. No new computations are needed.
- **Bad**, because ***crude***.
- May yield **cluster overlaps**.

Cluster = Class. Each data point is assigned a class label derived from the cluster it has been assigned. The new dataset D' constructed this way is used to train a classifier that has explanatory power (e.g., a *decision tree* or a set of *class association rules*).

The cluster is represented as the subset of classifier/set of rules identifying it in the classifier model.

- **Good,** because of ***explanatory power***. (We may be able to articulate what the cluster really represents.)
- **Bad,** because ***time-*** and ***resource-consuming***. Requires significant extra effort (see: Learning, Supervised.).
- May fail to produce *meaningful* explanation.

Cluster = frequent values. Represent each cluster via a small subset of *frequently occuring* or *typical data points*.

- **Good**, because follows the *"Stuff like this"* intuition.
- May take some extra work to identify **good representatives**.
- May allow for comparison of new data points.
- But may also miss important information about the structure of the cluster (e.g., Figure 1.(c)).

References

[1] S.P. Lloyd. Least Squares quantization in PCM. *Unpublished Bell Labs Tech. Note.* (1957). Portions presented at *Institute for Mathematical Statistics Meeting*, 1957. *IEEE Transactions on Information Theory*, vol. IT-28, pp. 129—137, MArch 1982.

[2] E. Forgey. Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Calssification. *Biometrics*, vol. 21, p. 768 (abstract), 1965.