Distance-Based Clustering

Moo K. Chung

University of Wisconsin-Madison, USA

mkchung@wisc.edu

Abstract. In this short lecture, we explain how to perform distance-based clustering and compute the clustering accuracy.

1 Basics on Clustering

Clustering is the process of partitioning a dataset into groups such that data points within a cluster share more similarity with each other than with those in other clusters. The goal of clustering is to uncover underlying structures in data. Various clustering algorithms exist, including k-means clustering, hierarchical clustering, and density-based clustering, each suited for different types of data and applications.

A commonly used method is k-means clustering, where the data is partitioned into k clusters by minimizing the sum of squared Euclidean distances between points and their respective cluster centroids. The algorithm iteratively updates cluster assignments and centroids until convergence. Another approach is hierarchical clustering, which builds a tree of nested clusters through either agglomerative (bottom-up) or divisive (top-down) strategies. Density-based clustering, such as DBSCAN, identifies clusters as dense regions separated by lower-density areas, making it robust to noise and capable of finding arbitrarily shaped clusters.

Given a dataset $X = \{x_1, x_2, \dots, x_n\}$, clustering algorithms aim to assign each data point x_i to a cluster label $y_i \in \{1, \dots, k\}$. The effectiveness of clustering is most often evaluated using the clustering accuracy when true labels are available.

2 Computational Complexity

Consider a dataset of n data points indexed by $[n] = \{1, \cdots, n\}$. We aim to partition the dataset into k clusters labeled as $C_1, \cdots, C_k$ such that

$$\bigcup_{i=1}^{k} C_i = [n], \quad C_i \cap C_j = \emptyset, \quad \forall i \neq j.$$

Let $\boldsymbol{\Pi}_n$ denote the space of all partitions of n data points. We list partitions in lexicographical order, ensuring that elements within each cluster appear in

increasing order. For example, $\{\{1,4\},\{2,3\},\{5\}\}$ is a partition of $\{1,\cdots,5\}$ (Simion 2000).

A partition of $\{1,\cdots,n\}$ is defined as *non-crossing* if for any four elements $a<b<c<d$ such that a,c belong to the same cluster and b,d belong to the same cluster, then both clusters must be identical (Simion 2000). The number of non-crossing partitions corresponds to the number of possible hierarchical clusterings or, equivalently, the number of binary trees with n nodes. Let κ_n be the number of possible non-crossing partitions, also known as the Catalan number, given by

$$\kappa_n = \frac{1}{n+1}\binom{2n}{n}.$$

For example, $\kappa_1 = 1$ and $\kappa_2 = \frac{1}{3}\binom{4}{2} = 2$. The space of non-crossing partitions is a subset of $\boldsymbol{\Pi}_n$. The total number of all possible partitions in $\boldsymbol{\Pi}_n$, denoted as τ_n, is called the Bell number. The Bell number does not have a closed-form analytic expression but is given iteratively as

$$\tau_n = \sum_{i=0}^{n-1}\binom{n-1}{i}\tau_i.$$

For large n, the Catalan number serves as a lower bound for the Bell number. Using Stirling's approximation (Feller 2008, Chung et al. 2019), the Catalan number can be asymptotically approximated as

$$\kappa_n \sim \frac{4^n}{\sqrt{\pi n}(n+1)}.$$

Thus, **the number of partitions increases exponentially with data size, making partition-related problems computationally expensive (NP-hard).** An NP-hard problem (Nondeterministic Polynomial-time hard) has no known efficient (polynomial-time) algorithm to solve it for all cases.

The total number of ways to partition n data points into k clusters is given by the Stirling number of the second kind,

$$S(n,k) = \left\{ n \atop k \right\}.$$

There are $n(n-1)/2$ possible two-cluster partitions and $n(n-1)(n-2)/6$ possible three-cluster partitions. More generally, there are $\mathcal{O}(n^k)$ possible k-cluster partitions, making global optimization computationally infeasible. Most k-means clustering algorithms require $\mathcal{O}(n^k)$ runtime to achieve the optimal clustering assignment.

3 Clustering Accuracy

Let y_i be the true classification label for the i-th data. Let $\widehat{y}_i$ be the estimate of y_i we determined from classification algorithms. Let $y = (y_1,\cdots,y_n)$ and

$\widehat{y} = (\widehat{y}_1, \cdots, \widehat{y}_n)$. The classification accuracy $A(y, \widehat{y})$ is given by

$$A(\widehat{y}, y) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(\widehat{y} = y),$$

where $\mathbf{1}$ is the indicator function.

In clustering, there is no direct association between true clustering labels and predicted cluster labels. Given k clusters $C_1, \cdots, C_k$, its permutation $\pi(C_1), \cdots, \pi(C_k)$ is also a valid cluster for $\pi \in \mathbb{S}_k$, the permutation group of order k. Suppose [1 1 2 1 1 3 3] is the estimated cluster labels when the true labels are [1 1 1 2 2 3 3]. Then any permutation of estimated cluster labels such as [2 2 1 2 2 3 3] and [3 3 1 3 3 2 2] are other valid cluster labels. There are $k!$ possible permutations in $\mathbb{S}_k$ (Chung et al. 2019). Thus the clustering accuracy is modified as

$$A(\widehat{y}, y) = \frac{1}{n} \max_{\pi \in \mathbb{S}_k} \sum_{i=1}^{n} \mathbf{1}(\pi(\widehat{y}) = y).$$

This a modification to an assignment problem and can be solved using the Hungarian algorithm in $\mathcal{O}(k^3)$ run time (Edmonds & Karp 1972). In Matlab, it can be solved using `confusionmat.m`, which tabulates misclustering errors between the true cluster labels and predicted cluster labels. The confusion matrix $C(\widehat{y}, y)$ is a matrix of size $k \times k$ tabulating the correct number of clustering in each cluster. The diagonal entries show the correct number of clustering while the off-diagonal entries show the incorrect number of clusters. In Matlab, it can be computed using `confusionmat.m`:

```
ytrue = [ 1 1 1  2 2 3 3]
ypred = [ 1 1 2  1 1 3 3]
C = confusionmat(ypred, ytrue)

C =
     2     2     0
     1     0     0
     0     0     2
```

Alternately, we can compute the confusion matrix by simply counting the number of correct clustering:

```
C=zeros(k);
n=length(ytrue);
for i=1:n
    C(ypred(i),ytrue(i))=C(ypred(i),ytrue(i))+1;
end
```

To compute the clustering accuracy, we need to sum the diagonal entries. But the above matrix `C` is one possible confusion matrix. Under the permutation of cluster labels, we can get different confusion matrices. For large k, it is prohibitive

expensive to search for all permutations. Thus we need to maximize the sum of diagonals of the confusion matrix under permutation with weight $C = (c_{ij})$:

$$\frac{1}{n} \max_{Q \in \mathbb{S}_k} \operatorname{tr}(QC) = \frac{1}{n} \max_{Q \in \mathbb{S}_k} \sum_{i,j} q_{ij} c_{ij}, \tag{1}$$

where $Q = (q_{ij})$ is the permutation matrix consisting of entries 0 and 1 such that there is exactly single 1 in each row and each column. This is a linear sum assignment problem (LSAP), a special case of linear assignment problem (Bougleux & Brun 2016). LSAP is solved using `matchpairs.m` in Matlab (Duff & Koster 2001):

```
M=matchpairs(C, 0, 'max');

M =
     2     1
     1     2
     3     3

accuracy = sum(C(sub2ind(size(C), M(:,1), M(:,2))))/n

accuracy=
   0.7143
```

The whole procedure is packaged into a single Matlab function `clustering_accuracy.m`, which can be downloaded from http://pages.stat.wisc.edu/~mchung/dynamicTDA/matlab/clustering_accuracy.m.

4 Cluster Distances

Distance-based clustering is one of the most fundamental approaches in unsupervised learning, where data points are grouped based on a predefined distance or similarity. The objective is to form clusters such that points within the same cluster are more closer to each other than to points in different clusters.

For now, we assume a sufficiently large number of data points $n \gg k$ with a fixed number of clusters k. Further, we assume that each cluster contains at least one data point. Assume we are able to compute the a set of *distances* $d(i, j) = d_{ij}$ between points i and j somehow. We further assume $d_{ii} = 0$. Ignoring symmetry, the within-cluster distance $\mathcal{L}_W$ and the between-cluster distance $\mathcal{L}_B$ are defined as (Songdechakraiwut & Chung 2023):

$$\mathcal{L}_W(C_1, \dots, C_k) = \sum_{l=1}^{k} \sum_{i,j \in C_l} d_{ij},$$

$$\mathcal{L}_B(C_1, \dots, C_k) = \sum_{l \neq m} \sum_{i \in C_l, j \in C_m} d_{ij}.$$

The total pairwise distance is then given by:

$$\mathcal{L} = \sum_{i,j} d_{ij} = \mathcal{L}_W + \mathcal{L}_B.$$

To determine clusters $C_1, \ldots, C_k$, we either maximize $\mathcal{L}_B$ or minimize $\mathcal{L}_W$, which are equivalent formulations.

Denote the number of data points in the l-th cluster C_l as $|C_l|$. The total number of unique pairs within a cluster is $\binom{|C_l|}{2}$. The mean intra-cluster distance μ_l for cluster C_l is given by:

$$2\binom{|C_l|}{2}\mu_l = \sum_{i,j \in C_l} d_{ij}. \tag{2}$$

The factor of 2 accounts for symmetry $d_{ij} = d_{ji}$. Thus, the within-cluster loss can be rewritten as:

$$\mathcal{L}_W = \sum_{l=1}^{k} |C_l|(|C_l| - 1)\mu_l. \tag{3}$$

5 Online Algorithm for Clustering

The recurrence relation for the Catalan number suggests the possibility of an online algorithm for clustering. Consider an optimal partition $\phi_k \in \boldsymbol{\Phi}_n$ that partitions n data points $1, \ldots, n$ into k clusters $C_1, \ldots, C_k$ such that

$$\phi_k = \arg\max_{\phi \in \boldsymbol{\Phi}_n} \mathcal{L}_B = \arg\min_{\phi \in \boldsymbol{\Phi}_n} \mathcal{L}_W. \tag{4}$$

Maximizing the between-cluster loss is equivalent to minimizing the within-cluster loss. However, directly optimizing such a loss function is computationally challenging. Therefore, we adopt an online strategy for incremental clustering by analyzing how the within-cluster loss changes as new data points are added.

The addition of a data point a to cluster C_q increases the within-cluster loss as

$$\mathcal{L}_W(C_1, \ldots, C_q \cup \{a\}, \ldots, C_k) = \mathcal{L}_W(C_1, \ldots, C_k) + 2\sum_{j \in C_q} d_{aj}. \tag{5}$$

Similarly, the removal of a data point a from cluster C_p decreases the within-cluster loss as

$$\mathcal{L}_W(C_1, \ldots, C_p \setminus \{a\}, \ldots, C_k) = \mathcal{L}_W(C_1, \ldots, C_k) - 2\sum_{j \in C_p} d_{aj}.$$

Thus, reassigning point a from cluster C_p to C_q changes the loss by

$$\Delta_{p \to q}\mathcal{L}_W = 2\sum_{j \in C_q} d_{aj} - 2\sum_{j \in C_p} d_{aj}.$$

This provides gradient information to determine whether the within-cluster distance increases or decreases with cluster assignment, enabling efficient updates in linear time.

For online clustering, we sequentially assign one data point to an existing cluster at a time using equation (5). The algorithm proceeds as follows:

1. Randomly select k data points from n and assign each to an initial cluster $C_1, \ldots, C_k$.
2. For each of the remaining $n - k$ data points, pick a point a at random and assign it to the cluster that results in the smallest within-cluster distance using equation (5).
3. Repeat step 2 until all data points have been assigned to clusters.
4. Restart the process with different randomly selected initial cluster points and repeat multiple times to obtain different clustering results.
5. Select the clustering configuration that achieves the minimum within-cluster distance.

A similar online update (called transposition) is used in updating the ratio statistic of within distance over between distance in an ANOVA setting (Songdechakraiwut & Chung 2023).

Bibliography

Bougleux, S. & Brun, L. (2016), 'Linear sum assignment with edition', *arXiv preprint arXiv:1603.04380* .

Chung, M., Xie, L., Huang, S.-G., Wang, Y., Yan, J. & Shen, L. (2019), 'Rapid acceleration of the permutation test via transpositions', *International Workshop on Connectomics in Neuroimaging* **11848**, 42–53.

Duff, I. & Koster, J. (2001), 'On algorithms for permuting large entries to the diagonal of a sparse matrix', *SIAM Journal on Matrix Analysis and Applications* **22**(4), 973–996.

Edmonds, J. & Karp, R. (1972), 'Theoretical improvements in algorithmic efficiency for network flow problems', *Journal of the ACM (JACM)* **19**, 248–264.

Feller, W. (2008), *An introduction to probability theory and its applications*, Vol. 2, John Wiley & Sons.

Simion, R. (2000), 'Noncrossing partitions', *Discrete Mathematics* **217**(1-3), 367–409.

Songdechakraiwut, T. & Chung, M. (2023), 'Topological learning for brain networks', *Annals of Applied Statistics* **17**, 403–433.