

## Clustering

Now begins our foray into unsupervised learning. Most unsupervised learning problems deal only with *input* data, typically considered as a matrix  $X \in \mathbb{R}^{N \times D}$  over  $N$  data points and  $D$  dimensions. There are no labels.

Clustering is the problem of grouping data according to similarity.

Typically one uses a *distance metric* to judge similarity (though this is not always required). A metric  $d$  must satisfy:

1.  $d(x, y) = 0$  iff  $x = y$  – isolation
2.  $d(x, y) = d(y, x)$  – distance is symmetric
3.  $d(x, y) + d(y, z) \geq d(x, z)$  – triangle inequality

If (1) doesn't hold, it is a pseudo-metric.

All of our favorite norms *are* metrics. Non-real metrics include things like string edit distance.

Basically two types of clustering:

1. Hierarchical
2. Flat

Hierarchical tries to build a tree over the data ( $N$  leaves). Typically binary. Flat just tries to find  $K$  clusters of the data.

### Hierarchical Clustering

Bottom-up strategy:

1. Put every element into its own cluster
2. Merge two most similar clusters
3. If  $> 1$  cluster exists, go to (2)

How to measure “similar.” For individual data points, just use the metric  $d$ . What about  $\text{sim}(\{x_1, x_2\}, \{x_3, x_4, x_5\})$ ?

1. Min-link (aka “single link”):

$$\text{sim}(R, S) = \min_{x_R \in R, x_S \in S} d(x_R, x_S)$$

2. Max-link (aka “complete”):

$$\text{sim}(R, S) = \max_{x_R \in R, x_S \in S} d(x_R, x_S)$$

3. Average-link:

$$\text{sim}(R, S) = \frac{1}{|R||S|} \sum_{x_R \in R, x_S \in S} d(x_R, x_S)$$

These have very different properties. Min-link results in chaining. Max-link results in very “round” clusters. Average-link is an intermediary.

Top-down strategy:

1. Put every element into a single cluster
2. Remove the “outsider” (or outsiders) from the group
3. If a cluster of size  $> 1$  exists, go to (2)

Same story as bottom up with linkage. Here, if we remove a single element, we get a funny tree. If we want to split clusters in half, it can be *very* computationally expensive. Typically bottom-up is preferred.

### Flat Clustering

Want to break  $N$  elements into  $K$  groups. How to choose the groups?

1. Want high intercluster similarity:  $\sum_k \sum_{n \neq m \in k} d(x_n, x_m)$  should be low.
2. Want low intracluster similarity:  $\sum_{k \neq k'} \sum_{n \in k} \sum_{m \in k'} d(x_n, x_m)$  should be high.

Too hard to optimize a linear combination of these... there are  $\frac{1}{K!} \sum_k (-1)^{K-k} \binom{K}{k} k^N$  possible clusters (for  $N = 19, K = 4$ , this is over  $10^{10}$ ).

Idea: propose clusters, then iteratively fix errors.

1. Initialize cluster centers
2. Assign each data point to the closest center
3. Recompute the centers as the means of the data points
4. If assignments have changed, go to (2)

This is the  $k$ -means algorithm. It is highly sensitive to initialization, but converges quite quickly. (There are various ways to speed it up, as well.)

Initializing means:

1. Random points drawn from a “reasonable” input space
2. Random data points
3. First a random data point, then a second data point as far away as possible, then a third data point again as far away as possible, etc.

In general, it’s a good idea to do a few runs of  $k$ -means.

$k$ -medioids is similar, but forces each centroid to be a data point. This is advantageous for non-real-valued data.

---

**Choosing the number of clusters**

1. Prior knowledge of the domain
2. Visual inspection (in low  $D$ )
3. Try a bunch and use them somehow
4. Look for the “elbow” in the  $k$  versus  $\text{int}(er/ra)$  cluster similarity