

## Affinity Propagation

### Definition and purpose

**Affinity propagation (AP)** is a clustering algorithm based on the concept of "message passing" between data points. **It does not require the number of clusters to be determined or estimated before running the algorithm (differ from the prototype-based clustering algorithms).**

At first, we need to define the similarity between node  $x_i$  and  $x_j$ , and it should satisfy that  $S(i, j) > S(i, k)$  if and only if  $x_i$  is more similar to  $x_j$  than to  $x_k$ . So, we use the negative squared distance of two data points, which as

$$S(i, k) = -||x_i - x_k||^2$$

**The diagonal of  $S$  (i.e.,  $S(i, i)$ )** is a hyper-parameter which represents the **instance preference**, meaning how likely a particular instance is to become an exemplar(prototype). **When it is set to the same value for all inputs, it controls how many classes the algorithm produces.** Smaller value produces fewer classes, vice versa. **We usually use the minimum.**

$$S(i, i) = \min\{S(i, k)\}$$

The Process is

1. **Each node, initially, is a cluster**
2. At each round, **we merge any cluster to the cluster that is closest to it**
3. We stop the algorithm when we have the desired number of clusters

The algorithm proceeds by alternating between two message-passing steps, which updates two matrices. Initially, both matrices are set to 0, For all  $i, k$ , let:

1. The "Responsibility"  $r_{i,k}$  "sent" from  $i$  to  $k$

$$r(i, k) \leftarrow s(i, k) - \max_{k' \neq k} \{a(i, k') + s(i, k')\}$$

2. The "Availability"  $a_{i,k}$  "sent" from candidate  $k$  to point  $i$

$$a(i, k) \leftarrow \min \left( 0, r(k, k) + \sum_{i' \notin \{i, k\}} \max(0, r(i', k)) \right) \text{ for } i \neq k \text{ and}$$

$$a(k, k) \leftarrow \sum_{i' \neq k} \max(0, r(i', k)).$$

## Examples

if we have the following data

**Table 1: Preferences of Five Participants**

Participant	Tax Rate	Fee	Interest Rate	Quantity Limit	Price Limit
Alice	3	4	3	2	1
Bob	4	3	5	1	1
Cary	3	5	3	3	3
Doug	2	1	3	3	2
Edna	1	1	3	2	3

Treat every node as one cluster, and calculate the similarity matrix using  $S(i, k) = -||x_i - x_k||^2$

**Table 2: Similarity Matrix**

Participant	Alice	Bob	Cary	Doug	Edna
Alice	-22	-7	-6	-12	-17
Bob	-7	-22	-17	-17	-22
Cary	-6	-17	-22	-18	-21
Doug	-12	-17	-18	-22	-3
Edna	-17	-22	-21	-3	-22

Then, calculate the responsibility matrix using  $r(i, k) = s(i, k) - \max\{a(i, k) + s(i, k')\}$ . For example, the responsibility of Bob (column) to Alice (row) is -1, which is the similarity of Bob to Alice (-7) minus the maximum of the remaining similarities of Alice's row (-6). For my personal, **the responsibility represents “how this point compares to the best choice from others to the row”**

**Table 3: Responsibility Matrix**

Participant	Alice	Bob	Cary	Doug	Edna
Alice	-16	-1	1	-6	-11
Bob	10	-15	-10	-10	-15
Cary	11	-11	-16	-12	-15
Doug	-9	-14	-15	-19	9
Edna	-14	-19	-18	14	-19

Next to update the availability matrix. **We use a separate equation for updating the elements on the diagonal of the availability matrix** than we do the elements off the diagonal of the availability matrix.

The proceeding formula is used to fill in the elements on the diagonal:

$$a(k, k) \leftarrow \sum_{i' \text{ such that } i' \neq k} \max\{0, r(i', k)\}, \quad (2)$$

where i refers to the row and k the column of the associated matrix. In essence, the equation is telling you to **sum all the values above 0 along the column except for the row whose value is equal to the column in question**. For example, the self-availability of Alice is the sum of the positive responsibilities of Alice's column excluding Alice's self-responsibility (10 + 11 + 0 + 0 = 21).

The following equation is used to update off-diagonal elements:

$$a(i, k) \leftarrow \min \left\{ 0, r(k, k) + \sum_{i' \text{ such that } i' \notin \{i, k\}} \max\{0, r(i', k)\} \right\} \quad (3)$$

In other words, say you're trying to fill in a(Cary, Edna). Considering the elements along Edna's column, you exclude the Edna/Edna relationship and the Cary/Edna relationship and sum all the remaining positive responsibilities together. For example, the availability of Bob (column) to

Alice (row) is Bob's self-responsibility plus the sum of the remaining positive responsibilities of Bob's column excluding the responsibility of Bob to Alice ( $-15 + 0 + 0 + 0 = -15$ ).

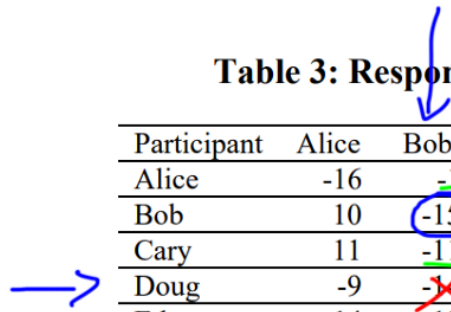
**Table 3: Responsibility Matrix**

Participant	Alice	Bob	Cary	Doug	Edna
Alice	-16	-1	1	-6	-11
Bob	10	-15	-10	-10	-15
Cary	11	-11	<del>1</del>	-12	-15
Doug	-9	-14	-15	-19	9
Edna	-14	-19	-18	14	<del>9</del>

$$\text{Cary: } 1 + 0 + 0 + 0 = 1$$

$$\text{Edna: } 0 + 0 + 0 + 9 = 9$$

**Table 3: Responsibility Matrix**



Participant	Alice	Bob	Cary	Doug	Edna
Alice	-16	-1	1	-6	-11
Bob	10	-15	-10	-10	-15
Cary	11	-11	-16	-12	-15
Doug	-9	<del>-14</del>	-15	-19	9
Edna	-14	-19	-18	14	-19

$$-15 + 0 + 0 + 0 = -15$$

$$A[\text{Doug}, \text{Bob}] = -15$$

After calculating the rest, we wind up with the following availability matrix. **For my personal, the diagonal  $A(i, i)$  show if we select this point ( $i$ ) as prototype, how many "variances / distance" we could save.**

**Table 4: Availability Matrix**

Participant	Alice	Bob	Cary	Doug	Edna
Alice	21	-15	-16	-5	-10
Bob	-5	0	-15	-5	-10
Cary	-6	-15	1	-5	-10
Doug	0	-15	-15	14	-19
Edna	0	-15	-15	-19	9

### Criterion Matrix (c)

Each cell in the criterion matrix is simply the sum of the availability matrix and responsibility matrix at that location. The criterion value of Bob (column) to Alice (row) is the sum of the responsibility and availability of Bob to Alice ( $-1 + -15 = -16$ ).

$$c(i, k) \leftarrow r(i, k) + a(i, k). \quad (4)$$

## Table 5: Criterion Matrix

Participant	Alice	Bob	Cary	Doug	Edna
Alice	<b>5</b>	-16	-15	-11	-21
Bob	<b>5</b>	-15	-25	-15	-25
Cary	<b>5</b>	-26	-15	-17	-25
Doug	-9	-29	-30	<b>-5</b>	-10
Edna	-14	-34	-33	<b>-5</b>	-10

The highest criterion value of each row is designated as the **exemplar**. Rows that share the same exemplar are in the same cluster. Thus, in our example. Alice, Bob, and Cary form one cluster whereas Doug and Edna constitute the second.

### Properties

Although the algorithm is **computationally very extensive**, it resolves the issue where the **prototype is one of the data points**, rather than a hypothetical point.

It improves upon Agglomerative Clustering and **prevents putting points that are too far apart into the same group**.