CPSC 320: Clustering Solutions (part 1) [*]

You're working on software to organize people's photos. Your algorithm receives as input:

- A bunch of uncategorized photos.
- A *similarity measure* for each pair of photos, where a 0 similarity indicates two photos are nothing like each other; a 1 indicates two photos are exactly the same. All other similarities are in between.
- The number of categories to group them into.

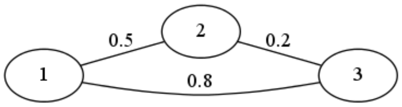
Your algorithm should create a *categorization*: the requested number of categories, where a category is a non-empty set of photos. Every photo belongs to some category, and no photo belongs to more than one category. So, a categorization is a *partition*. We'd like similar photos to be in the same category.

Step 1: Build intuition through examples.

1. Write down small and trivial instances of the problem. What data structure is useful to represent a problem instance? Write down also potential solutions for your instances. Are some solutions better than others? How so?

   SOLUTION: An empty graph (with zero categories) is clearly trivial, though it's not clear this should be counted as an instance. Any instance with one category is also trivial because every photo must go in that category. At the other end of the spectrum, if the number of photos equals the number of categories, then each photo must go in its own category, which is trivial. You might also find instances with all similarities equal or other interesting cases to be trivial.

   A small nontrivial instance could have three photos and two categories. We'll use a weighted graph representation, where nodes correspond to photos and edge weights represent similarities. We also want the graph to be complete—to contain every possible edge (except self-loops). Here is an example:

   # categories = 2

   For this small instance, a solution that groups 1 and 3 together, and leaves 2 in its own category, seems better than the alternatives.

Step 2: Develop a formal problem specification

1. Develop notation for describing a problem instance.

   SOLUTION:

   We'll let n be the number of photos. It's reasonable to assume that $n \geq 1$. We can think of an instance as a graph $G = (V, E)$ where each node in V is a photo. For simplicity, let's assume that the photos are numbered, so V is just the set $\{1, 2, \ldots, n\}$. For every pair $p, p' \in V$ there is an undirected,

[*]

weighted edge of the form (p, p', s) in E with $0 \leq s \leq 1$. The edge similarities could be represented as a 2D matrix, say S, where $S[p][p']$ is the similarity s of the pair (p, p') of photos.

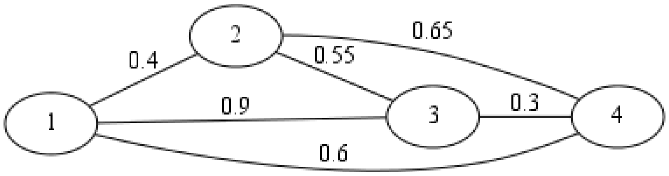
An instance can then be n, the set E, plus a number, say c, of categories, where $1 \leq c \leq n$.

For example, the small instance above has $n = 3$ and $c = 2$, and is represented as:

$$(3, \{(1, 2, 0.5), (1, 3, 0.8), (2, 3, 0.2)\}, 2).$$

2. Use your notation to flesh out the following group of photos into an instance.

SOLUTION: There are many reasonable solutions. Here's one. We simply chose plausible similarities; the correspondence between images and numbers is not too important. We'll choose $c = 2$.



$c = 2$

It's hard to eyeball a solution to this with no metric to guide us, but it does seem that images 1 and 3 should be together given their strong similarity (0.9). We might then put images 2 and 4 together.

3. Develop notation for describing a potential solution. Describe what you think makes a solution *good*. Can you come up with a reasonable criterion for deciding if one solution is better than another?

SOLUTION: A potential solution (or just "solution") will be a set of categories $\{C_1, C_2, \ldots, C_c\}$, where a category C_i is a set of nodes from V. This set should be a partition. That is, every photo must belong to one and only one category, and every category C_i is non-empty.

It's much harder to say what a **good** solution is. Clearly we want to reward having nodes with high similarity in the same category and penalize having nodes with low similarity in the same category. If we divide edges into *intra-category* (between nodes in the same category) and *inter-category* (between nodes in different categories), then we might choose a metric like "sum of the intra-category similarities minus sum of the inter-category similarities". However, this will push us toward large categories. (Since the number of intra-category edges in a category C scales as $O(|C|^2)$, there are more intra-category edges in a categorization with one big category and many small ones than with all even-sized categories (and the same total number of categories).)

We could instead look for a similarity measure based on the **average** similarity of each category. We have to decide then what to do with categories of a single node, since their average is undefined. Rate them zero?

Note that there's no "right" choice. We just have to decide what does a good job modeling what we care about and how efficient a solution we get. There's something fundamental about the fact that we will soon pick a metric that's totally reasonable. . . and which we really chose because it is "good enough" and admits a highly efficient solution. How much of our lives are now ruled by metrics that are easy to compute rather than "best"?

4. From here on, we'll all use the same definition of "good solution".

First, we define the similarity between two categories C_1 and C_2 to be the maximum similarity between any pair of photos p_1, p_2 such that $p_1 \in C_1$ and $p_2 \in C_2$.

Then, the *cost* of a categorization is the maximum similarity between any two of its categories. The lower the cost, the better the categorization, since we we don't **want** categories to be similar. So, we want to find a solution with minimum cost. We'll use the term "optimal solution" (rather than "good solution") to refer to solutions that have minimum cost.

Write down optimal solutions and their costs for your previous examples.

SOLUTION: The 4-node, 2-cluster problem's solution with categories $\{(\{1,3\},\{2,4\}\}$ has a cost of 0.6, defined by the edge between 1 and 4. The 3-node, 2-cluster problem's solution has a cost of 0.5, defined by the edge between nodes 1 and 2.

Step 3: Identify similar problems. What are the similarities?

SOLUTION: There are certainly similarities here to shortest path, minimum spanning tree, matching, and maximal matching. As we'll see, perhaps the most promising similarity is to MST!

Step 4: Evaluate brute force.

1. A potential solution is the set of partitions of n photos into c subsets (where c is the requested number of categories). Suppose that $c = 2$. Roughly, how does the number of potential solutions grow asymptotically with n? Polynomially? Exponentially?

   SOLUTION: One way to think about this is that when $c = 2$, then some subset, say A, of the photos go in one category, and the remaining photos (i.e., those in $V - A$) go in the other category. How many subsets A of n photos are there? A set of size n has 2^n subsets. However, A cannot be either the empty set or the set of all photos, since both categories must be non-empty. So, there are $2^n - 2$ possible choices for A. Finally, we consider the pairs of categories $A, V - A$ and $V - A, A$ to be the same (order of categories doesn't matter), so we need to divide our count by 2. Thus the number of potential solutions when $c = 2$ is $(2^n - 2)/2 = 2^{n-1} - 1$, which grows exponentially with n.

2. Given a potential solution, how can you determine how good it is, i.e., what is its cost? Asymptotically, how long will this take?

   SOLUTION: We would need to find the intercategory edge with the highest similarity; this would take $O(n^2)$ time. For each pair of nodes, check if they're in the same category. (A matrix $C[1..n]$, where $C[i]$ is the number of the category containing photo i, makes this check possible in $O(1)$ time.) If they're not in the same category, check if their similarity exceeds the highest we've found so far, updating if necessary.

Step 5: Design a better algorithm.

There is a **much** better approach.

1. Find the edge in each of your instances with the highest similarity. Should the two photos incident on that edge go in the same category? Prove a more general result.

   SOLUTION:

   We can show that there is an optimal solution in which the two photos incident on the edge with highest similarity are placed in the same category.

To see why, let's revisit our cost measure. It ignores intra-category edges: any edge that's entirely contained within a single category (i.e., both nodes incident on the edge are in the same category). Its goal is to minimize the maximum *inter*-category edge.

If the two photos incident on the highest similarity edge are in different categories of a solution, that edge becomes an inter-category edge. That means the solution is (tied for) **the worst possible solution**. Any solution in which these two photos are in the same category has to be at least as good if not better.

What we're really saying is "start with every node being its own category, then you're best off merging the two categories connected by the highest-similarity edge".

2. Based on this insight, come up with algorithmic ideas for creating a categorization.

   SOLUTION: We can build on the insight to create a categorization as follows. We start with every photo in its own category. We then put the two photos with the highest-similarity edge in one category, effectively "merging" two of the initial categories. We continue to merge categories, choosing two to merge if the highest inter-category edge connects them. We stop when we are down to c categories. The following pseudocode captures this idea.

```
function CLUSTERING-GREEDY($n, E, c$)
▷ $n \geq 1$ is the number of photos
▷ $E$ is a set of edges of the form $(p, p', s)$, where $s$ is the similarity of $p$ and $p'$
▷ $c$ is the number of categories, $1 \leq c \leq n$
    create a list of the edges of $E$, in decreasing order by similarity
    let $\mathcal{C}$ be the categorization with each photo in its own category
    Num-$\mathcal{C} \leftarrow n$            ▷ Initial number of categories
    while Num-$\mathcal{C} > c$ do
        remove the highest-similarity edge $(p, p', s)$ from the list
        if $p$ and $p'$ are in different categories of $\mathcal{C}$ then
            "merge" the categories containing $p$ and $p'$
            Num-$\mathcal{C} \leftarrow$ Num-$\mathcal{C} - 1$
    return $\mathcal{C}$
```