Whispers Algorithm

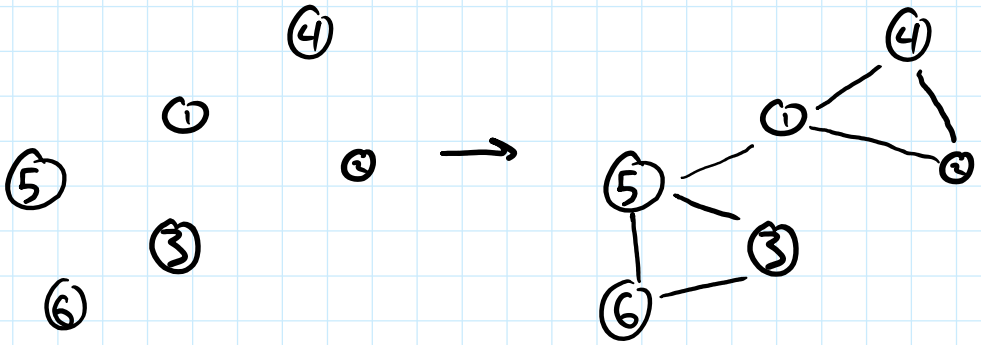
Thursday, July 20, 2017 11:44 AM

Given a bunch of unlabelled pictures of individuals, can we group the photos based on who is in them?

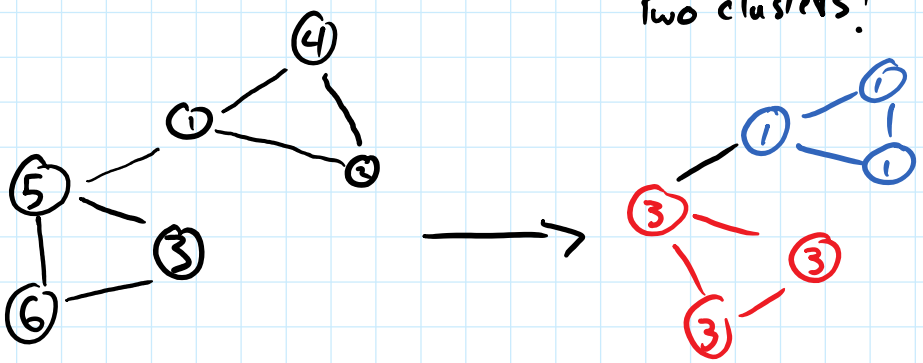
		initial label
Image 1	descriptor 1	1
Image 2	descriptor 2	2
Image 3	descriptor 3	3
⋮		⋮
N		N

We have N unlabeled images. We will think of these as nodes in a graph. At the outset, each node has its own label

We want to figure out how to determine which nodes in our graph should have edges between one another.



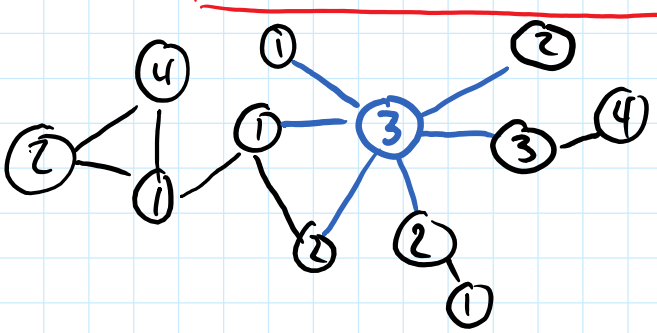
Then, we want to use the "whispers" method to propagate node labels, this will allow us to identify the "clusters" in our graph



Label propagation: whispers algorithm

This is a simple but effective way to propagate labels and 'discover' clusters in your graph.

Pick a random node.



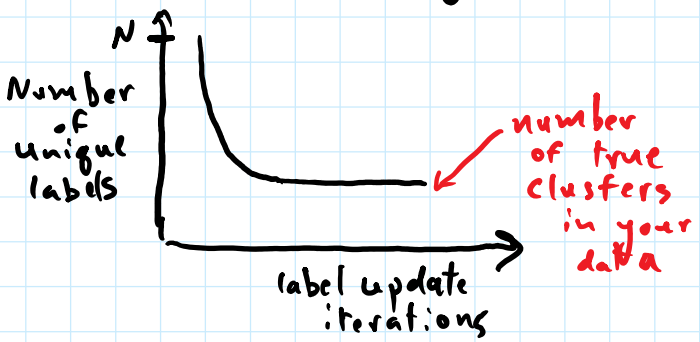
Count up all of the labels of its neighbors

label	count
1	2
2*	3* → Highest count
3	1
4	0

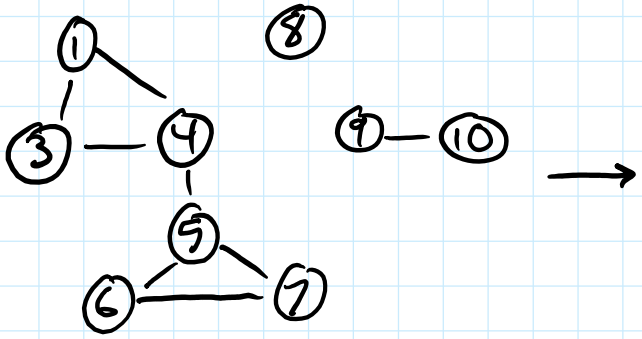
The label with the highest count is adopted as the new label for that node

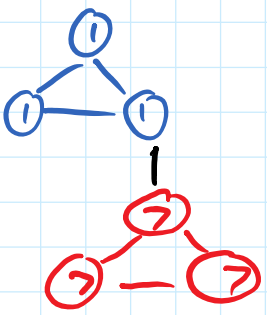
If there is a tie in counts, pick the new label randomly from the tied labels

Repeat this process and keep track of the number of unique labels in your graph. At first you start with N labels → a unique one for each node. This number will decrease and eventually converge



Once this process has converged, the nodes with the same label belong to the same cluster.





4 distinct clusters

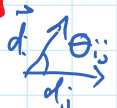
Picking Edges

We do not want to draw edges between all pairs of nodes.

Instead, we want edges between nodes that we are confident are related — their descriptor vectors are close to each other.

We also want to include edges between nodes whose relationship is questionable

Take labeled images, and compute the cosine distance between all of them.



$$\text{cosine distance} = 1 - \frac{\vec{d}_i \cdot \vec{d}_j}{\|\vec{d}_i\| \|\vec{d}_j\|} = 1 - \theta_{ij}$$

Ryan-1 Melanie-1
Ryan-2 ⟷ Melanie-2 . . .
Ryan-3 :
:

should have large distances between each other

↑ should have small distances between each other

Plot a histogram of all distances

Use this histogram to pick a good distance threshold:

> two nodes whose descriptor distance is within this threshold should have an edge between them

---

Improving the whispers algorithm (Weighted Edges)

Because we know that two nodes with a small distance between their descriptors are likely

more related than two that are far apart, we can include this in the "whisper" portion of our label propagation

We can add a "weight" to each edge: nodes with a small distance should have a large edge weight

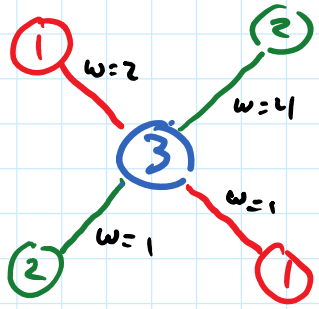
Instead of tallying up the labels, we tally up the weights. The label with the largest total weight is adopted by the node

Suppose $w(d) = \frac{1}{d^2}$ small distance → large weights

d: cosine distance between descriptors

w: edge-weight

If two nodes have a distance 0.1 between them, their edge will have a weight: $\frac{1}{0.1^2} = 100$



label	total weight
1	2+1 = 3
2	4+1 = 5* → winner!

This will make our label propagation much more accurate, since we are providing our understanding of the descriptor distances

---

How do I code up a "graph"

There are a couple of simple ways to summarize our graph.

First, what does a node need? | see node.py for

First, what does a node need?

- ID
- label
- neighbor list
- truth?
- file-path of image?

→ make this a class!

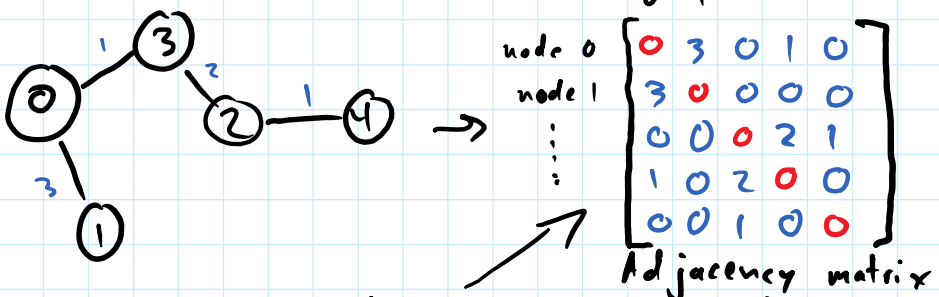
see node.py for example implementation!

These nodes store all of the connection info. Our "graph" can simply be a list of them:

graph = [Node_0, Node_1, ... Node_N-1]

We also need to be able to look up our edge weights. We can do this using an "adjacency matrix"

Consider the graph w/ weighted edges and 5 nodes in total:



Adjacency matrix

Element i, j is the weight of the edge between

This is a very easy way to keep track of how your nodes are connected, and the weights of their edges