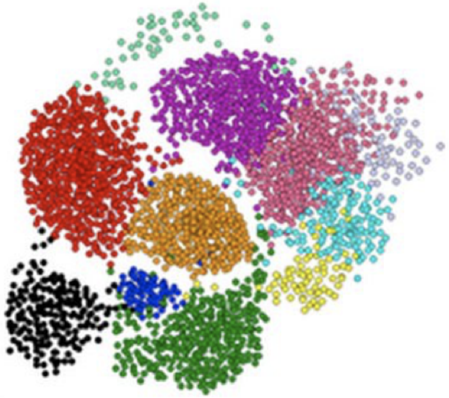
CSCI 4022 Fall 2021
Minhash Wrapup; Clustering



Announcements and To-Dos

Announcements:

1. HW 1 up!

Some homework hints and discussion follow:

2. The version that's posted lost a comma in the "rotate" function. See Piazza for a fix.
3. You should not need any formulas from your hypothesis testing sections of your intro course. Instead you're using simulations to approximate probabilities: probabilities such as falsely classifying a score as "unreasonably unlucky" or "normal".

Last time Recap: minhashing

Minhashing was a method to *approximate* Jaccard similarities by sharing the ratio of "both 1" rows to the ratio of "at least one 1" rows from the original matrix.

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
6	0	0	1
8	1	1	0
13	1	0	0
16	0	1	1

Definition: To *minhash* a set represented by a column of the characteristic matrix, pick a permutation of the rows. The *minhash value* of any column is the number of the first row, in the permuted order, in which the column has a 1.

Definition: A row in the *signature matrix* is built up from the minhash values of all columns under a given permutation

Idea: Two columns share a minhash value with probability proportionate to their shared "1's" versus either having a "1"... also known as their Jaccard similarity.

Permuting

So we're at this point:

1. Collapse documents into smaller matrices using shingles, and possible hashing to reduce the total number of rows.
2. Permuting the rows around allows us to save on *memory*. Instead of loading all the documents at once, we can create their *signatures*, or the first first row with a "1" under a given permutation. We then apply the *same permutation* to other documents and compare *signatures*

But did this actually save memory? Now we have to save permutations of huge matrices to use the same permutations on new documents!

Worse... aren't truly random permutations pretty computationally expensive??

Not Permuting

It turns out we don't *actually have to permute*. Instead, another hash function allows us to **approximate** permutations. If we use a hash function we can randomly grab rows in a way that *looks* like a random permutation.

This is called a **universal hash**. For *random* integers a and b and a large prime number p (much greater than the total number of rows N,

14 x +2 mod 7

2 mod 7

$$h_{a,b}(x) = ((\boxed{a}\, x + \boxed{b}) \mod p) \mod N$$

18 mod 7

30 mod 7

is a function that in practice *scans the rows* of the **characteristic matrix** in an approximately random order.

(NB: alternatively, you can *fix* that the number of rows of the characteristic matrix N is prime, and just use $(a \cdot x + b) \mod N$. We choose a, b to be less than N, mostly to ensure that they don't end up being multiples of N.)

The full minhash:

min $= \infty$ $[4 / 8 / 9 / 10 / 2 / 1 / 7]$

4 4 4 4 2 1 1

return $\boxed{1}$

1. Step 1: pick n hash functions.
2. Step 2: initialize the *signature matrix* with all infinities. We'll replace the terms once we find "first 1's".
3. Step 3: For each row r of the characteristic matrix, compute $h_i(r)$ for that row for each and every one of the i hash functions. This represents "final permuted location of the original row r" for n different permutations.
4. Step 4: For each document or column x:
    4.1 If the char. matrix has a 0 in column c, row r: do nothing.
    4.2 If the char. matrix has a 1 in column c, row r, then take each of the hash functions and replace $sig(i,c)$ by $\min(sig(i,c), h_i(r))$

Why this last bit? We're looking for the *first row* of a 1, which will be the minimum time that it occurred.

q = np.random.choice(range(11)+1)

The full minhash: Example

So here's our data, with row labels now indices:

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

Step 1:
Suppose we choose a few hash functions. We use a prime of 11 since it's more than the buckets we have (8). Our functions: a r + b

$h_1(r) = (r + 3) \mod 11$ 0+3 mod 11 → row 0 ended @ row 3

$h_2(r) = (2r + 5) \mod 11$ 0+5 mod 11 row 5

$h_3(r) = (3r + 7) \mod 11$ 0+7 mod 11 row 7

Step 2: Initialize: where was the first 1?
We set *sig*=

$$\begin{array}{cc} & \begin{array}{ccc} banana & bandit & brand \end{array} \\ \begin{array}{c} \text{row 3} \quad h_1 \\ \text{row 5} \quad h_2 \\ \text{row 7} \quad h_3 \end{array} & \left[\begin{array}{ccc} \infty, 3 & \infty \; 3 & \infty \; 3 \\ \infty, 5 & \infty \; 5 & \infty \; 5 \\ \infty, 7 & \infty \; 7 & \infty \; 7 \end{array}\right] \end{array}$$

The full minhash: Example

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$$h_1(r) = (r+3) \mod 11$$

$$h_2(r) = (2r+5) \mod 11$$

$$h_3(r) = (3r+7) \mod 11$$

Step 3: For each row r of the characteristic matrix, compute $h_i(r)$ for that row for each and every one of the i hash functions. This represents "pick a random row" for n different permutations.

Step 4: For each column:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1 in column c, row r, then take each of the hash functions and replace $sig(i,c)$ by $\min(sig(i,c), h_i(r))$

Solution: in row 0, only *bandit* has a 1. It's eligible for replacement. **The minimum "location"** we've observed a non-zero for *bandit* is at the hashed value of "3".

$$\begin{array}{cccc} & banana & bandit & brand \\ h_1 & \infty & \infty & \infty \end{array}$$

minhash Example, row 0

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$$h_1(r) = (r+3) \mod 11$$
$$h_2(r) = (2r+5) \mod 11$$
$$h_3(r) = (3r+7) \mod 11$$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1(0) = 3,\ h_2(0) = 5,\ h_3(0) = 7$

Step 4: For each document or column x:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1 in column c, row r, then take each of the hash functions and replace $sig(i, c)$ by $\min\left(sig(i, c), h_i(r)\right)$

Solution: in row 1, all docs have a 1. For all 3 hash fns and all 3 docs, we have an estimate for the "first location of a 1."

$$\begin{array}{c|ccc} & banana & bandit & brand \\ h_1 & 3 & 3 & 3 \\ h_2 & 5 & 5 & 5 \\ h_3 & 7 & 7 & 7 \end{array}$$

minhash Example, row 1

	banana	bandit	brand
0	1	1	1
1	$\cancel{0}$	1	$\cancel{0}$
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$h_1(r) = (r+3) \mod 11$

$h_2(r) = (2r+5) \mod 11$

$h_3(r) = (3r+7) \mod 11$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1\boxed{(1)} = 4,\ h_2\boxed{(1)} = 7,\ h_3\boxed{(1)} = 10$

Step 4: For each document or column x:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1, replace $sig(i,c)$ by $\min\left(sig(i,c), h_i(r)\right)$

Solution: in row 1, only *bandit* has a 1. It's eligible for replacement.

$$\begin{array}{c|ccc} & \cancel{banana} & bandit & \cancel{brand} \\ \hline h_1 & \cancel{3} & 3\ vs.\ 4 & \cancel{3} \\ h_2 & \cancel{5} & 5\ vs.\ 7 & \cancel{5} \\ h_3 & \cancel{7} & 7\ vs.\ 10 & \cancel{7} \end{array}$$

[0, . . . , 10]

minhash Example, row 1

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$h_1(r) = (r+3) \mod 11$

$h_2(r) = (2r+5) \mod 11$

$h_3(r) = (3r+7) \mod 11$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1(1) = 4$, $h_2(1) = 7$, $h_3(1) = 10$

$h_1(2)$ $h_2(2)$ $h_3(2)$

Step 4:
Solution: in row 1, only *bandit* has a 1. It's eligible for replacement, but its current values of 3,5,7 are smaller than the current evaluations of 4, 7, 10. **The minimum "location"** we've observed a non-zero for *bandit* is still at the hashed value of "3" for function 1.

$$\begin{array}{cc} & \begin{array}{ccc} banana & bandit & brand \end{array} \\ \begin{array}{c} h_1 \\ h_2 \\ h_3 \end{array} & \left[\begin{array}{ccc} 3 & 3 & 3 \\ 5 & 5 & 5 \\ 7 & 7 & 7 \end{array}\right] \end{array}$$

minhash Example, row 2

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$h_1(r) = (r+3) \mod 11$

$h_2(r) = (2r+5) \mod 11$

$h_3(r) = (3r+7) \mod 11$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1(2) = 5,\ h_2(2) = 9,\ h_3(2) = 2$

Step 4: For each document or column x:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1, replace $sig(i, c)$ by $\min(sig(i, c), h_i(r))$

Solution: only *brand* is eligible now.

$$\begin{array}{cccc} & banana & bandit & brand \\ h_1 & 3 & 3 & 3\ vs.\ 5 \\ h_2 & 5 & 5 & 5\ vs.\ 9 \\ h_3 & 7 & 7 & 7\ vs.\ 2 \end{array}$$

minhash Example, row 3

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$h_1(r) = (r+3) \mod 11$

$h_2(r) = (2r+5) \mod 11$

$h_3(r) = (3r+7) \mod 11$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1(3) = 6,\ h_2(3) = 0,\ h_3(3) = 5$

Step 4: For each document or column x:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1, replace $sig(i,c)$ by $\min(sig(i,c), h_i(r))$

Solution: only *bandit* is eligible now.

$$\begin{array}{c|ccc} & banana & bandit & brand \\ \hline h_1 & 3 & 3\ 6 & 3 \\ h_2 & 5 & 0\ 0 & 5 \\ h_3 & 7 & 5\ 5 & 2 \end{array}$$

minhash Example, row 4

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$$h_1(r) = (r+3) \mod 11$$
$$h_2(r) = (2r+5) \mod 11$$
$$h_3(r) = (3r+7) \mod 11$$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1(4) = 7,\ h_2(4) = 2,\ h_3(4) = 8$

Step 4: For each document or column x:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1, replace $sig(i,c)$ by $\min(sig(i,c), h_i(r))$

Solution: only *brand* is eligible now.

$$\begin{array}{c|ccc} & banana & bandit & brand \\ \hline h_1 & 3 & 3 & 3 \\ h_2 & 5 & 0 & 2 \\ h_3 & 7 & 5 & 2 \end{array}$$

minhash Example, row 5

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$h_1(r) = (r + 3) \mod 11$

$h_2(r) = (2r + 5) \mod 11$

$h_3(r) = (3r + 7) \mod 11$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1(5) = 8,\ h_2(5) = 4,\ h_3(5) = 0$

Step 4: For each document or column x:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1, replace $sig(i, c)$ by $\min(sig(i, c), h_i(r))$

Solution: banana and bandit are eligible now.

$$\begin{array}{cc} & \begin{array}{ccc} banana & bandit & brand \end{array} \\ \begin{array}{c} h_1 \\ h_2 \\ h_3 \end{array} & \left[\begin{array}{ccc} 3 & 3 & 3 \\ 4 & 0 & 2 \\ 0 & 0 & 2 \end{array}\right] \end{array}$$

minhash Example, row 6

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$h_1(r) = (r+3) \mod 11$

$h_2(r) = (2r+5) \mod 11$

$h_3(r) = (3r+7) \mod 11$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1(6) = 9,\ h_2(6) = 6,\ h_3(6) = 3$

Step 4: For each document or column x:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1, replace $sig(i,c)$ by $\min(sig(i,c), h_i(r))$

Solution: only *banana* is eligible now.

$$\begin{array}{c} \\ h_1 \\ h_2 \\ h_3 \end{array} \begin{array}{c} \begin{array}{ccc} banana & bandit & brand \end{array} \\ \left[\begin{array}{ccc} 3 & 3 & 3 \\ 4 & 0 & 2 \\ 0 & 0 & 2 \end{array}\right] \end{array}$$

minhash Example, row 7

	banana	bandit	brand
0	1	1	1
1	0	1	0
2	0	0	1
3	0	1	0
4	0	0	1
5	1	1	0
6	1	0	0
7	0	1	1

$h_1(r) = (r+3) \mod 11$

$h_2(r) = (2r+5) \mod 11$

$h_3(r) = (3r+7) \mod 11$

Step 3: Compute $h_i(r)$ for that row.

Solution : $h_1(7) = 10,\ h_2(7) = 8,\ h_3(7) = 6$

Step 4: For each document or column x:

1. If the char. matrix has a 0 in column c, row r: do nothing.
2. If the char. matrix has a 1, replace $sig(i,c)$ by $\min(sig(i,c), h_i(r))$

Solution: bandit and brand are eligible.

$$\begin{array}{c|ccc} & banana & bandit & brand \\ \hline h_1 & 3 & 3 & 3 \\ h_2 & 4 & 0 & 2 \\ h_3 & 0 & 0 & 2 \end{array}$$

minhash wrapup

Here's our final signature matrix:

$$\begin{array}{cc} & \begin{array}{ccc} banana & bandit & brand \end{array} \\ \begin{array}{c} h_1 \\ h_2 \\ h_3 \end{array} & \begin{bmatrix} 3 & 3 & 3 \\ 4 & 0 & 2 \\ 0 & 0 & 2 \end{bmatrix} \end{array}$$

which leads to similarities of 2/3 between *banana* and *bandit* and 1/3 for either other word with *brand*. But to be clear: we would do this *at least* hundreds of times more!

This would consolidate each document - which may have hundreds of thousands of rows in the characteristic matrix - into just the hundreds of rows in the signature matrix, matching our number of hash functions.

minhash wrapup

There are a lot of approximations, here

1. Shingles simplify language
2. Hashing buckets simplify/collapse shingles
3. Random row comparisons approximate Jaccard similarity
4. Permutations approximate random rows
5. Hash functions approximate permutations

... all to solve one major problem: not loading the characteristic matrices for multiple documents into memory at once.

It turns out there are some more things that can approximate/streamline the process for huge parallel data sets. See the text for how LSH (locally sensitive hashing) uses **bands** on the signature matrix to compare documents rather than computing their “full” Jaccard similarities.

Recall: Similarity

Many problems - both in this course and in general - can be expressed as the task of finding **similar** elements. We often phrase this as finding “near-neighbors.”

minhashing was a method that explored distances of **Jaccard similarity**,

$$\text{sim(S,T)} = \frac{|S \cap T|}{|S \cup T|}$$

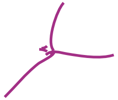
with its associated **distance**: $d = 1 - \text{sim}$.

For many data sets, we’ll want our distances in **Euclidean** spaces, like the general L_r-norm:

$$d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^r \right)^{1/r}$$

which worked well for vectors in a continuous space.

Angular Distance



A distance that in some ways can bridge both sets and points in continuous spaces is **cosine distance**.

Example: Consider two documents, each of which is composed of a single sentence.
$D_1 :=$ "the plane prepared for touchdown."
$D_2 :=$ "the prepared quarterback scored a touchdown."
Supposing that we remove the **stop words** like "the," "and," "for," and "a," we could represent these as a set and a matrix as before (**NB:** the equivalent of a k-shingle that groups words instead of characters is a k-gram):

$$\begin{array}{c c} & \begin{array}{cc} D_1 & D_2 \end{array} \\ \begin{array}{c} plane \\ prepared \\ touchdown \\ quarterback \\ scored \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}$$

... **or** we could think of these bitstrings as vectors in $\mathbb{R}^5$, and use our other forms of distance! Now we'd have:
$D_1 := [1, 1, 1, 0, 0]$
$D_2 := [0, 1, 1, 1, 1]$

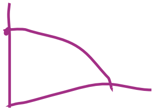
Cosine Distance

$$\begin{array}{c|cc} & D_1 & D_2 \\ \hline plane & 1 & 0 \\ prepared & 1 & 1 \\ touchdown & 1 & 1 \\ quarterback & 0 & 1 \\ scored & 0 & 1 \end{array}$$

$D_1 := [1, 1, 1, 0, 0]$
$D_2 := [0, 1, 1, 1, 1]$
We could use **edit** distance, **hamming** distance, **Jaccard** similarity, or an L-norm. But L-norms tend to use some concept ot *units*. Units of bitstrings are a little abstract, so...

Consider the angle θ formed by two vectors.

Definition: the **cosine** distance between two points (vectors) is the angle between the vectors that define those points. It is often computed using the dot product and cosine relationship:



$$\vec{x} \cdot \vec{y} = |\vec{x}||\vec{y}| \cos \theta$$

Cosines

$$\begin{array}{rcc}
 & \begin{array}{ccc} D_1 & & D_2 \end{array} & \\
\begin{array}{r} \textit{plane} \\ \textit{prepared} \\ \textit{touchdown} \\ \textit{quarterback} \\ \textit{scored} \end{array} & \left[\begin{array}{ccc} 1 & \cdot & 0 \\ 1 & \cdot & 1 \\ 1 & \cdot & 1 \\ 0 & \cdot & 1 \\ 0 & \cdot & 1 \end{array}\right] & \begin{array}{c} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{array}
\end{array}$$

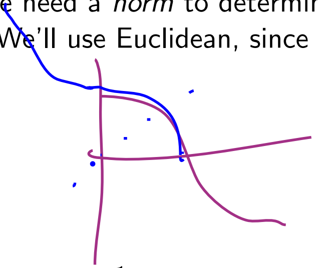
$$\Sigma = 2$$

Example: find the cosine distance between $D_1 := [1, 1, 1, 0, 0]$ and $D_2 := [0, 1, 1, 1, 1]$

Cosines

$$\begin{array}{r|cc} & D_1 & D_2 \\ \hline \textit{plane} & 1 & 0 \\ \textit{prepared} & 1 & 1 \\ \textit{touchdown} & 1 & 1 \\ \textit{quarterback} & 0 & 1 \\ \textit{scored} & 0 & 1 \end{array}$$

Example: find the cosine distance between $D_1 := [1, 1, 1, 0, 0]$ and $D_2 := [0, 1, 1, 1, 1]$
Solution: We need a *norm* to determine the magnitude of the vectors. We'll use Euclidean, since it's the standard!



arcos
cos

$$|D_1| = \sqrt{1^2 + 1^2 + 1^2 + 0^2 + 0^2} = \sqrt{3}$$
$$|D_2| = \sqrt{0^2 + 1^2 + 1^2 + 1^2 + 1^2} = \sqrt{4}$$

$$\cos\theta = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \boxed{\frac{2}{2\sqrt{3}}} \implies \theta = \arccos\frac{1}{\sqrt{3}} \approx 0.955rad\,(\approx 55^\circ)$$

Cosines

$$\begin{array}{r c} & \begin{array}{cc} D_1 & D_2 \end{array} \\ \begin{array}{r} plane \\ prepared \\ touchdown \\ quarterback \\ scored \end{array} & \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array}$$

Example: find the cosine distance between $D_1 := [1,1,1,0,0]$ and $D_2 := [0,1,1,1,1]$

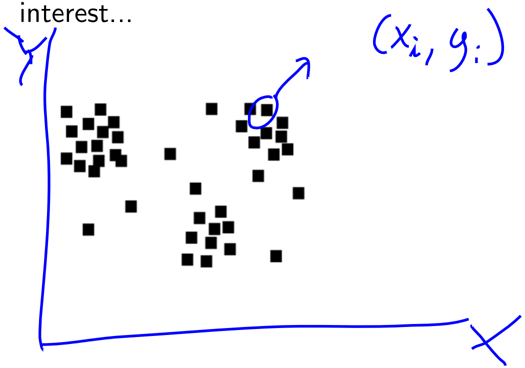
$$|D_1| = \sqrt{1^2+1^2+1^2+0^2+0^2} = \sqrt{3}$$
$$|D_2| = \sqrt{0^2+1^2+1^2+1^2+1^2} = \sqrt{4}$$

$$\cos\theta = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{2}{2\sqrt{3}} \implies \theta = \arccos\frac{1}{\sqrt{3}} \approx 0.955rad\,(\approx 55^\circ)$$

$\cos\theta$ is called the *cosine similarity*. It's *very closely* related to the correlation between the two vectors! ...more on that, later.

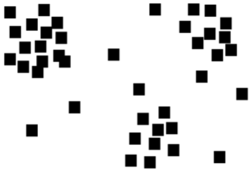
Clustering

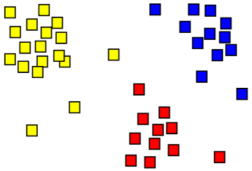
Our task now is to take data that's represented by vectors of some variable(s) of interest...



Clustering

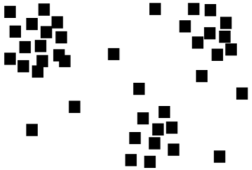
Our task now is to take data that's represented by vectors of some variable(s) of interest...
and we want to understand its underlying **structure**





Clustering

Our task now is to take data that's represented by vectors of some variable(s) of interest and understand its underlying **structure**.



Given a set of points, with a notion of distance between points, group the points into some number of clusters, so that:

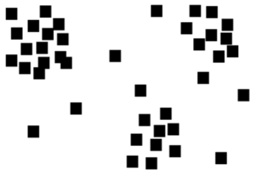
1. Members of a cluster are close/similar to each other
2. Members of different clusters are dissimilar

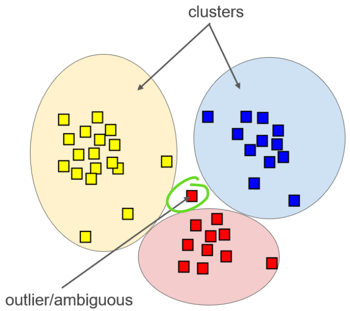
Usually:

1. Points are in a high-dimensional space
2. Similarity is defined using a distance measure
3. Euclidean, cosine, Jaccard, edit distance, etc.

Clustering Nomenclature

We define: **clusters**, and **outliers**.

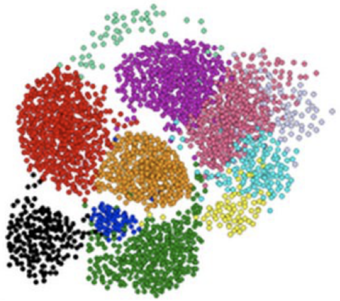




This can be a hard problem!

Clustering Issues

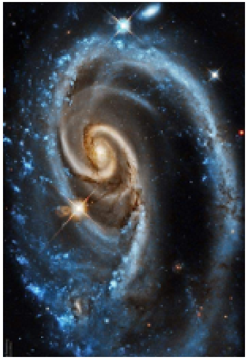
- Clustering looks easy in two dimensions
- Clustering small amounts of data looks easy
- in most cases, looks are not deceiving...
- but many applications have not 2, but 10 or 10,000 dimensions. What does that even *look* like?
- **The curse of dimensionality:** High-dimensional spaces look different... almost all pairs of points are at about the same distance!



Clustering Applications

Examples: Data points could represent...

- Different characteristics of songs:
  Goal: cluster together similar songs into genres
- Vehicle weights, milages, other characteristics:
  Goal: cluster together similar vehicles into classes (SUV, sedan, hybrid...)
- Sky object radiation intensities into frequency ranges
  Goal: cluster together into groups of similar objects.
- Words in a document
  Goal: cluster together into groups of similar topics.



Hierarchical Clustering

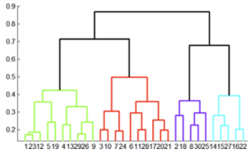
Our first technique is the agglomerative approach:
hierarchical clustering.

Idea:

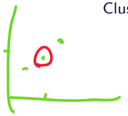
1. Each point starts as its own cluster
2. **Do:** until a stopping condition is met...
   Combine the two nearest clusters into one larger cluster.

Concerns:

1. How do we represent a cluster of more than one points?
2. How do we conceive of distances between clusters?
3. What is the stopping condition?

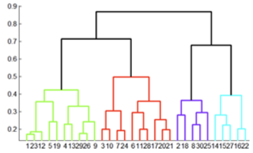


Hierarchical Clustering

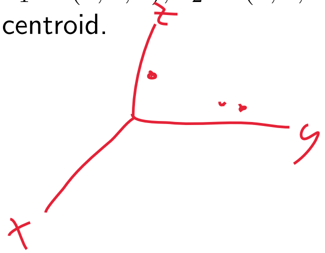


$$((1,2) , (4,3)) \Rightarrow \text{Centroid} \left(\frac{1+4}{2}, \frac{2+3}{2} \right)$$

How do we represent a cluster of more than one points?
Definition: For the Euclidean case, we can use the **centroid** of the cluster. This is the *average of its data points.*



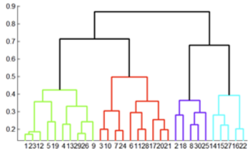
Example: Suppose a cluster contains the data points $x_1 = (1, 3, 2), x_2 = (0, 5, 1)$ and $x_3 = (1, 4, 1)$. Find its centroid.



$$\frac{1+0+1}{3}, \quad \frac{3+5+4}{3}, \quad \frac{2+1+1}{3}$$

Hierarchical Clustering

How do we represent a cluster of more than one points?
Definition: For the Euclidean case, we can use the **centroid** of the cluster. This is the *average of its data points.*



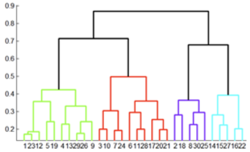
Example: Suppose a cluster contains the data points $x_1 = (1, 3, 2), x_2 = (0, 5, 1)$ and $x_3 = (1, 4, 1)$. Find its centroid.

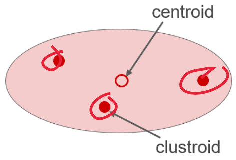
Answer: centroid is:$((1 + 0 + 1)/3, (3 + 5 + 4)/3, (2 + 1 + 1)/3) = (2/3, 4, 4/3)$

Hierarchical Clustering

How do we represent a cluster of more than one points?
Definition: For the non-Euclidean case, we can use the **clustroid** of the cluster. This is the *data point in the cluster that is closest to the other points in the cluster.* This could be the...

1. smallest maximum distance to the other points
2. smallest average distance to the other points
3. smallest sum of squares/aggregated distances to the other points...



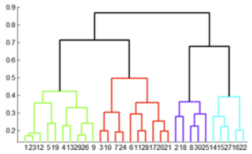


Hierarchical Clustering

Using **centroids** or **clustroids** answered our first two questions: how to represent a cluster of more than one points and conceive of distances between clusters?

Final question: what is the stopping condition?
Some options include:

1. Stop once clusters are of an appropriate "size"
   1.1 Max distance between points in a cluster?
   1.2 Average distance between points in a cluster?
   1.3 Density of points in a cluster?
2. Once there is a target number of clusters.



Hierarchical Implementation

Implementation Notes

At each step, we compute *all* distances between pairs of clusters. Then merge the nearest cluster.

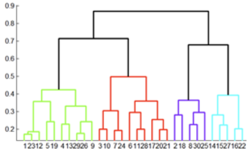
With N data points, this is $\boxed{N^2}$ comparisons to make! (or $\binom{N}{2}$, at least).

IF we want k clusters at the end, and $k << N$, then we need to iterate about N times to merge down to k clusters.

This means $\mathcal{O}(N^3)$ complexity.

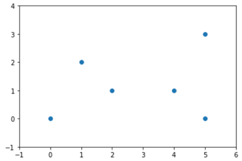
With some cleverness, we can get this down to $\mathcal{O}(N^2 \log N)$.

... that's still rough for very large data sets.



Hierarchical Example

Example: Consider the data set $(0,0),(1,2),(2,1),(4,1),(5,0),(5,3)$. Use hierarchical clustering and Euclidean distance to group the data into 2 clusters. If there are ties in distance, merge first the data points with lower x-coordinates.

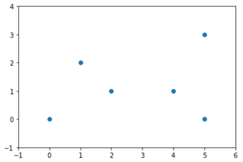


Hierarchical Example

Example: Consider the data set $(0,0),(1,2),(2,1),(4,1),(5,0),(5,3)$. Use hierarchical clustering and Euclidean distance to group the data into 2 clusters. If there are ties in distance, merge first the data points with lower x-coordinates.

Solution: We might initially construct a full distance matrix!

$$\begin{array}{c} \\ (0,0) \\ (1,2) \\ (2,1) \\ (4,1) \\ (5,0) \\ (5,3) \end{array} \begin{array}{c} \begin{array}{cccccc} (0,0) & (1,2) & (2,1) & (4,1) & (5,0) & (5,3) \end{array} \\ \begin{bmatrix} & \sqrt{5} & \sqrt{5} & \sqrt{17} & 5 & \sqrt{34} \\ & & \sqrt{2} & \sqrt{10} & \sqrt{20} & \sqrt{17} \\ & & & 2 & \sqrt{10} & \sqrt{13} \\ & & & & \sqrt{2} & \sqrt{5} \\ & & & & & 3 \\ & & & & & \end{bmatrix} \end{array}$$

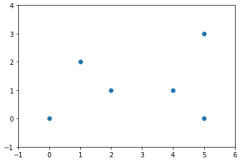


There are a couple of $\sqrt{2}$ in there, so we pick the tiebreaker of the lowest x-values, which are $(1,2)$ and $(2,1)$.

Hierarchical Example

Example: Consider the data set $(0,0),(1,2),(2,1),(4,1),(5,0),(5,3)$. Use hierarchical clustering and Euclidean distance to group the data into 2 clusters. If there are ties in distance, merge first the data points with lower x-coordinates.

Solution: Now we'd have a **cluster** inside our distance matrix. Points $(1,2)$ and $(2,1)$ get folded into a **cluster** with **centroid** at $(3/2,3/2)$.
We could ostensibly recreate the matrix, but now in a 5x5 instead of 6x6 format. For this smaller problem, let's proceed visually, instead.

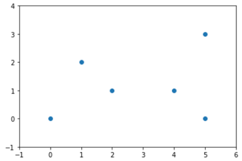


Hierarchical Example

Example: Consider the data set $(0,0),(1,2),(2,1),(4,1),(5,0),(5,3)$. Use hierarchical clustering and Euclidean distance to group the data into 2 clusters. If there are ties in distance, merge first the data points with lower x-coordinates.

Solution: full combine order

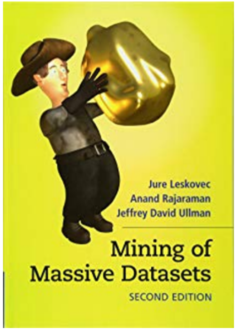
1. Combine $(1,2)$ and $(2,1)$ into group red
2. Combine $(4,1)$ and $(5,0)$ into group blue
3. Fold $(0,0)$ into red group.
4. Fold $(5,3)$ into blue group.
5. STOP: we're down to 2 groups, since every points is either red or blue.



Acknowledgments

We'll pick up with how to fix how slow it is to compute these matrices, next time!

Some material is adapted/adopted from Mining of Massive Data Sets, by Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford University) `http://www.mmds.org`



Special thanks to Tony Wong for sharing his original adaptation and adoption of slide material.