Strongly Connected Components and DBSCAN

Joshua Benabou

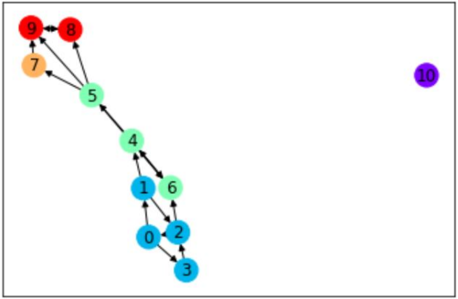
May 29, 2020

Strongly Connected Components

A digraph $G = (V, E)$ is **strongly connected** (SC) if it contains a path from u to v for all nodes $u, v \in V$.

A sub-digraph of G is a **strongly connected component** (SCC) of G if it is SC and maximal with this property.

Any digraph has a unique SCC-decomposition.



We present 2 algos to find SCC's in digraphs in time $O(|V| + |E|)$.

Algorithms to find SCC's : Kosaraju

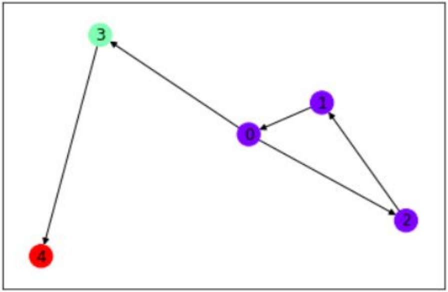
Kosaraju's algorithm (1981) identifies the SCC of a node u as the set of nodes reachable from u by both forward and backward traversal:

1. Initialize stack $st = []$. Construct transpose G^t.
2. Do DFS on G: **after** node u is processed, push u to st.
3. Do DFS on G^t: while $st \neq []$, pop a node u from st. If label of u not already assigned, assign u a new label l, and assign l to all members of u's connected component in G^t.

Main principle: after the forward-traversal DFS (step 2), if G contains an edge $u \to v$, then u appears before v in st (unless u and v are in the same SCC)

Algorithms to find SCC's : Kosaraju

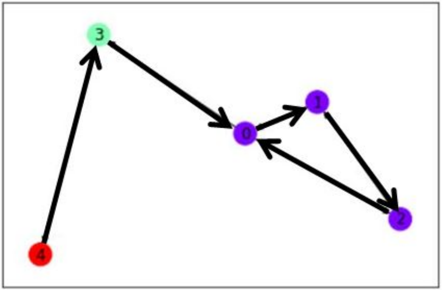
Kosaraju example:



After forwards DFS (start at 0): $st = [1, 2, 4, 3, 0]$

Algorithms to find SCC's : Kosaraju

Transposed graph:



$st = [1, 2, 4, 3, 0]$

Pop 0: SCC $\{0, 1, 2\}$ is found

Pop 3: SCC $\{3\}$ is found

Pop 4: SCC $\{4\}$ is found

Pop 2,1: already labeled

Algorithms to find SCC's : Tarjan

Tarjan's algorithm (1972) only uses one DFS from arbitrary start node, and a counter.

$dt(u)$=counter value when we encounter u during the DFS

$low(u)$= smallest value of $dt(v)$ over all nodes already-visited nodes v in the DFS-subtree of u.

Initialize stack $st = []$.

Do DFS on G: push each node u to st **as it is explored**. Then:

- Define $dt(u)$, and explore children of u to update $low(u)$.
- If a node u is such that $dt(u) = low(u)$, then assign a new label l to u and all the nodes above u in st. Pop all these nodes (including u) from st.

Main principle: nodes of an SCC of G form a subtree in the DFS spanning tree of G. Thus, if we encounter u such that $dt(u) = low(u)$, then u is the node of its SCC that was explored first.

Performance on SNAP datasets

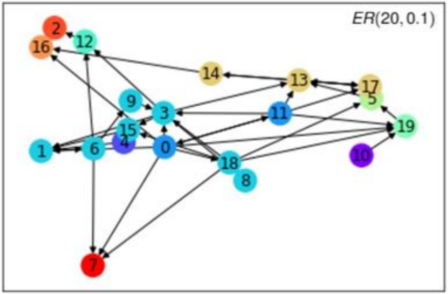
Complexity for Kosaraju and Tarjan is same as DFS: $O(|V| + |E|)$

Dataset					Runtimes (ms)	
Name	$\|V\|$	$\|E\|$	#SCC	largest SCC	Kosaraju	Tarjan
p2p-Gnutella08	6301	20777	4234	2068	18	12
p2p-Gnutella24	26518	65369	20167	6352	88	62
Wiki-Vote	7115	103689	5816	1300	69	22
p2p-Gnutella31	62586	147892	?	14149	DK	DK

DK="Dead Kernel"

Erdos-Renyi Graphs

Given $p \in [0, 1]$, an **Erdos-Renyi digraph** $ER(n, p)$ on n vertices is constructed by defining, for each pair of distinct vertices (i, j) an edge $i \to j$ with probability p.

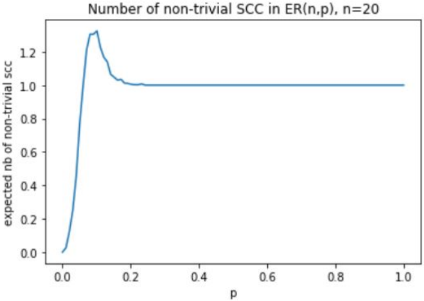


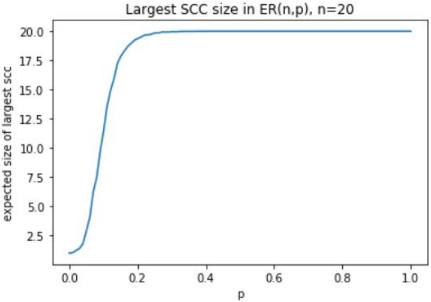
How do SCC's behave in these graphs?

For small p, $ER(n, p)$ has fewer edges and is less likely to be strongly connected than for large p.

ER Graphs: Statistics of SCC's

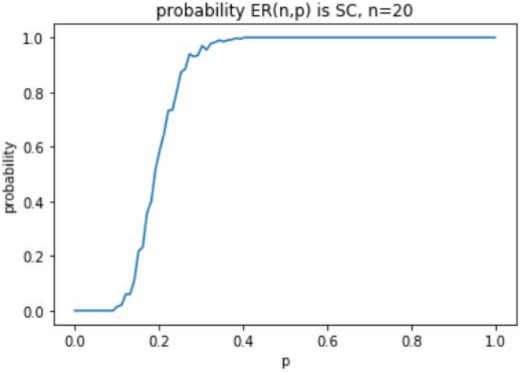
Statistics on the expected **number** and expected **maximal size** of SCC's in $ER(20, p)$ for 50 equally-spaced p in $[0, 1]$ (100 generations for each p):





The maximal expected value attained is only slightly greater than 1,

ER Graphs: Statistics of SCC's



Interval of p for which the graph is likely neither strongly connected nor has 0 non-trivial SCC's is small.

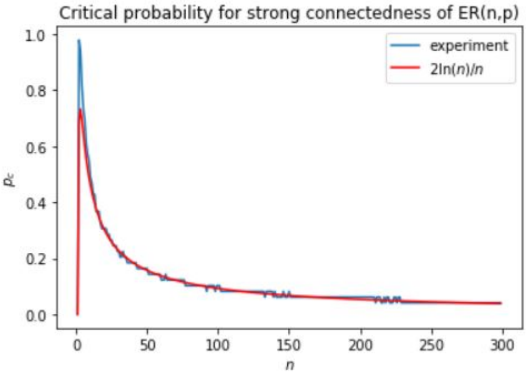
Abrupt phase transition for large n?

Critical probability for strong connectedness

Erdos and Renyi (1960) showed $p = \ln(n)/n$ is a sharp bound for the **connectedness** of $ER(n, p)$.

Define $p_c(n)$ as the smallest p such that $\Pr[ER(n, p) \text{ is SC}] > 0.95$.

We conjecture that $p_c = 2\ln(n)/n$ is a sharp asymptotic bound for **strong-connectedness**:



DBSCAN

DBSCAN clusters a *geometric* dataset by grouping together points with many nearby neighbors, and marking as outliers points whose nearest neighbors are too far away.

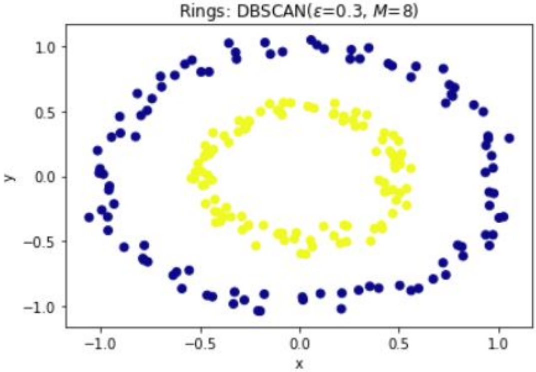
Input: set P of points; a distance function $d : P \times P \to \mathbb{R}$.

Hyperparameters: search radius ϵ; minimum number M of points to form a dense region.

- Given some unvisited point p, if $|P| \cap B(p, \epsilon) \geq M$, p will be part of a cluster, which will also contain all its ϵ-neighbors $|P| \cap B(p, \epsilon)$.
- Else, p is tentatively labeled as an outlier (it could be determined later to be part of a cluster).
- Explore ϵ-neighbors of p, their ϵ-neighbors and so on via DFS, to determine cluster of p,
- To find other clusters, repeat above on unvisited points.

DBSCAN: results

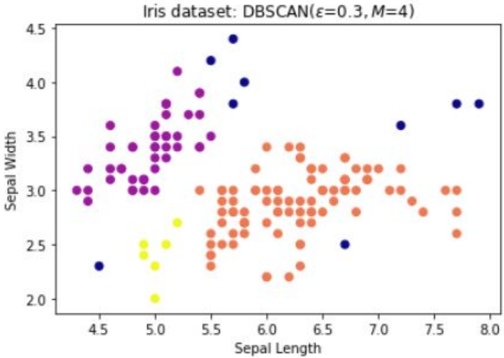
Synthetic rings, $N = 200$:



Two detected clusters, 0 outliers (silhouette=0.13).

DBSCAN: results

Iris dataset for $N = 150$ flowers belonging to two species:



3 detected clusters, 9 outliers (silhouette=0.45).

Heuristics for selecting M and ϵ

We select M using prior knowledge about the dataset.

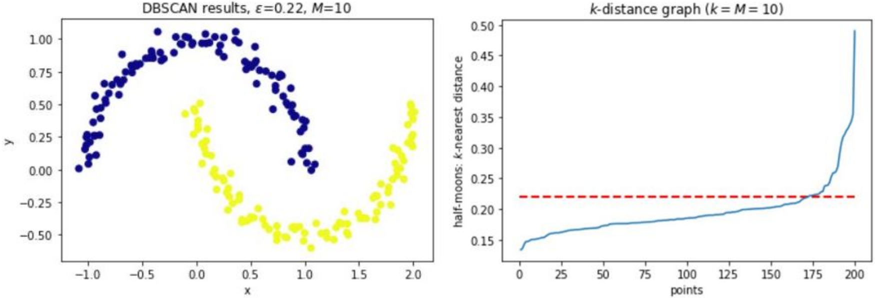
For data in $\mathbb{R}^d$ with low noise, M on the order d gave moderate results.

For fixed M, we tried two heuristics for choosing ϵ.

Heuristic 1 for ϵ: KNN-plot

KNN plot: For each point, plot distance to its *M*-nearest neighbor, and order these distances from smallest to largest.

Optimal ϵ is chosen as the distance coinciding with the "elbow" of this plot, where the distance increases sharply.



knn-plot for "half-moons" gives $\epsilon = 0.22$, which gives the correct result (2 clusters).

Heuristic 2 for ϵ: silhouette

Silhouette score: Given a DBSCAN output, define for each non-outlier point p:

- $a(p)$= average distance of p to points in its cluster C
- $b(p)$= minimum over $C' \neq C$ of average distance to points in cluster C'
- The silhouette coefficient

$$s(p) = \frac{a(p) - b(p)}{\max(a(p), b(p))}$$

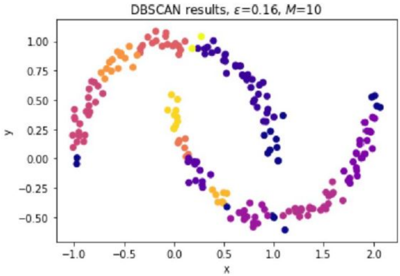
- Silhouette score s= average of $s(p)$ over all non-outliers. So $s \in [-1, 1]$.

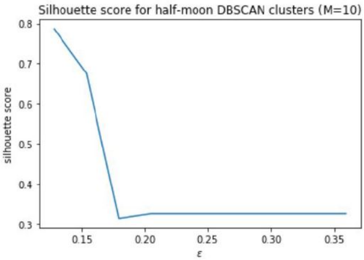
We choose the ϵ which maximises the silhouette score

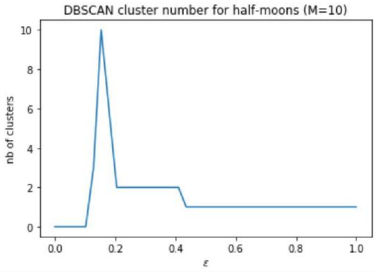
Not so efficient as a heuristic, more useful as metric for solution quality.

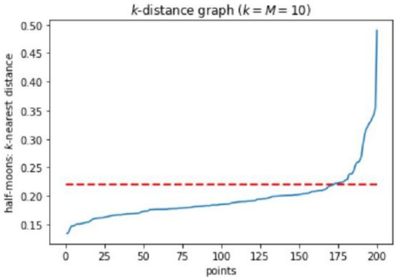
Worse, silhouette coefficient does not always correspond to optimal clustering!

Heuristic 2 for ϵ: silhouette









DBSCAN Runtime

Our neighbor query to find $B(p, \epsilon)$ is implemented with linear scan.

Using a kd-tree, could could be done in $O(\log n)$ on average (if ϵ is not too large). $\Rightarrow$ average runtime $O(n \log n)$.

In any implementation, worst case is $O(n^2)$.

Dataset	#points	M	ϵ	#clusters	#outliers	Runtime (s)
Iris	150	4	0.3	3	9	0.027
Airports (US)	1298	50	300	4	238	4.3
Airports (World)	7698	10	50	21	7318	215
half-moons (synthetic)	1000	10	0.22	2	2	2
half-moons (synthetic)	10000	10	0.22	2	1	227
half-moons (synthetic)	30000	10	0.22	2	1	2006

SCC-finders vs DBSCAN

To compare SCC-finders and DBSCAN on same data, we need a geometrical dataset which is also a graph.

We will use aviation data:

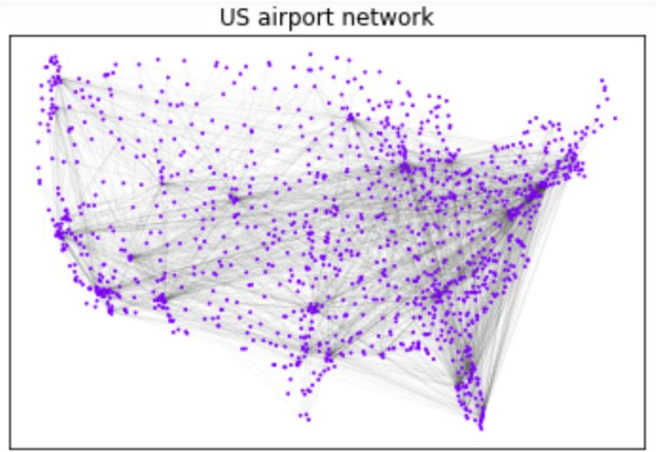
- Information about world airports (including **latitude/longitude**)
- List of commercial **routes** between airports (source, destination, etc)

We focus on mainland USA for easy visualization.

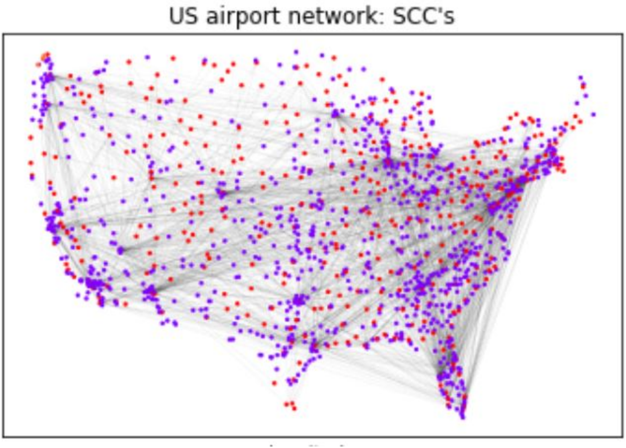
- DBSCAN (using great-circle distance), shows where airports are clustered geographically. Expect clustering on coast and many outliers (small airports).
- Constructing a graph with airports as nodes and flight routes as directed edges, the SCC's tell us about connectivity of the US airports. Ideally, the US network should be SC.

SCC-finders vs DBSCAN

Locations of the 1298 mainland US airports and 4948 logged commercial routes:



SCC-finders vs DBSCAN

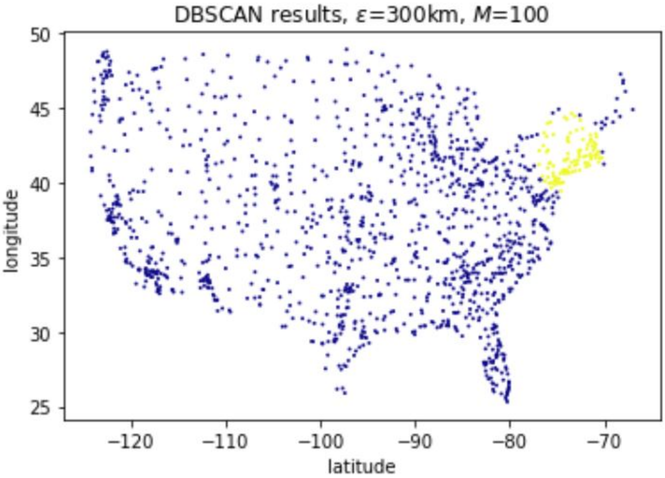


SCC of size 398, one of size 4, 896 singletons. In fact, red airports have 0 logged in or out flights - incomplete data!

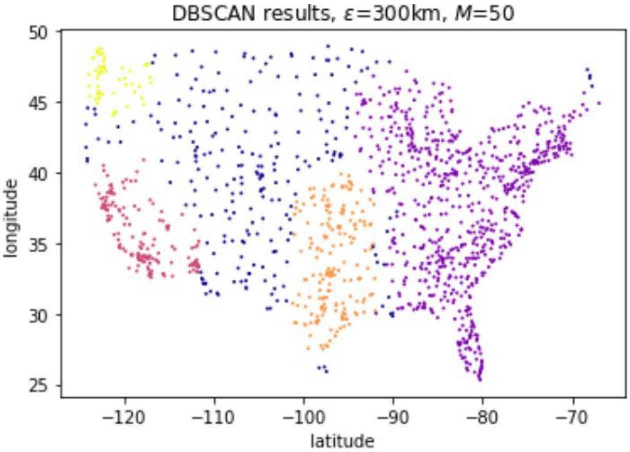
The US flight network is thus strongly connected!

SCC-finders vs DBSCAN

Let's fix $\epsilon = 300$ km and vary M. For $M = 100$, only the Northeast cluster (JFK, Boston, etc) survives, the remaining points are outliers:



SCC-finders vs DBSCAN



Silhouette score: 0.46. We have identified the hubs of US flight activity: Southern California (e.g LAX), Northern California (e.g SFO), Eastern seaboard (JFK, Washington, Atlanta, etc), and center-south (Dallas, Denver, etc)