POLITECNICO DI MILANO

SLINK





Andrea Battistello 873795

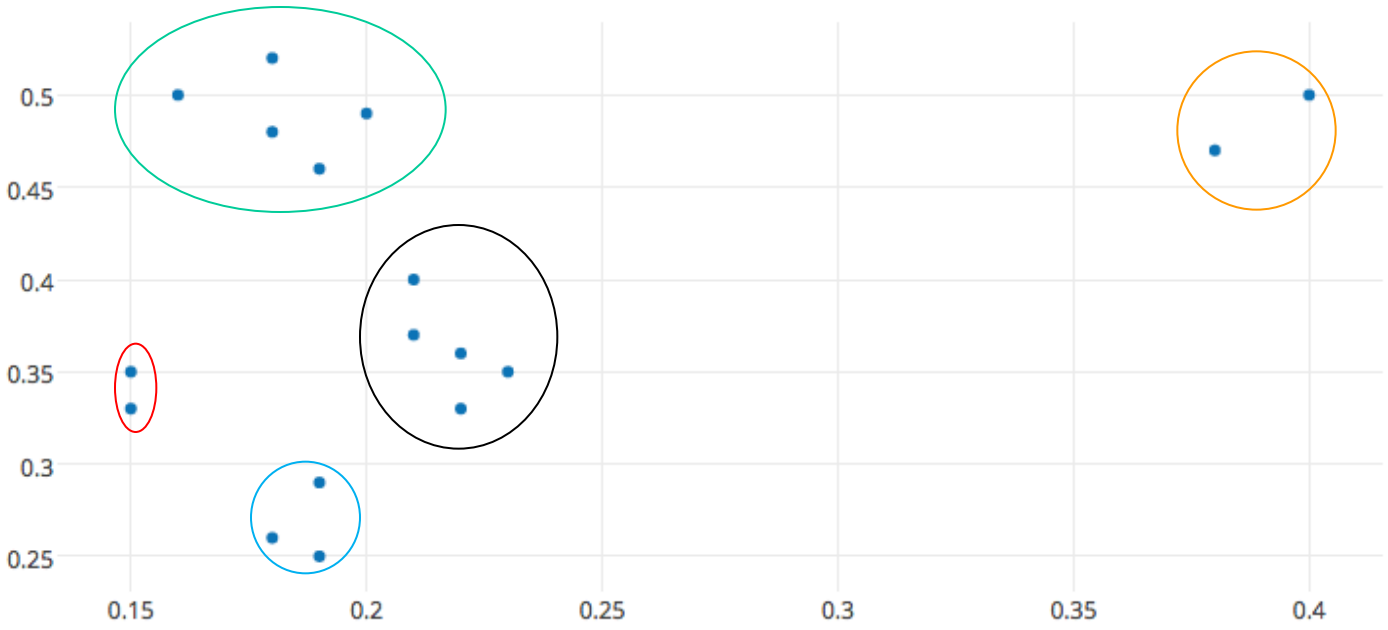
Fabio Chiusano 874294

ROADMAP

- Clustering
- Single linkage naive method
- Theoretical limits
- Pointer representation
- SLINK algorithm
- Classifiability
- Presentation of results
- Benchmark and analysis of results

Clustering

Clustering is the process of grouping objects so that similar objects are in the same cluster and dissimilar objects are in different clusters



Clustering

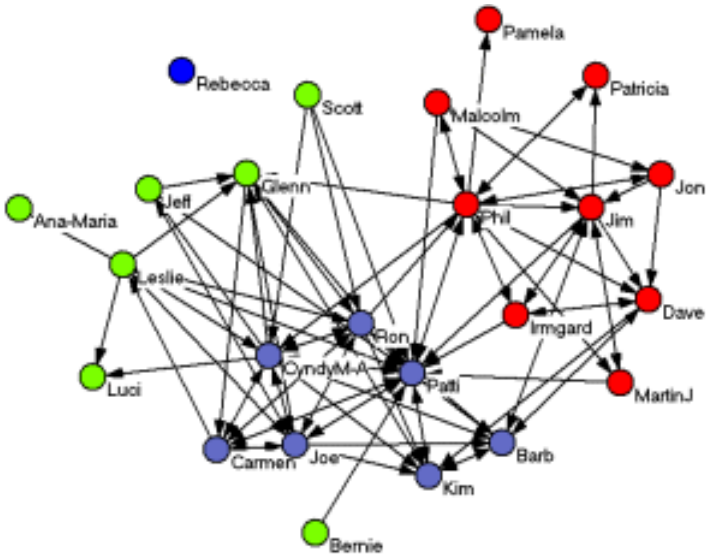
Applications:

Business: marketing segmentation, product positioning

WWW: social network analysis

Computer science: image segmentation, recommender systems, evolutionary algorithms

Medicine: analysis of antimicrobial activity



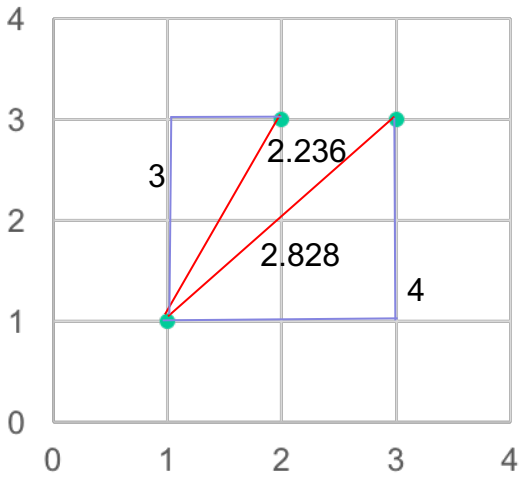
Clustering

What does ”similar objects” mean?

A definition of similarity is needed. We can define a distance/dissimilarity function that accepts two objects and whose result is a measure of how the two objects are different from each other.

Some examples of dissimilarity functions:

- Euclidean distance
- Manhattan distance
- Custom dissimilarity functions built even upon non numerical objects



Clustering

How is the dissimilarity function used?

Connectivity models: hierarchical clustering where grouping is done according to the dissimilarity between members of different groups. E.g. SLINK

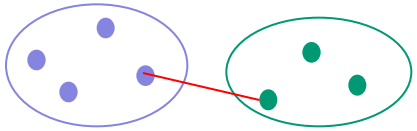
Density models: a point is in a group if it is sorrounded by at least a certain number of other points whose dissimilarity with that point is lower than a certain amount. E.g. DBSCAN

Centroid models: each cluster is represented by a single point, which is its centroid. E.g. k-means

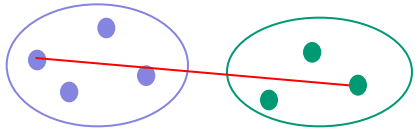
Clustering

In connectivity models, how do we compare two clusters where at least one has more than one point?

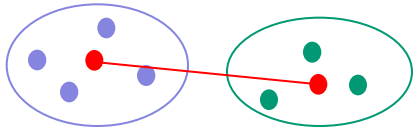
Single linkage: use the minimum dissimilarity between a point of the first cluster and a point of the second
cluster



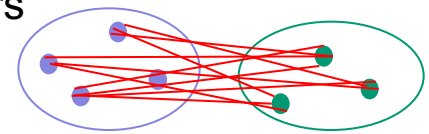
Complete linkage: use the maximum dissimilarity between a point of the first cluster and a point of the second
cluster



Use the dissimilarity between the centroids of the two clusters



Average linkage: use the average dissimilarity between each possible pair of points belonging to different clusters



Clustering

Is it possible to say that a point belongs to a cluster with a certain probability?

Hard clustering: each point belong to a cluster or not. E.g. SLINK

Soft/fuzzy clustering: each point belong to each cluster with a certain degree. E.g. Fuzzy C-means

Can a point belong to more than one cluster?

String partitioning clustering: each point belong to only one cluster. E.g. SLINK

Overlapping clustering: each point belong to a cluster or more. E.g. COGNAC

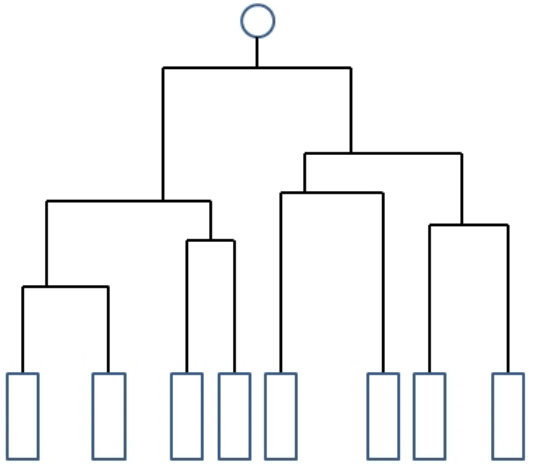
Clustering

So, what is SLINK?

SLINK is an intelligent and efficient implementation of hierachical clustering based on connectivity, where each point belongs to only one cluster and the single linkage criterion is used.It's an online algorithm.

How are the results presented?

The results are often presented as a dendrogram.



Single linkage naive method

1. Compute the dissimilarity matrix in $O(n^2)$ time and space.
2. Find the rows with the lowest dissimilarity value (there could be many rows with same minimum value)
3. Merge such rows in a cluster
4. Repeat until you are left with only one point (At most will take O(n) iterations)

Total complexity: $O(n^3)$ time and $O(n^2)$ space

e.g.

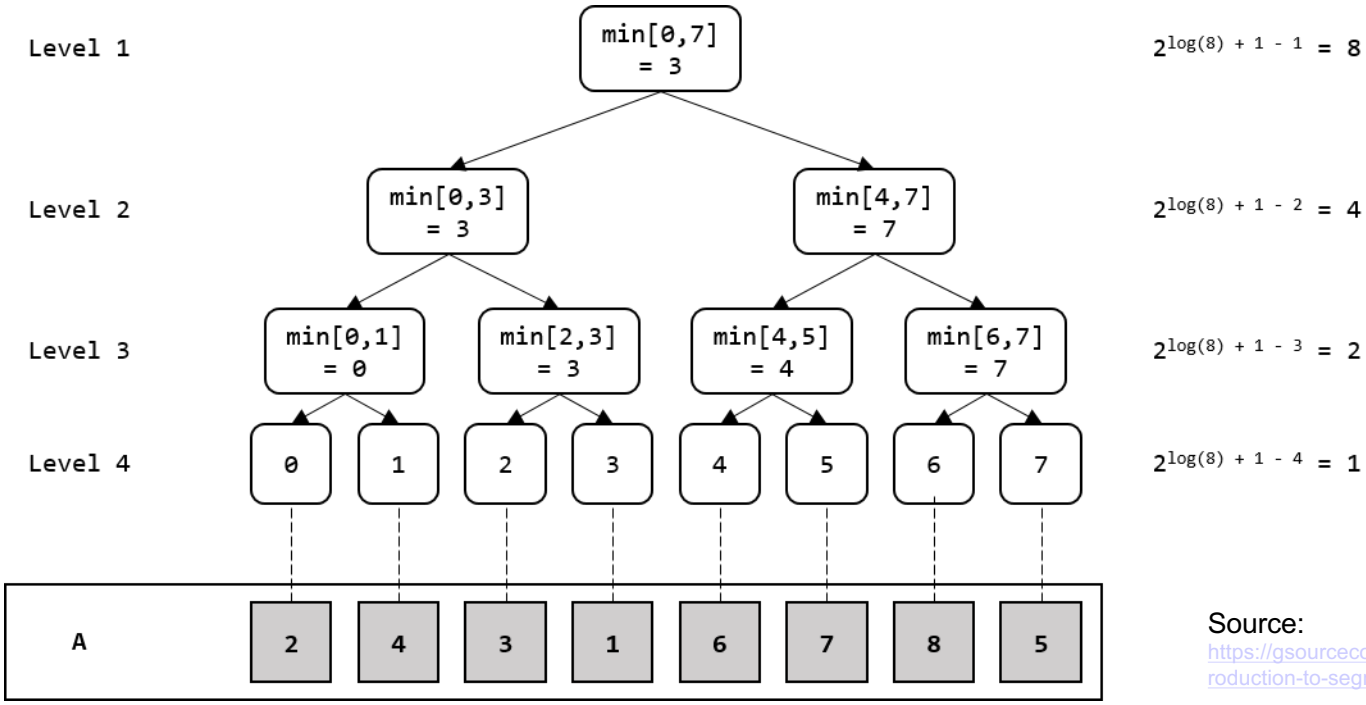
$$\begin{matrix} & 1 & 2 & 3 & 4 \end{matrix} \quad D = \begin{pmatrix} 0 & & & \\ 1.2 & 0 & & \\ 0.7 & 1.4 & 0 & \\ 3.4 & 2.3 & 3.6 & 0 \end{pmatrix} \xrightarrow{\text{Join 1\&3}} \begin{matrix} & 1\&3 & 2 & 4 \end{matrix} \quad D = \begin{pmatrix} 0 & & \\ 1.2 & 0 & \\ 3.4 & 2.3 & 0 \end{pmatrix} \xrightarrow{\text{Join 1\&(2\&3)}} \begin{matrix} & 1\&2\&3 & 4 \end{matrix} \quad D = \begin{pmatrix} 0 & \\ 2.3 & 0 \end{pmatrix}$$

Optimizing naive method

Observation: we can avoid the visit to all $O(n^2)$ cells to find the minimum.

We can use a **segment tree** to find the minimum in O(1) and handle the O(n) updates in O(log n) time each.

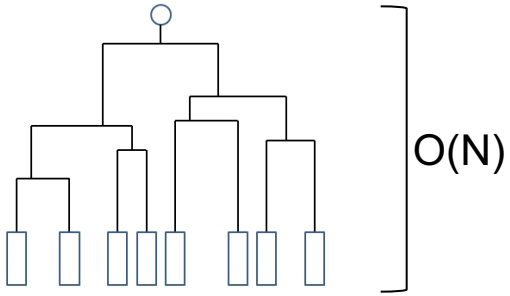
Total complexity: $O(n^2 \log n)$ time and $O(n^2)$ space



Source: https://gsourcecode.wordpress.com/2014/03/12/introduction-to-segment-trees-range-minimum-query/

Theoretical limits

Space: a dendogram of N points can have up to N - 1 dinstict splitting levels, i.e. levels where some clusters merge, and so at least storage O(N) is required.



Time: a cluster method operating on the dissimilarity measure will have a time-dependence of at least $O(N^2)$ because each dissimilarity must be examined at least once.

d(1,1)	d(1,2)	...	d(1,N)
d(2,1)	...	...	...
...	...	...	...
d(N,1)	...	...	d(N,N)

O(N)

O(N)

Pointer representation

If the dissimilarity matrix is held in memory, $O(N^2)$ locations will be needed, whereas both the original data and the dendrogram only require O(N) locations. Thus, we want to avoid holding the matrix in memory if possible.

The SLINK algorithm avoids this problem by using the dissimilarity values just once: at stage n, a random access is needed only to values of the form d(i,n) for i < n, and no sorting or rearrangement are employed.

Therefore, the SLINK algorithm uses smaller data structures that take O(N) space, and this is done by defining an equivalent representation of a dendrogram on which the algorithm will work, that is called **pointer representation**.

Pointer representation

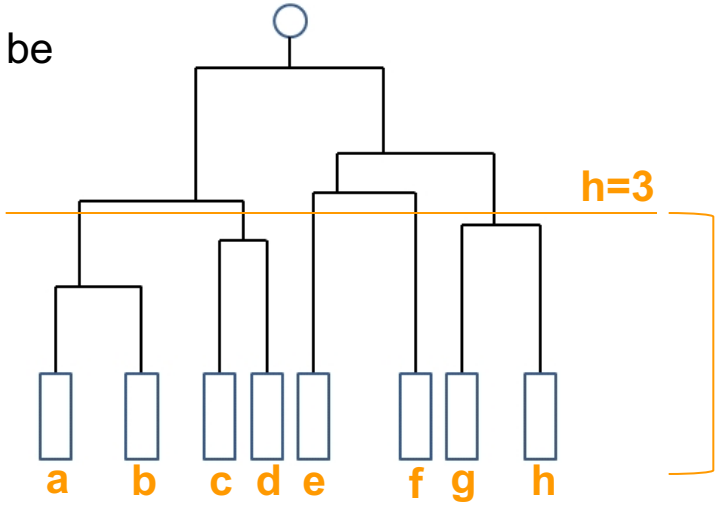
A dendrogram can be defined as the following function:

$$c: [0, +\infty) \rightarrow E(P)$$

Where P is the set of points that we want to cluster, E(P) is the set of equivalence classes (clusters) that we have using a specific maximum dissimilarity value (the argument of the function).

E.g. c(h) = E(P) can be represented in the following way:

P = {a,b,c,d,e,f,g,h)



{a,b} is an equivalence class because their dissimilarity is less than h. Same for {c,d}, {g, h}.

E(P) = {{a,b},{c,d},{e},{f},{g,h}}

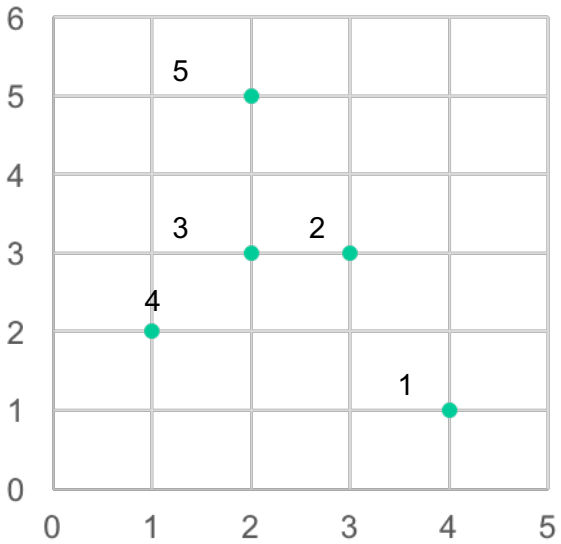
Pointer representation

A pointer representation is defined by two functions defined over the data, where each data point is given a number from 1 to N:

$$\lambda(i) = inf\{h: \exists j > i \; s.t. \; (i,j) \in c(h)\}$$

$$\pi(i) = max\{j : (i,j) \in c(\lambda(i))\}$$

Where λ(i) is the lowest level at which i is no longer the last object of its cluster and π(i) is the last object in the cluster which it then joins.



λ(1) = 3, π(1) = 5
λ(2) = 1, π(2) = 3
λ(3) = 2, π(3) = 5
λ(4) = 4, π(4) = 5
λ(5) = +∞, π(5) = 5

Pointer representation

It is proven in the paper of Sibson (1972) that **there is a 1-1 correspondence between dendrograms and pointer representations**.

Thus, by working only on the pointer representation, it's possible to convert it to a dendrogram. Indeed, **SLINK works only on the pointer representation**, by recursevely updating it.

SLINK starts considering only 1 point and building the pointer representation. Then, it considers also the next point and updates the pointer representation, and so on. There are N points to be considered and each pointer representation update takes O(N) time, so the total running time will be $O(N^2)$. It's proven that the final pointer representation obtained in this way is equivalent to the dendrogram built with the naive implementation of the algorithm.

Moreover, the memory is used only for the functions λ and π, which take O(N) storage, and for other auxiliary arrays that take again O(N) storage. Thus, the total space needed is O(N).

SLINK algorithm

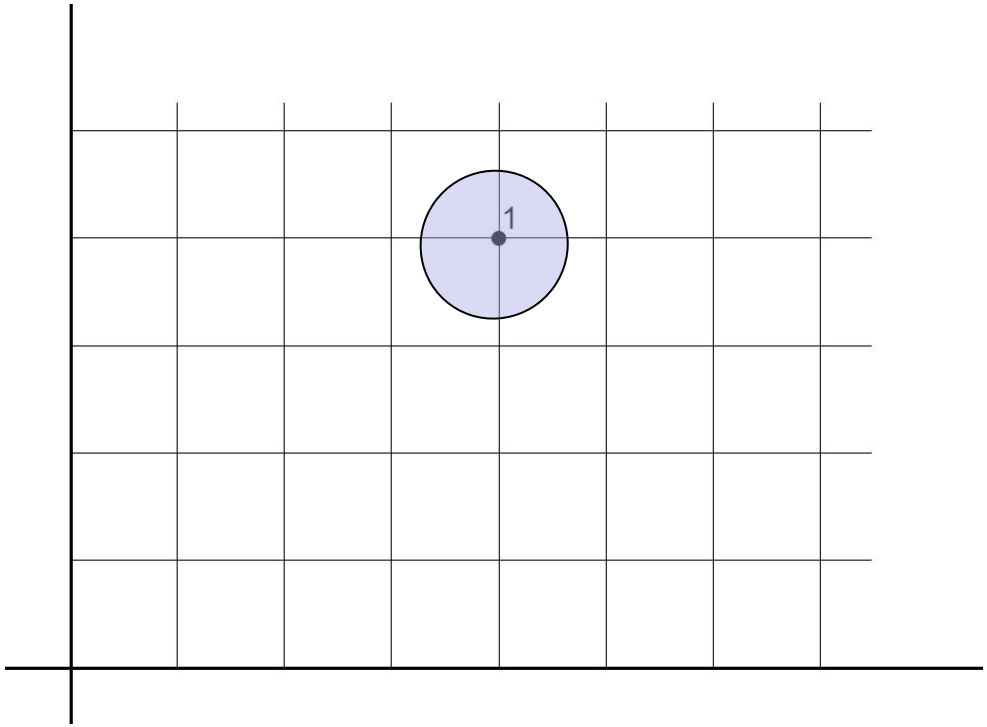
Suppose that Π, Λ contains π_n, λ_n in their first n positions. Then for the new point n+1 we will rewrite Π and Λ in this way:

```
1. Set Π(n+1) to n+1, Λ(n+1) to ∞
2. Set M(i) to d(i, n+1) for i = 1 .. N
3. For i increasing from 1 to n
        if Λ(i) ≥ M(i)
                set M(Π(i)) to min{ M(Π(i)), Λ(i) }
                set Λ(i) to M(i)
                set Π(i) to n + 1
        if Λ(i) < M(i)
                set M(Π(i)) to min{ M(Π(i)), M(i) }
4. For i increasing from 1 to n
        if Λ(i) ≥ Λ(Π(i))
                set Π(i) to n + 1
```

SLINK algorithm

n = 0

	1					
λ	∞					
π	1					
M	–					



Set:

$\lambda_0(1) = \infty$

$\pi_0(1) = n+1 = 1$

SLINK algorithm

n = 1

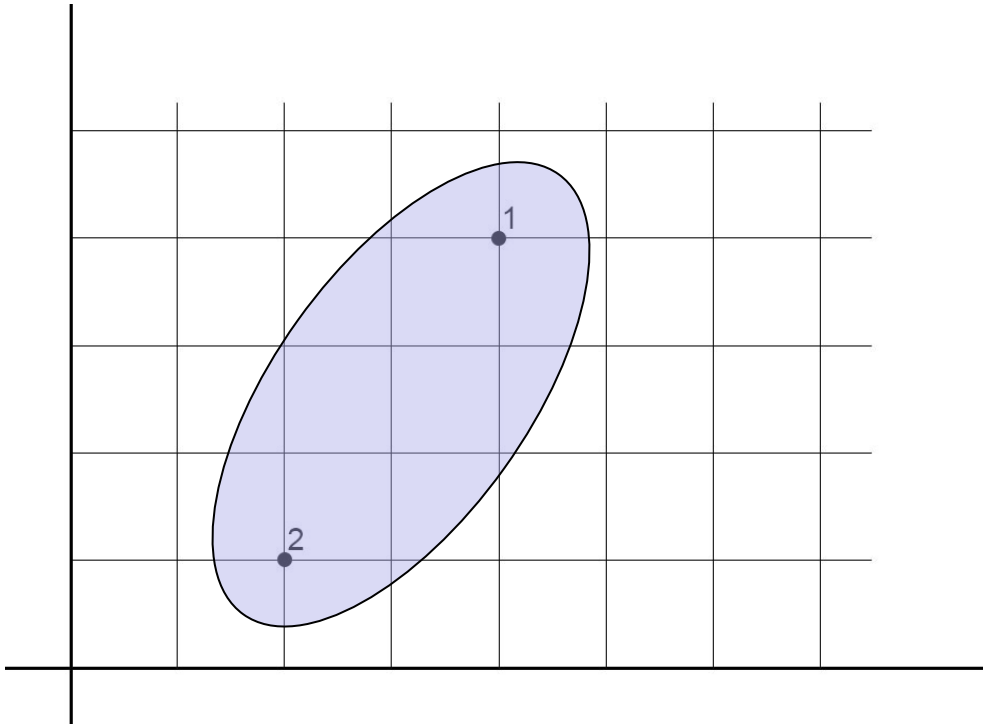
	1	2				
λ	~~∞~~ 5	∞				
π	~~1~~ 2	2				
M	5	–				

Set:

$\lambda_1(2) = \infty$

$\pi_1(2) = n+1 = 2$

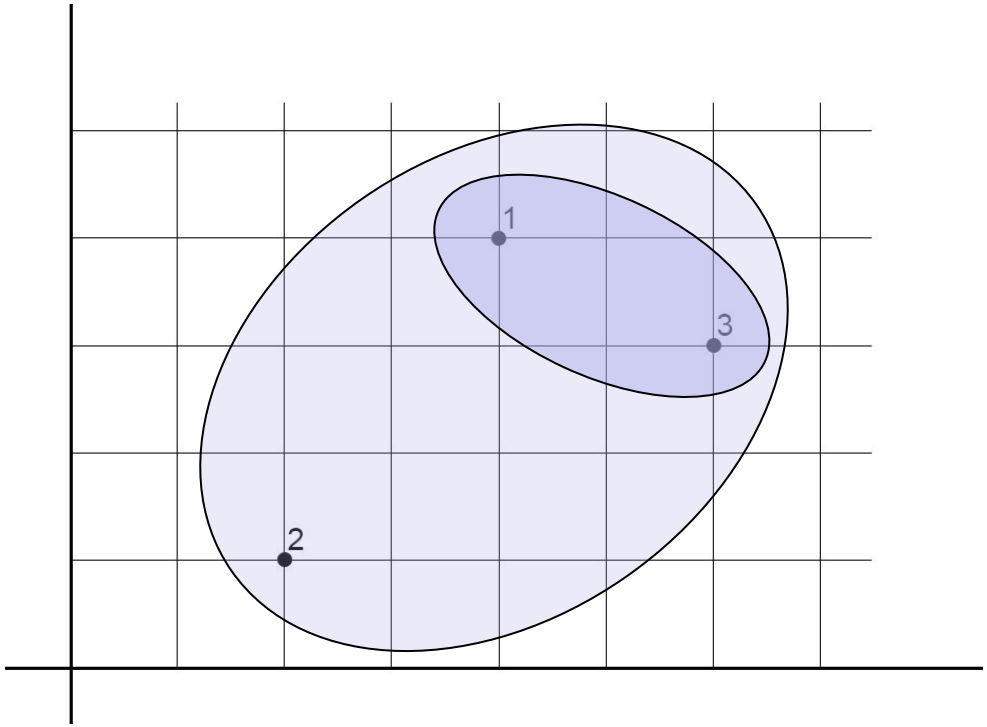
Point 2 is closer than ∞, so add it to its cluster.



SLINK algorithm

n = 2

	1	2	3			
λ	~~5~~ 3	~~∞~~ 5	∞			
π	~~2~~ 3	~~2~~ 3	3			
M	3	~~6~~ 5	–			



Set:

$\lambda_2(3) = \infty$

$\pi_2(3) = n+1 = 3$

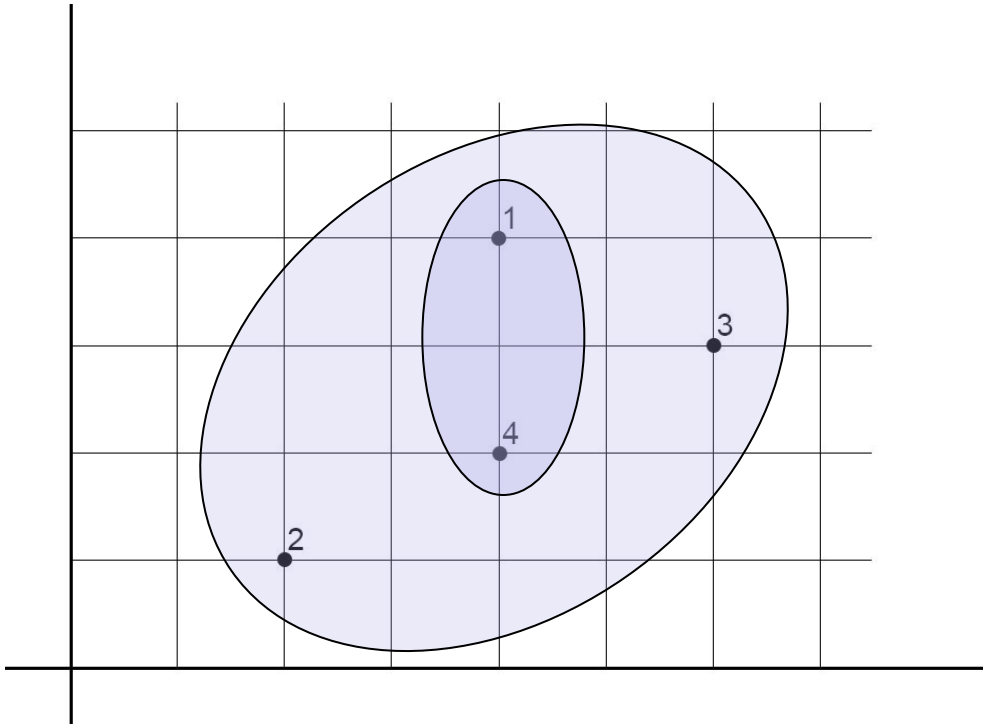
(3) is closer than 5 for(1), so update $\lambda_2(1) = 3$ and $\pi_2(1) = 3$. Now (2) can reach point (3) through (1), so the required distance is 5.

(3) is closer than ∞ for (2), so set $\lambda_2(2) = 5$ and $\pi_2(2) = 3$

SLINK algorithm

n = 3

	1	2	3	4		
λ	~~3~~ 2	~~5~~ 3	~~∞~~ 3	∞		
π	~~3~~ 4	~~3~~ 4	~~3~~ 4	4		
M	2	3	3	–		



Set:

$\lambda_3(4) = \infty$

$\pi_3(4) = n+1 = 4$

(4) is closer than 3 for(1), so set $\lambda_3(1) = 2$ and $\pi_3(1) = 4$.

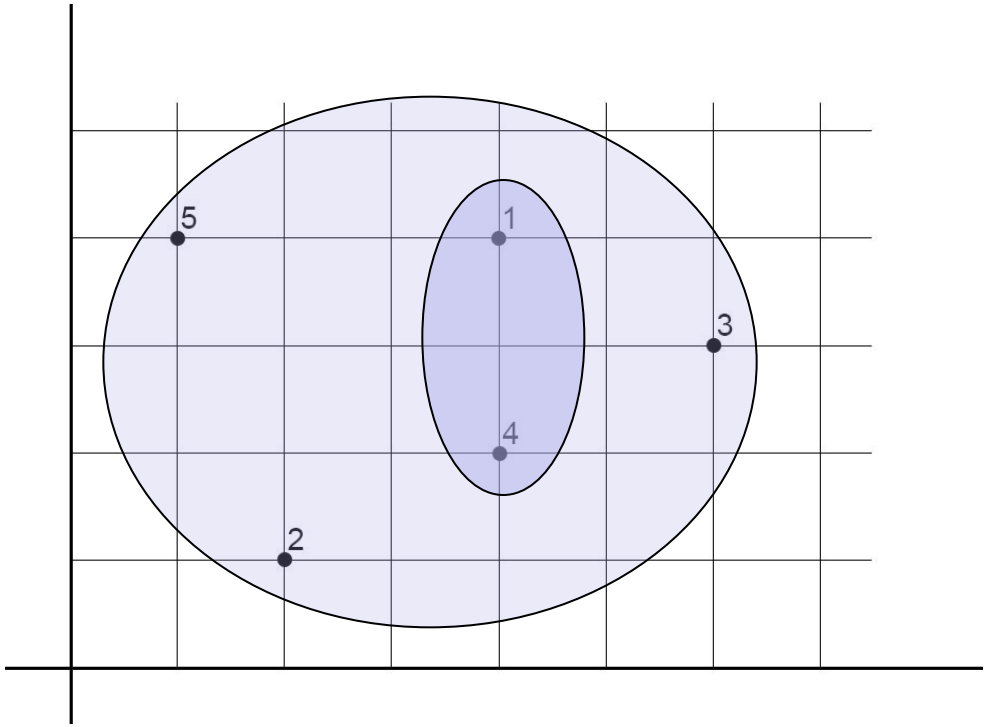
(4) is closer than 5 for (2), so set $\lambda_3(2) = 3$ and $\pi_3(2) = 4$

(4) is closer than ∞ for (3), so set $\lambda_3(3) = 3$ and $\pi_3(3) = 4$

SLINK algorithm

n = 4

	1	2	3	4	5	
λ	2	3	3	~~∞~~ 3	∞	
π	4	~~4~~ 5	~~4~~ 5	~~4~~ 5	5	
M	3	4	6	~~5~~ 3	–	



Set:

$\lambda_4(5) = \infty$

$\pi_4(5) = n+1 = 5$

(5) is not closer than 2 for (1), but (4) can reach (5) in 3 instead of 5. Update M(4) = 3

(5) is not closer than 3 for (2), and is not an improvement for M(4).

(5) is at same distance for (3), so only set $\pi_4(3) = 5$

(5) is closer than ∞ for (4), so set $\lambda_4(4) = 3$ and $\pi_4(4) = 5$

Now the point (2) has $\lambda_4(2) \geq \lambda_4(\pi_4(2))$ so we can merge the two clusters by setting $\pi_4(2) = 5$. Same for point (3)

Classifiability

Jardine and Sibson (1971) suggest the use of this quantity as a measure of how amenable to single-link classification the data is.

$$\Delta_1 = \frac{\sum_{i<j}(d(i,j) - d^*(i,j))}{\sum_{i<j} d(i,j)}$$

Where:

$$d^*(i,j) = inf\ \{h : (i,j) \in c(h)\}$$

The lower the Δ_1 the better.

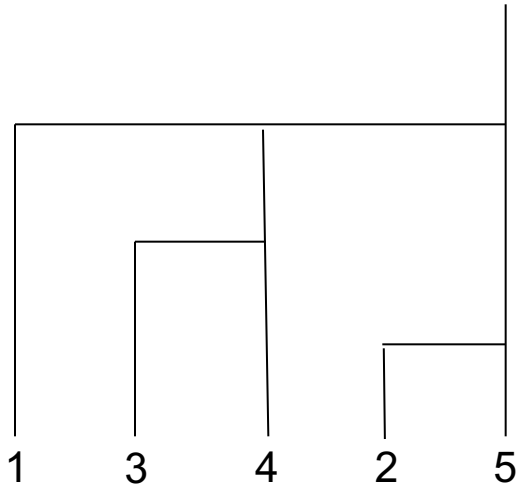
Presentation of results

The pointer representation of a dendogram is not particulary helpful from the user's point of view and it's desiderable to convert it into another representation called the **packed representation** for output, from which a tree-diagram can be easily drawn.

E.g.

N	1	2	3	4	5
λ	3	1	2	3	∞
π	5	5	4	5	5

We want to draw something like this:

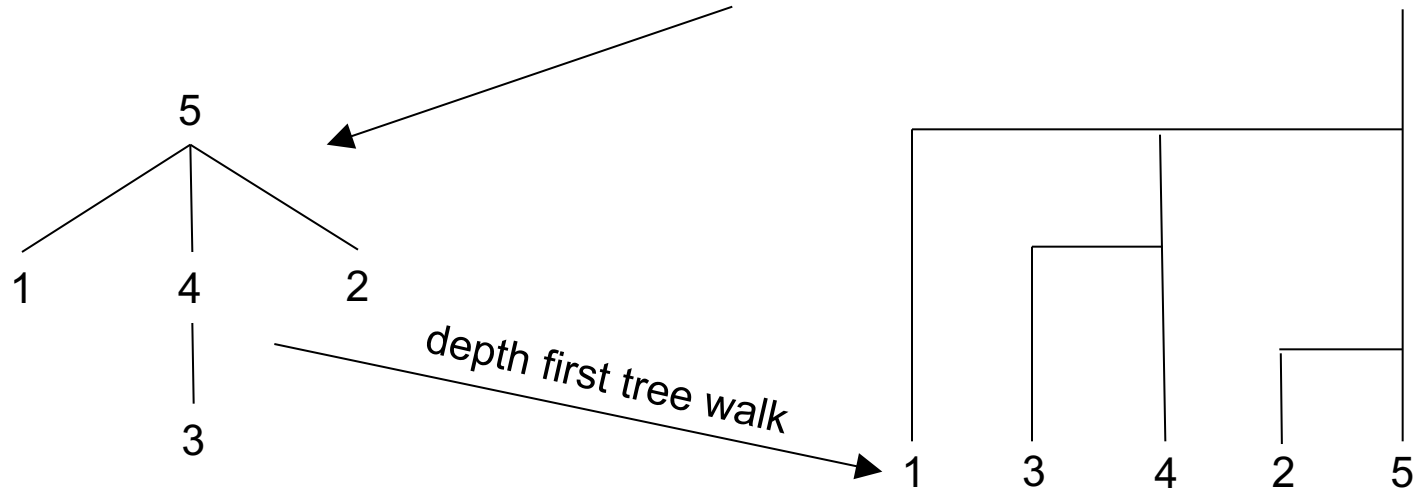


Presentation of results

N	1	2	3	4	5
λ	3	1	2	3	∞
π	5	5	4	5	5

It's possible to do this in $O(N^2)$ by building a tree, where:

- the last element is the root;
- if $\pi(i) = j$, then j is the parent of i, keeping children sorted according to decreasing λ. If we sort the indexes before creating the tree, we'll get the children already in order.



Presentation of results

Another way to build the dendogram is to create the **linkage matrix** and then pass it to plotting libraries that will carry the load of drawing a clear dendrogram. E.g. from *scipy.cluster.hierarchy* we can create a dendrogram using the *dendrogram* function together with the linkage matrix and then plot it with *matplotlib.pyplot*.

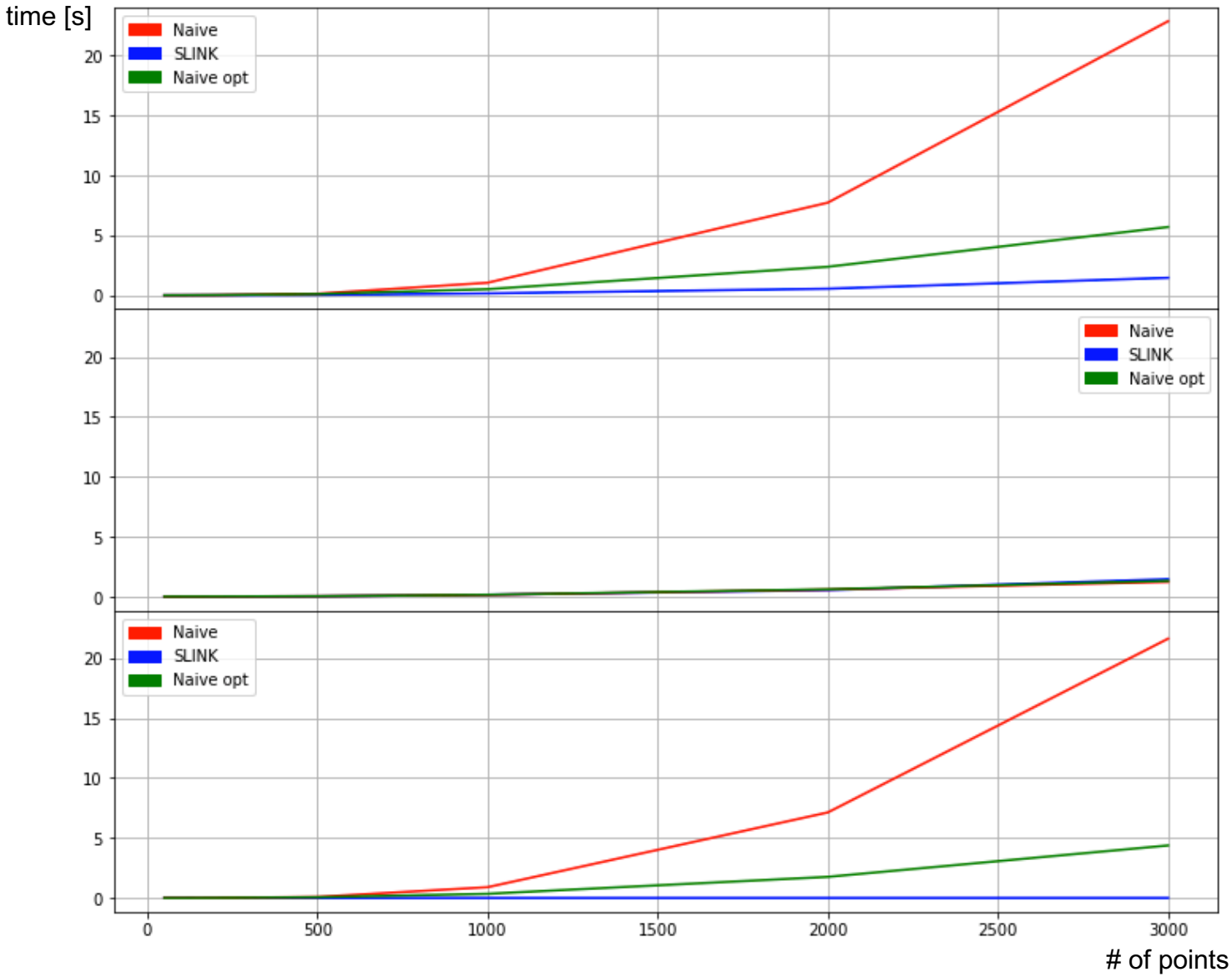
N	1	2	3	4	5
λ	3	1	2	3	∞
π	5	5	4	5	5

Benchmarks

We tested the three algorithms on datasets:
- with total number of points between 50 and 3000
- with point dimensions between 2 and 200

the results are...

Benchmarks



Total time

- SLINK: time to build the pointer representation
- Naive & Naive opt: time to build the dissimilarity matrix

- SLINK: time to convert pointer representation to linkage matrix
- Naive & Naive opt: time to create the linkage matrix iteratively from the dissimilarity matrix

Benchmarks

The naive algorithms spend the majority of the time working on the dissimilarity matrix rather than building it.

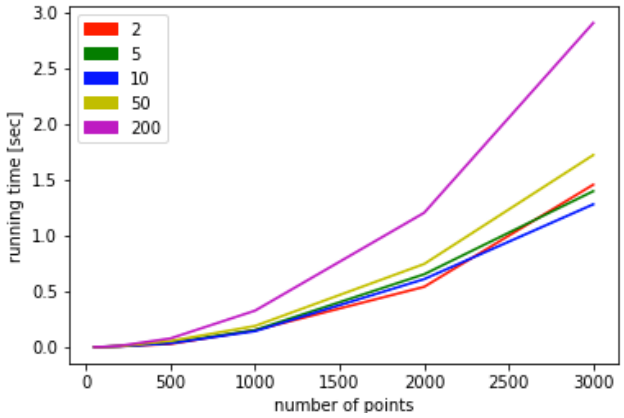
However, SLINK terminates its execution when the other algorithms are starting to work on the dissimilarity matrix!

# of points

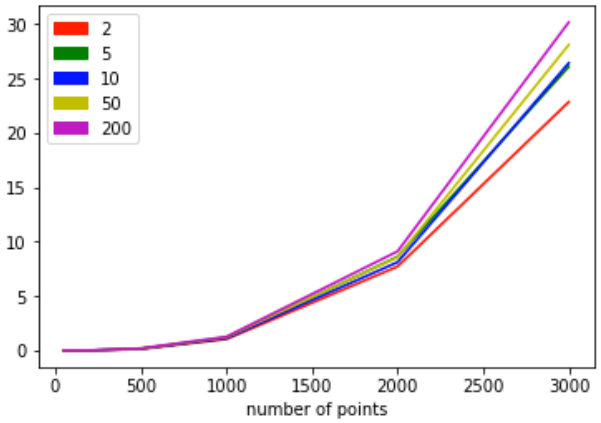
Benchmarks

Effect of data dimensionality over running time

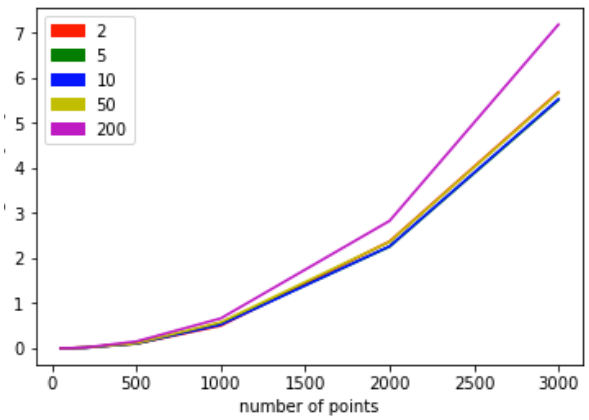
SLINK

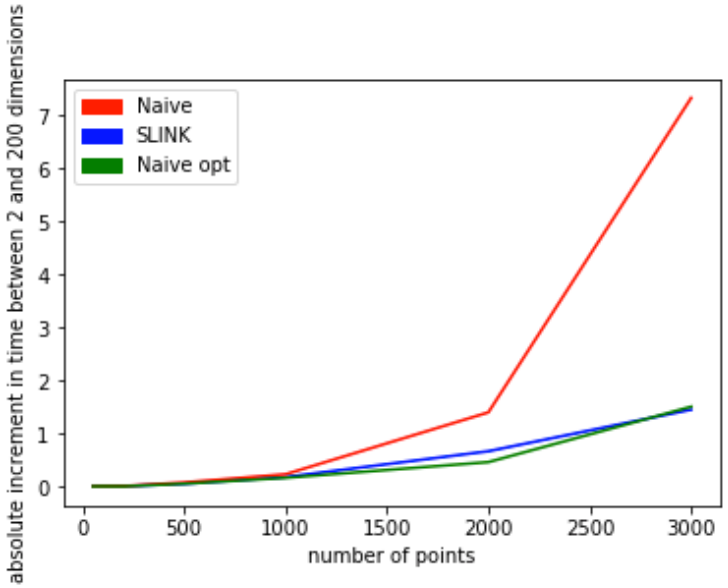


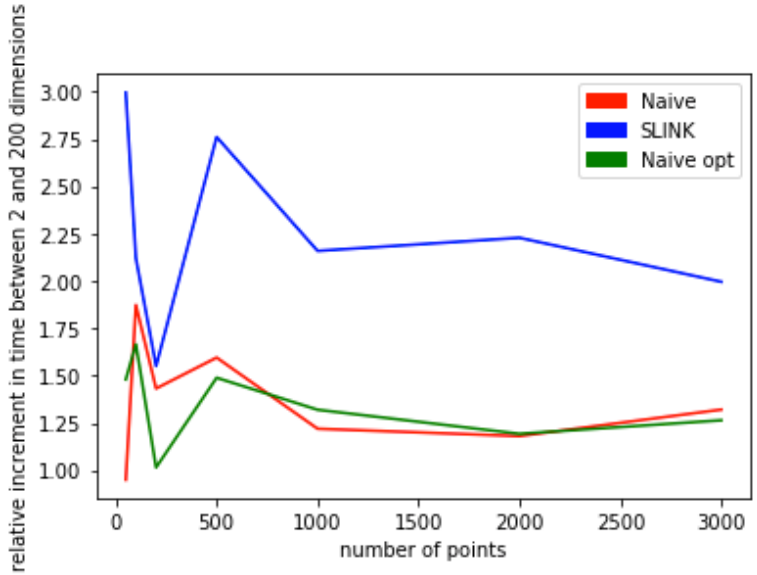
Naive



Naive optimized







Benchmarks

Each algorithm computes once the distance between each pair of distinct points. SLINK has the biggest relative increase of total running time simply because it’s the fastest wrt total time. Other variations may be dued to locality of data and instructions.

Conclusions

- The tests confirm the theoretical performance of the algorithm.
- **SLINK** is a huge improvement over the naive implementation and allows single linkage clustering to work on bigger datasets than before.

Source code: https://github.com/battuzz/slink