Ultrametric fitting by gradient descent

Benjamin Perret

Joint work with Giovanni Chierchia

Université Paris Est, LIGM, CNRS, ESIEE Paris

Publication accepted to NeurIPS 2019

In brief:

- Problem:
  - Most hierarchical clustering method are defined procedurally
  - Optimization of hierarchical cost functions are usually formulated as NP-Hard combinatorial problems

- We propose:
  - A continuous method (aka gradient descend) for hierarchical clustering optimization
  - A differentiable « ultrametric layer » that transforms any dissimilarity into an ultrametric (aka hierachical clustering)

1. Introduction

What is an ultrametric ? (1/2)

- It is a distance that satisfies the ultrametric inequality

$$U \text{ is an ultrametric} \quad \Leftrightarrow \quad \begin{cases} U_{ii} = 0 & \forall i \\ U_{ij} \geq 0 & \forall i, j \\ U_{ij} = U_{ji} & \forall i, j \\ U_{ij} \leq \max\{U_{ik}, U_{kj}\} & \forall i, j, k. \end{cases}$$

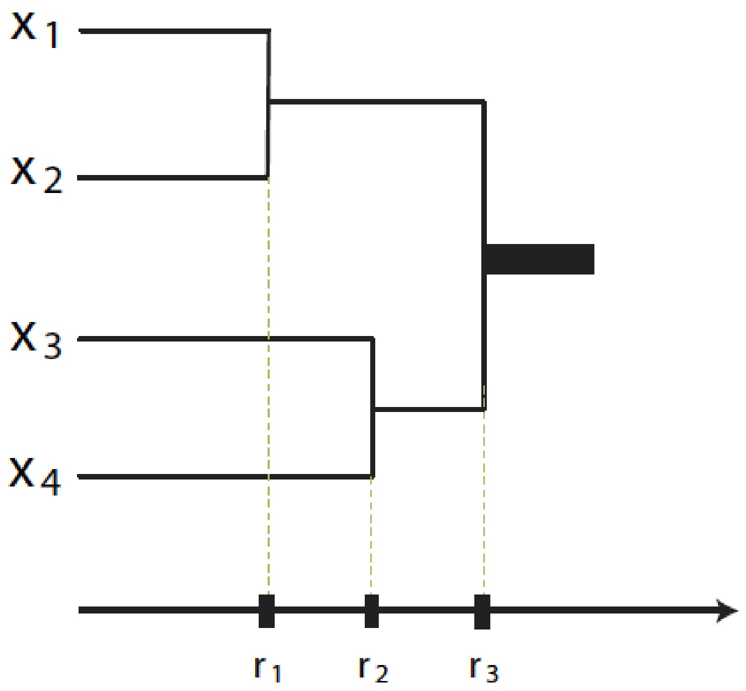
For every triplet (i,j,k), at least two of them are equal.

$$((u_\theta)) = \begin{array}{c c} & \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} & \begin{pmatrix} 0 & r_1 & r_3 & r_3 \\ r_1 & 0 & r_3 & r_3 \\ r_3 & r_3 & 0 & r_2 \\ r_3 & r_3 & r_2 & 0 \end{pmatrix} \end{array}$$

Ultrametric distance matrix

What is an ultrametric ? (2/2)

- Dual representation of hierarchical clusterings



$$((u_\theta)) = \begin{array}{c c} & \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} & \begin{pmatrix} 0 & r_1 & r_3 & r_3 \\ r_1 & 0 & r_3 & r_3 \\ r_3 & r_3 & 0 & r_2 \\ r_3 & r_3 & r_2 & 0 \end{pmatrix} \end{array}$$

Dendrogram:

- leaves represent data
- Inner nodes represent successive merging of clusters
- The distance between two leaves is given by their lowest common ancestor

Ultrametric fitting

- **Goal:** Find an ultrametric that "best" represents the dissimilarity data
  - ***Input*** → *Undirected graph with dissimilarity edge weights*
  - ***Input*** → *Cost function on the produced ultrametric*
  - ***Output*** → *Ultrametric edge weights*

2. Proposed approach

Optimization framework (1/4)

- Constrained optimization problem over a continuous domain

$$\underset{u \in \mathcal{W}}{\text{minimize}} \quad J(u; w) \quad \text{s.t.} \quad u \text{ is an ultrametric on } \mathcal{G}$$

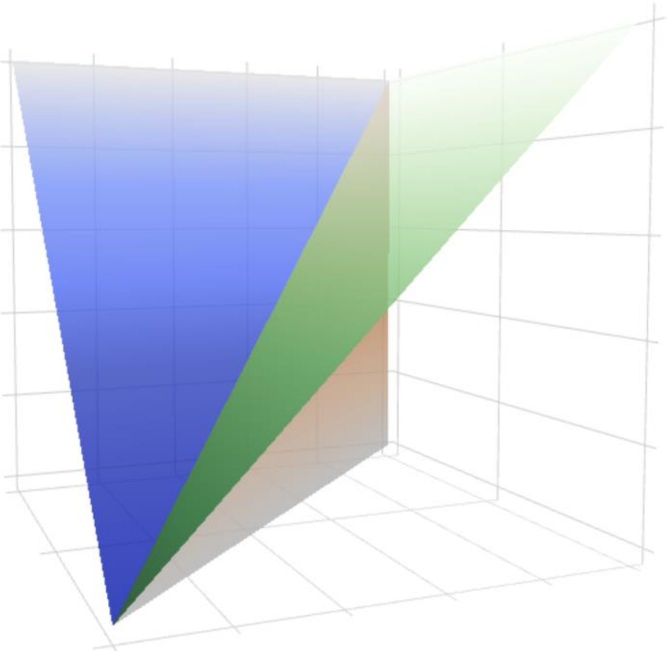
Input dissimilarity
(edge weights)

- U is an ultrametric on the graph G:

$$(\forall C \in \mathcal{C}, \forall e \in C) \qquad u(e) \leq \max_{e' \in C \setminus \{e\}} u(e')$$

Set of cycles of G

- This constraint is non-convex



Subset of triplets that satisfy the ultrametric inequality

Optimization framework (2/4)

- Reformulation with implicit constraint

$$\underset{\tilde{w} \in \mathcal{W}}{\text{minimize}} \; J\big(\Phi_{\mathcal{G}}(\tilde{w});\, w\big)$$

- Where $\Phi_{\mathcal{G}}$ is
  - an operator from the set of edge dissimilarities to the set of ultrametric
  - differentiable to allow for gradient descent
- $\tilde{w}$ is not an ultrametric anymore but $\Phi_{\mathcal{G}}(\tilde{w})$ is one

Optimization framework (3/4)

- Does such $\Phi_{\mathcal{G}}$ exists ?
- Good news: one very standard operator does it (so standard it has a lot of names):
  - Quasi-flat zone hierarchy/Alpha-Tree
  - Sub-dominant ultrametric
  - Single-linkage clustering
  - Min-max distance

Optimization framework (4/4)

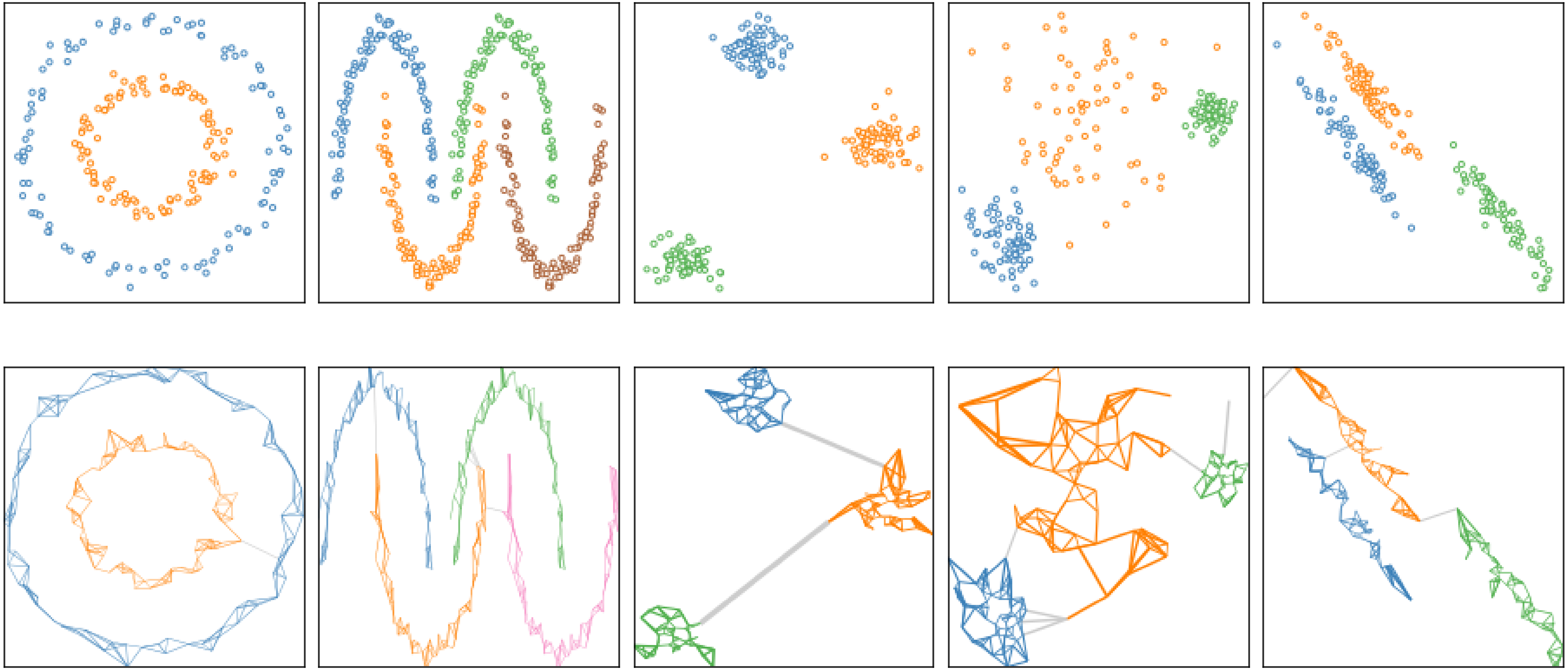
- Wait… QFZ is (sub-)differentiable ?
- Intuitive idea: the ultrametric associated to QFZ can be computed with the Floyd-Warshall algorithm:

$$(\forall k \in \{1, \ldots, N\}) \qquad U_{ij}^{[k+1]} = \min\left\{U_{ij}^{[k]}, \max\left(U_{ik}^{[k]}, U_{kj}^{[k]}\right)\right\}.$$

  - This is just *like* a sequence of max-pooling
- More efficient implementation in practice:
  - Compute QFZ: O(n*log(n))
  - Compute the ultrametric/saliency map associated to this hierarchy: O(n)
  - At the end: this is just playing with indices => automatic differentiation

3. Cost functions

Example - Toy datasets

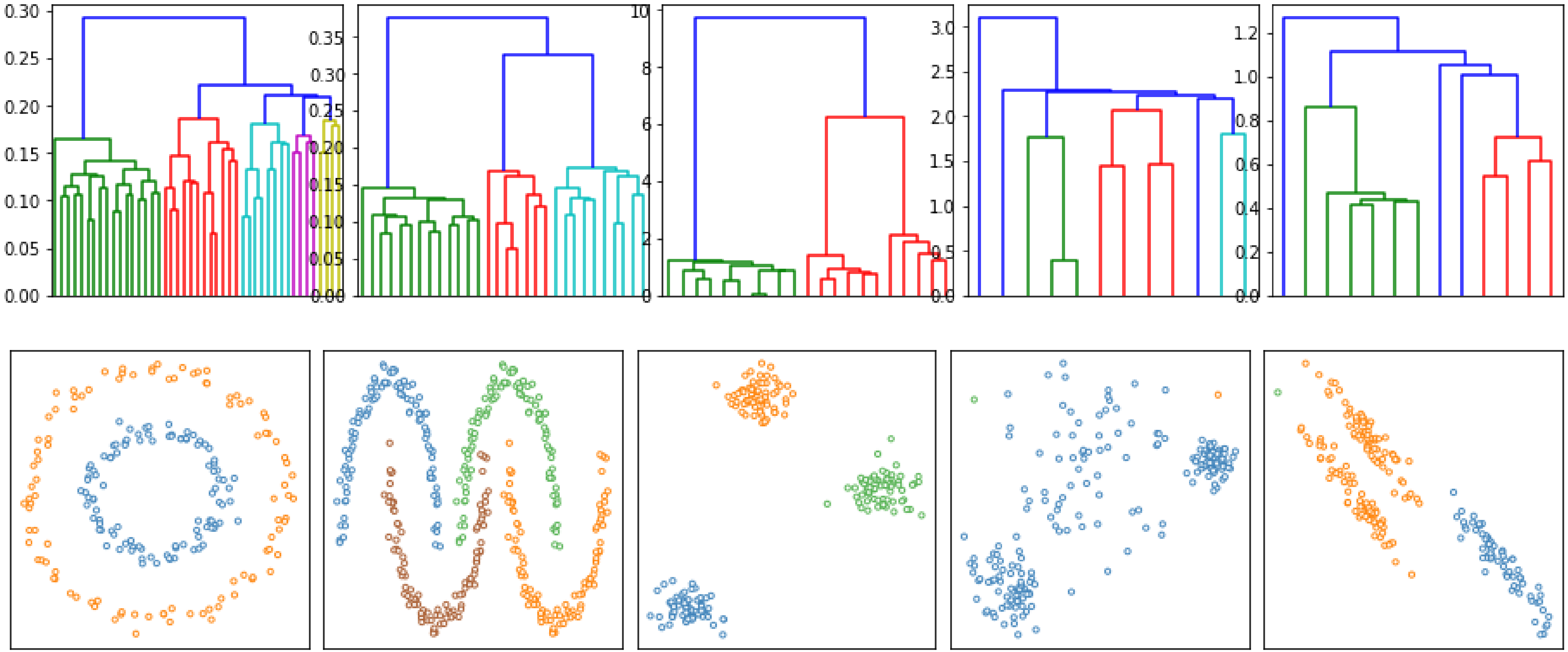


Closest ultrametric fitting

- Find the “closest” ultrametric to the given dissimilarity graph
  - *Ie. Find the ultrametric that minimizes the L-p distance to the input dissimilarity graph*

$$J_{\text{closest}}(u; w) = \sum_{e \in E} \Big(u(e) - w(e) \Big)^2$$

Example - Closest ultrametric fitting J_{closest}



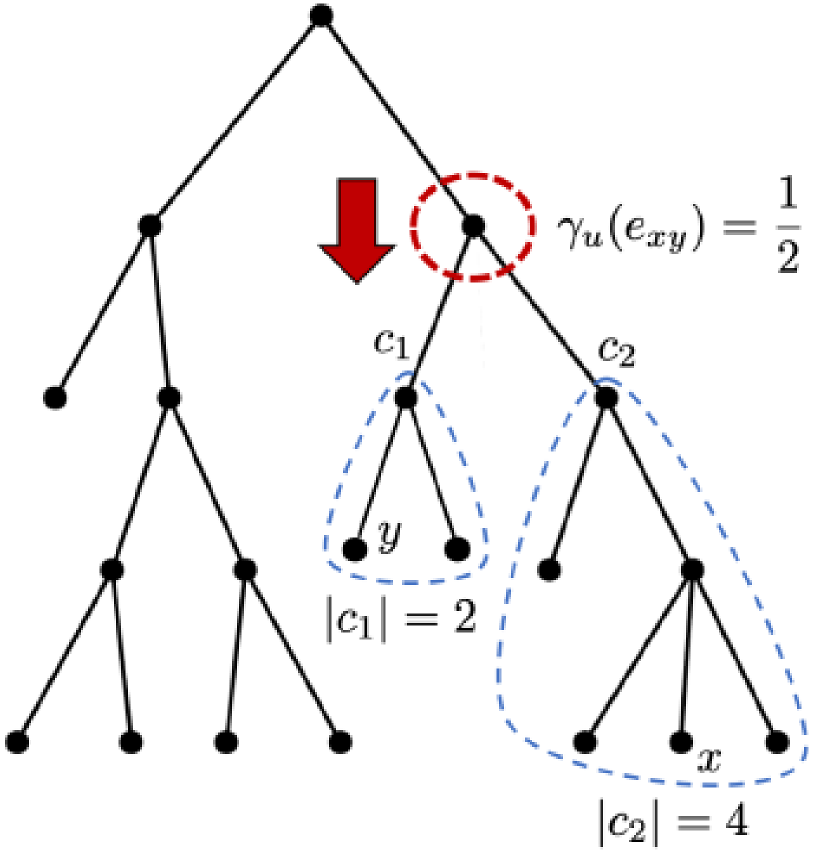
Regularized ultrametric fitting

- **Regularization →** Push down the nodes with few descendant leaves

$$J_{\text{size}}(u) = \sum_{e_{xy} \in E} \frac{u(e_{xy})}{\gamma_u(\text{lca}_u(x, y))}$$

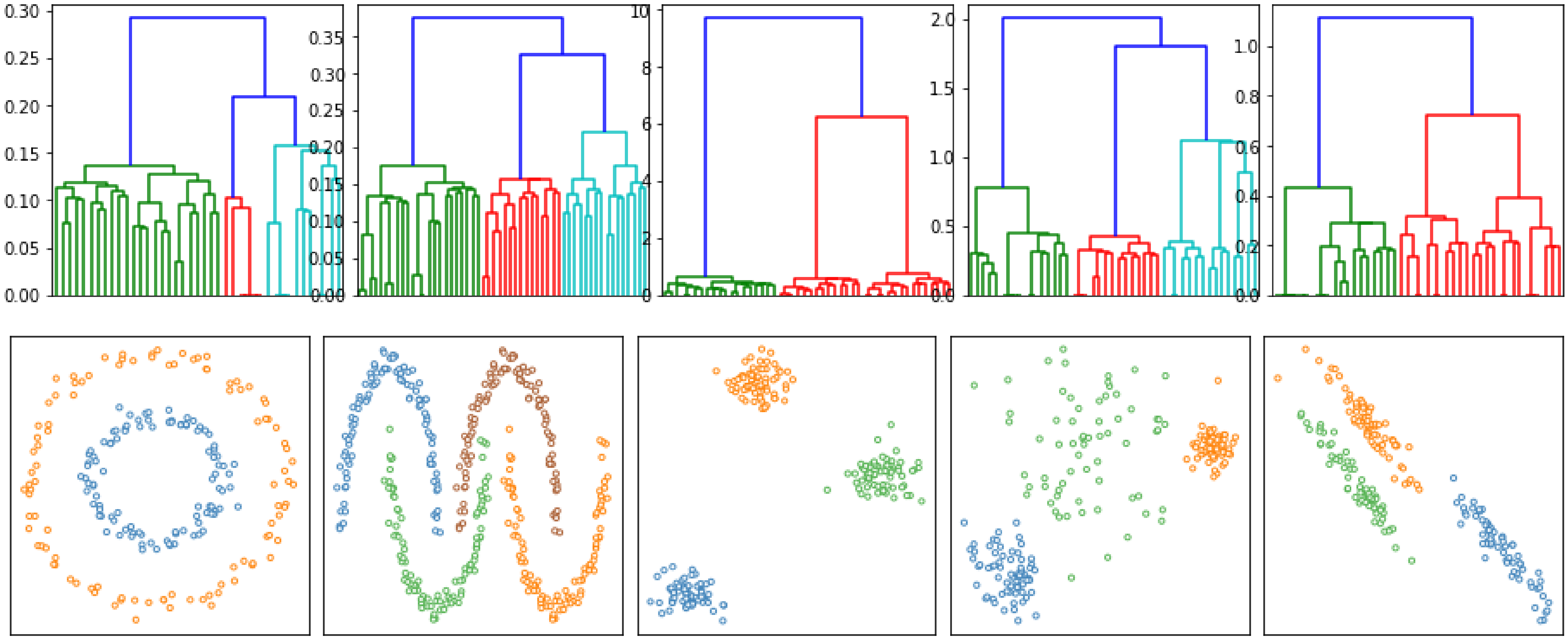
with

$$\gamma_u(n) = \min\{|c|, c \in \text{Children}_\text{u}(\text{n})\}$$



Example - Regularized ultrametric fitting

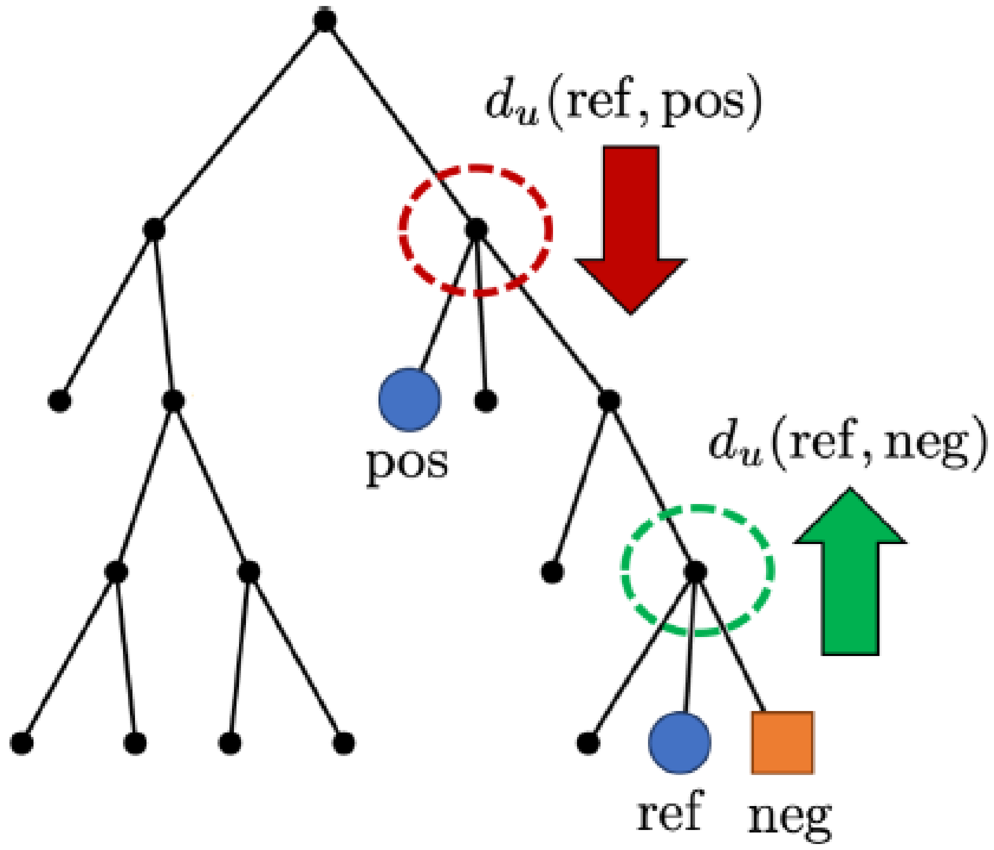
$$J_{\text{closest}} + J_{\text{size}}$$



Triplet-based ultrametric fitting

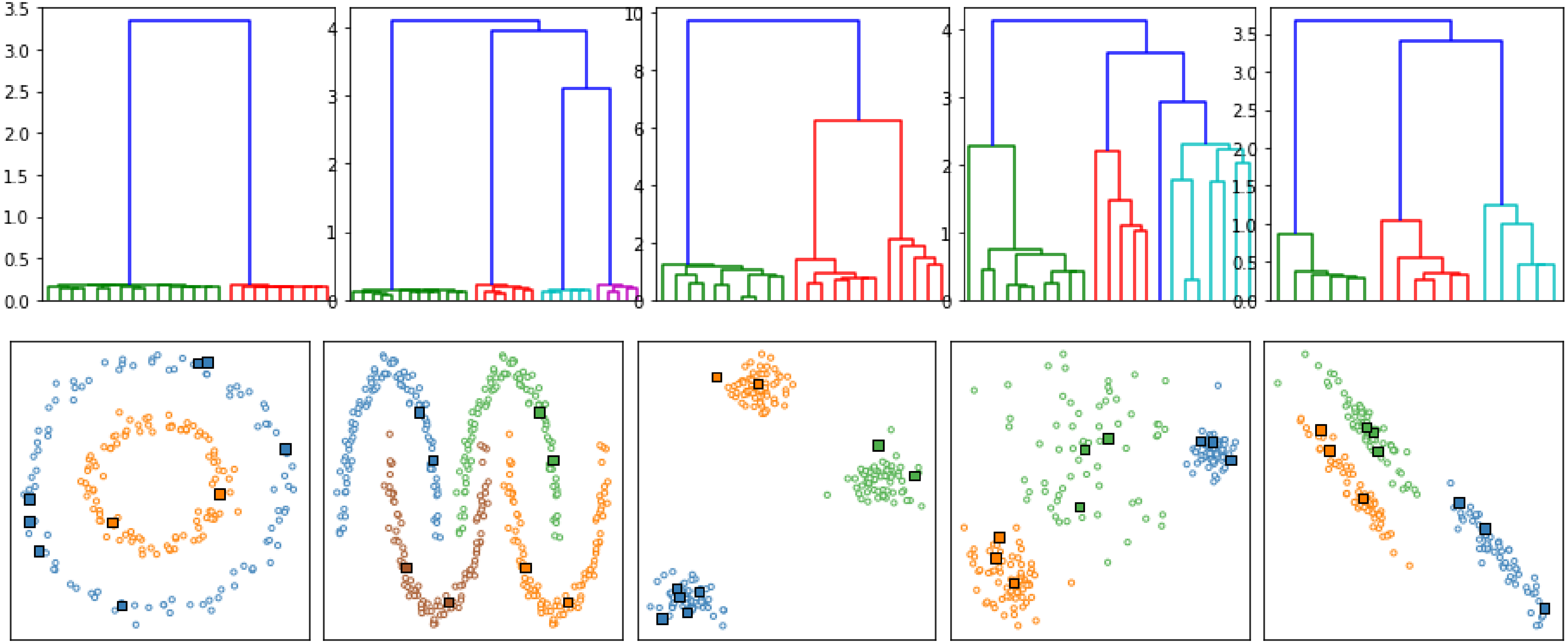
$$J_{\text{triplet}}(u) = \sum_{(\text{ref},\text{pos},\text{neg}) \in \mathcal{T}} \max\{0, \alpha + d_u(\text{ref},\text{pos}) - d_u(\text{ref},\text{neg})\}$$

- **Triplet loss (semi-supervised)**
  - Push down the edges between points in the same class
  - Push up the edges between points in different classes



Example – Triplet-based ultrametric fitting

$$J_{\text{closest}} + J_{\text{triplet}}$$



Dasgupta's cost function

- Well known cost function for hierarchical clustering in the algorithmic community

$$J_{\text{Dasgupta}}(u; w) = \sum_{e_{xy} \in E} \frac{|\text{lca}_u(x, y)|}{w(e_{x,y})}$$

- Problem: the cardinal of a set is not a differentiable function…
- Solution: relax the definition of the size of a node

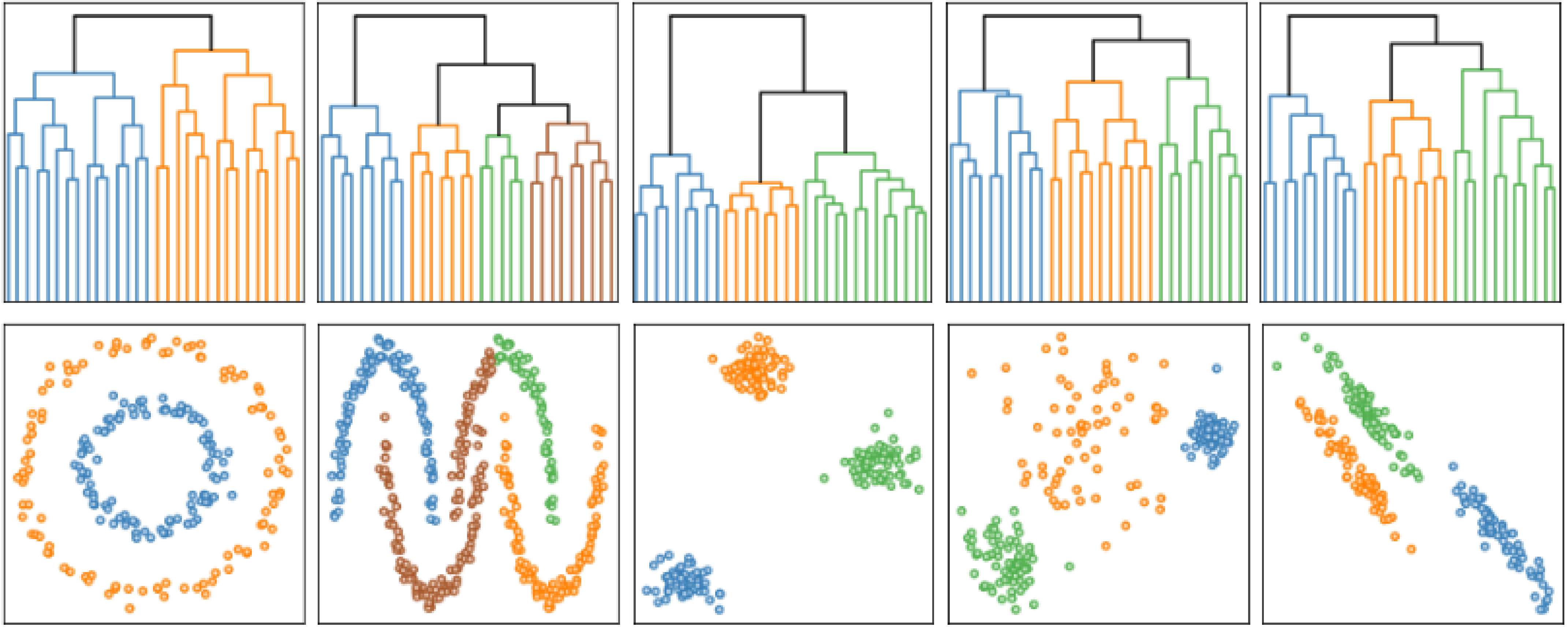
$$|n| = \sum_{y \in V} H(\text{alt}_u(n) - d_u(x, y))$$

Where H is the Heaviside function

  - Replace H by a continuous approximation (sigmoid) to get a differentiable area

Example – Dasgupta's optimization

J_{Dasgupta}



4. Validation

Validation of the optimization method

- Problem:
  - Non convex optimization
  - Gradient descend method offers no global guaranty
  - Do we manage to reach "good" solutions ?

- Comparison with more specific method with proven guaranties:
  - *"Closest Ultrametric via Cutting Plane"* (Yarkony & Fowlkes, NIPS 2015), almost exact solution to the closest ultrametric problem on planar graphs
  - "Linkage ++" (Cohen-Addad et al., NIPS 2017), provides a O(1) approximation of optimal Dasgupta's solution with high probability

Closest Ultrametric via Cutting Plane

- Generate many planar graphs of various sizes

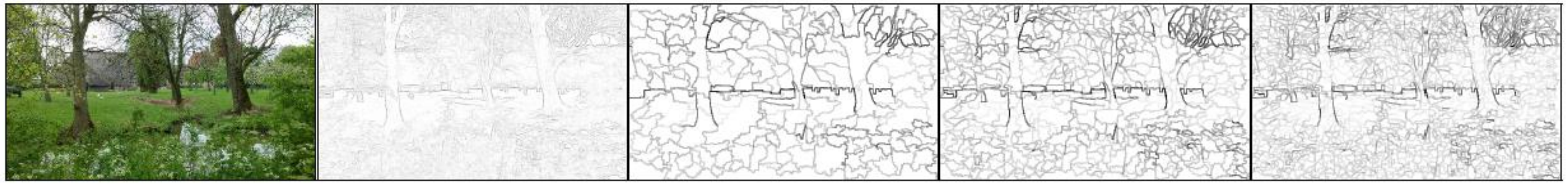
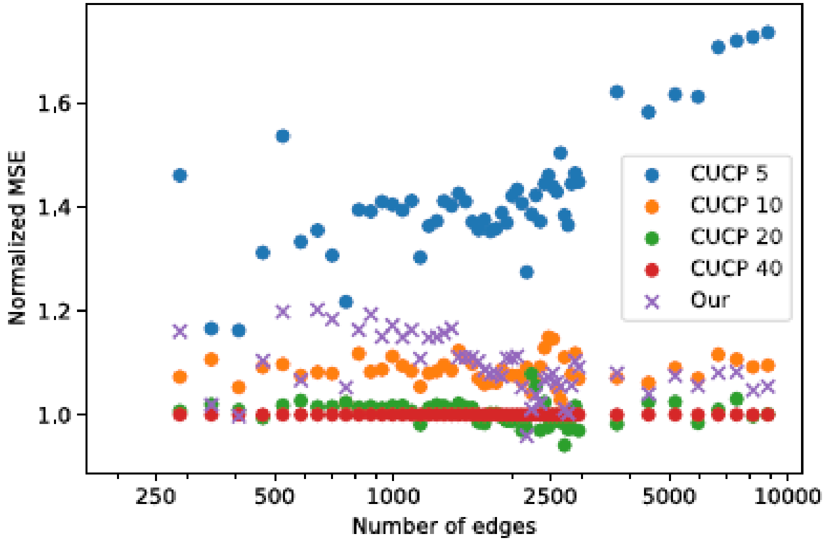
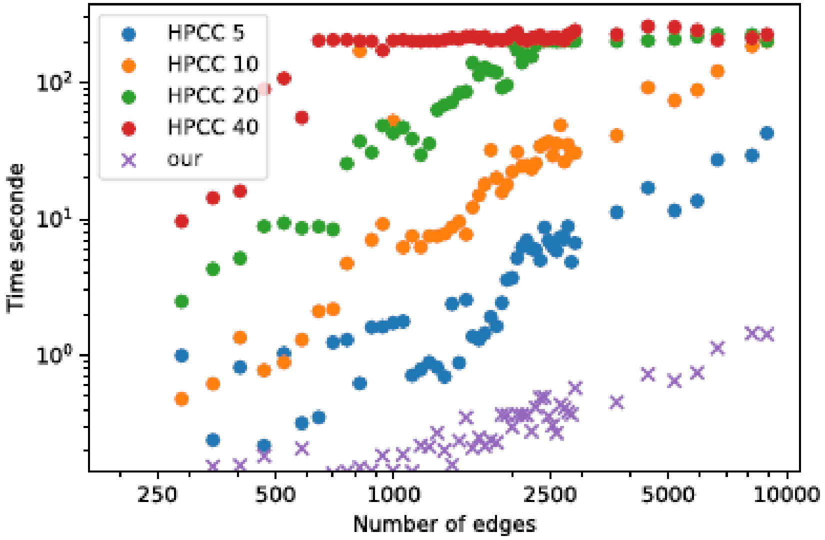


Figure 4: Test image, its gradient, and superpixel contour weights with 525, 1526, and 4434 edges.

- Compare CUCP solutions and runtime



(a) Mean square error

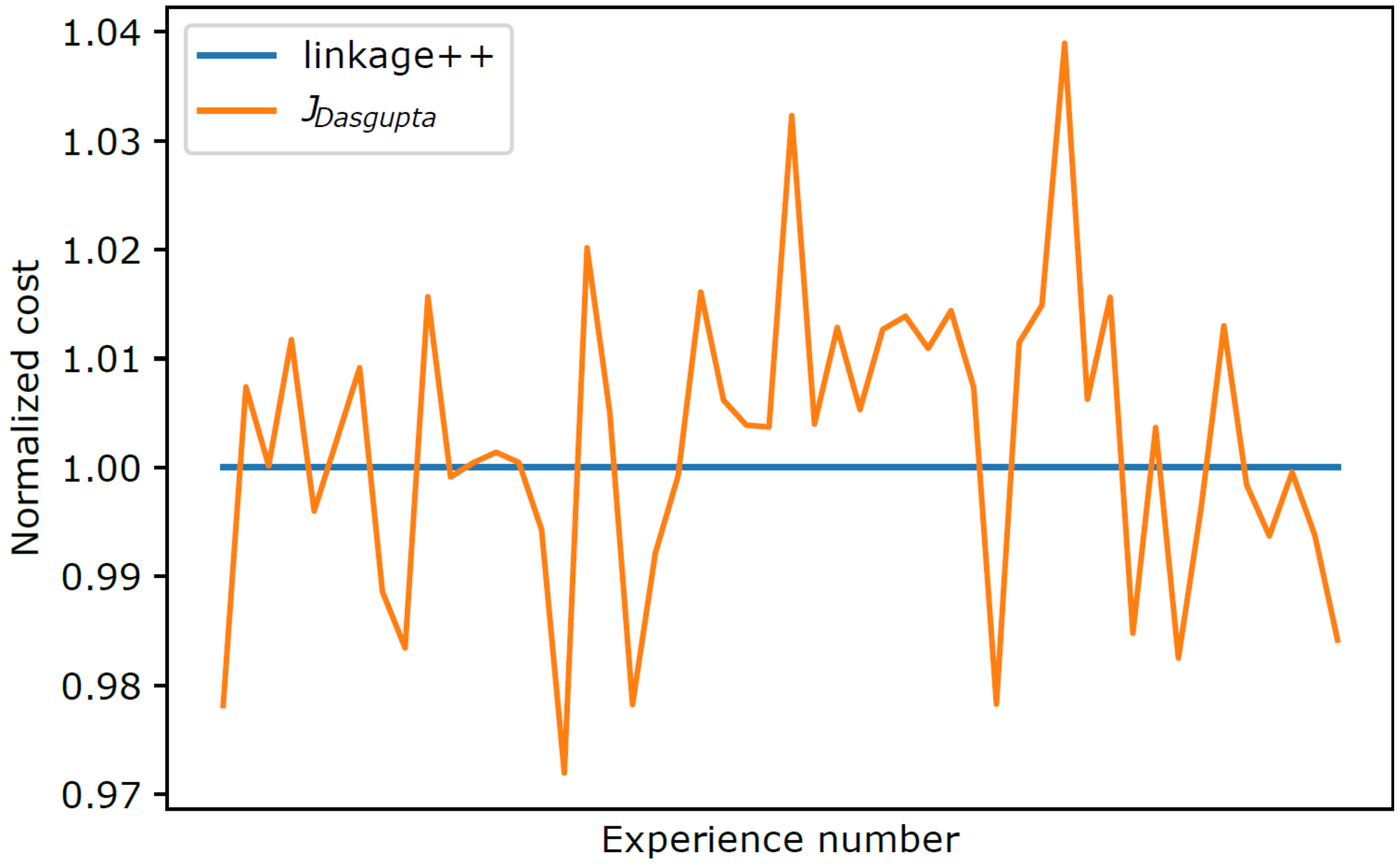


(b) Computation time

Not far from CUCP but much faster and scalable

Dasgupta's cost

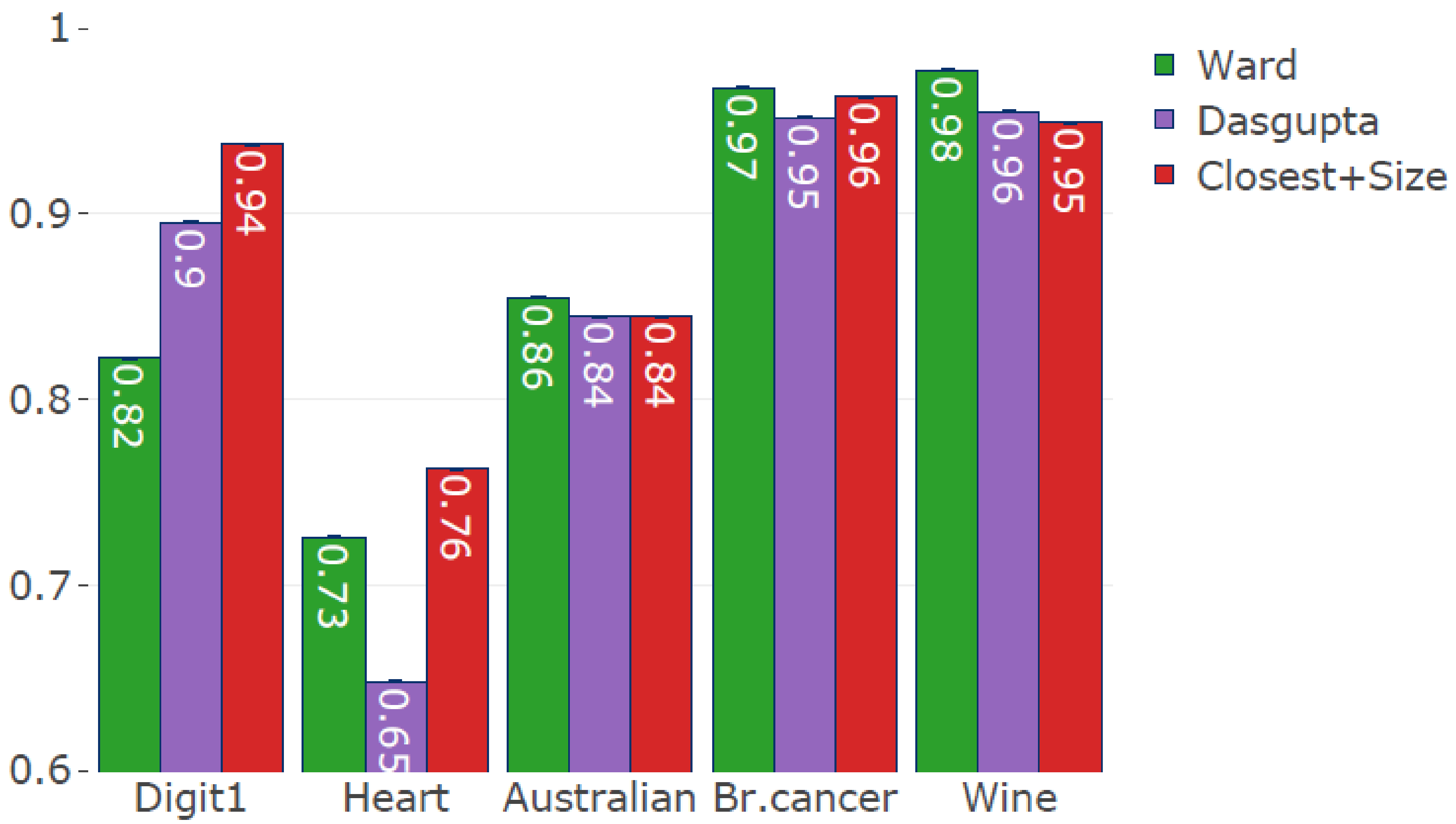
- Generate many blob like graphs with various sizes number of clusters



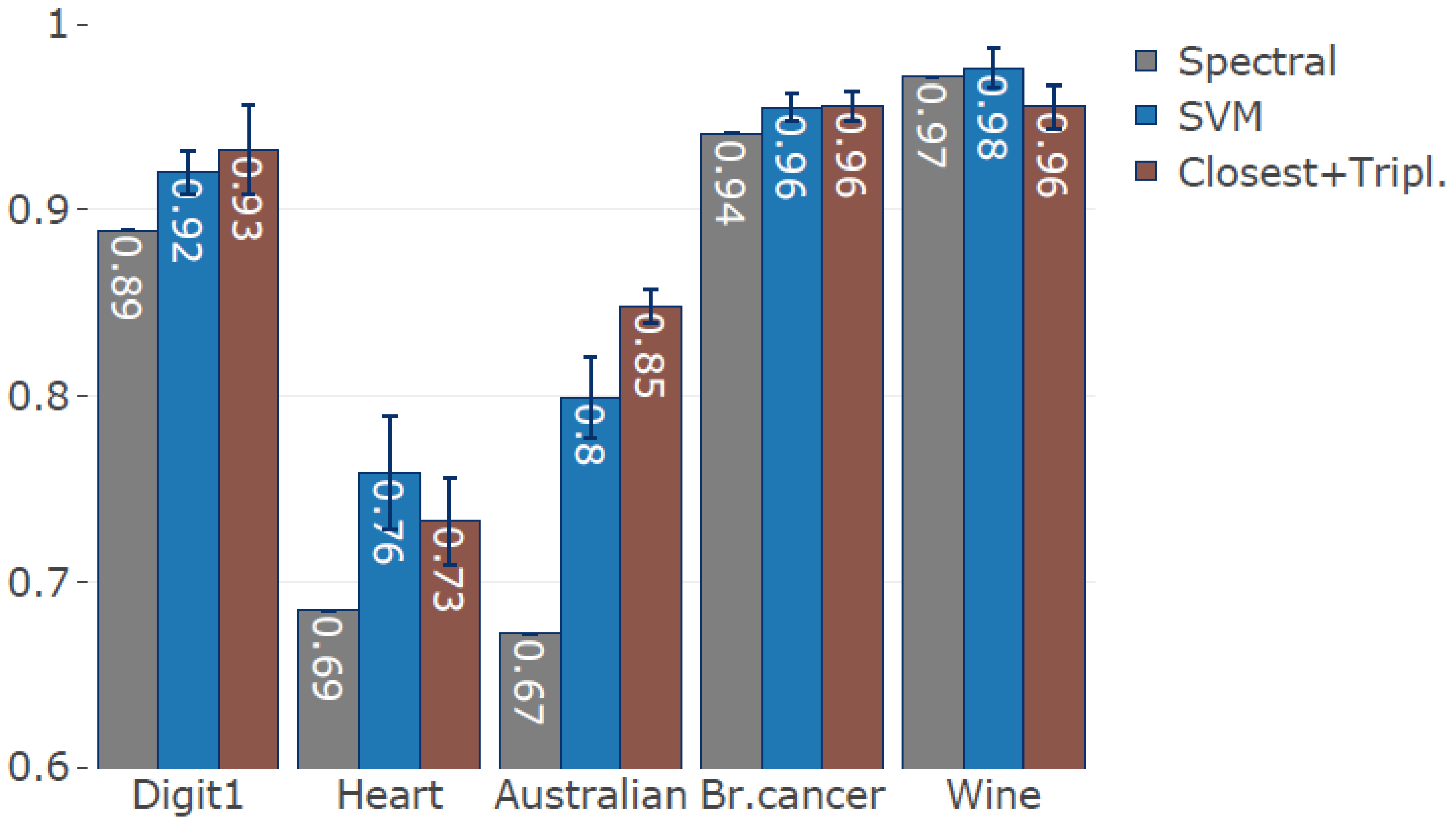
Comparable results

5. Demonstration on clustering

Clustering on real datasets



Semi-supervised classification on real data



6. Conclusion

Take-away message

- Ultrametric fitting
  - General optimization framework
  - Flexible choice of the cost function

- Hierarchical clustering
  - First results: performance comparable to well-established methods (Ward linkage & SVM)
  - Planning to integrate graph estimation/learning via deep learning

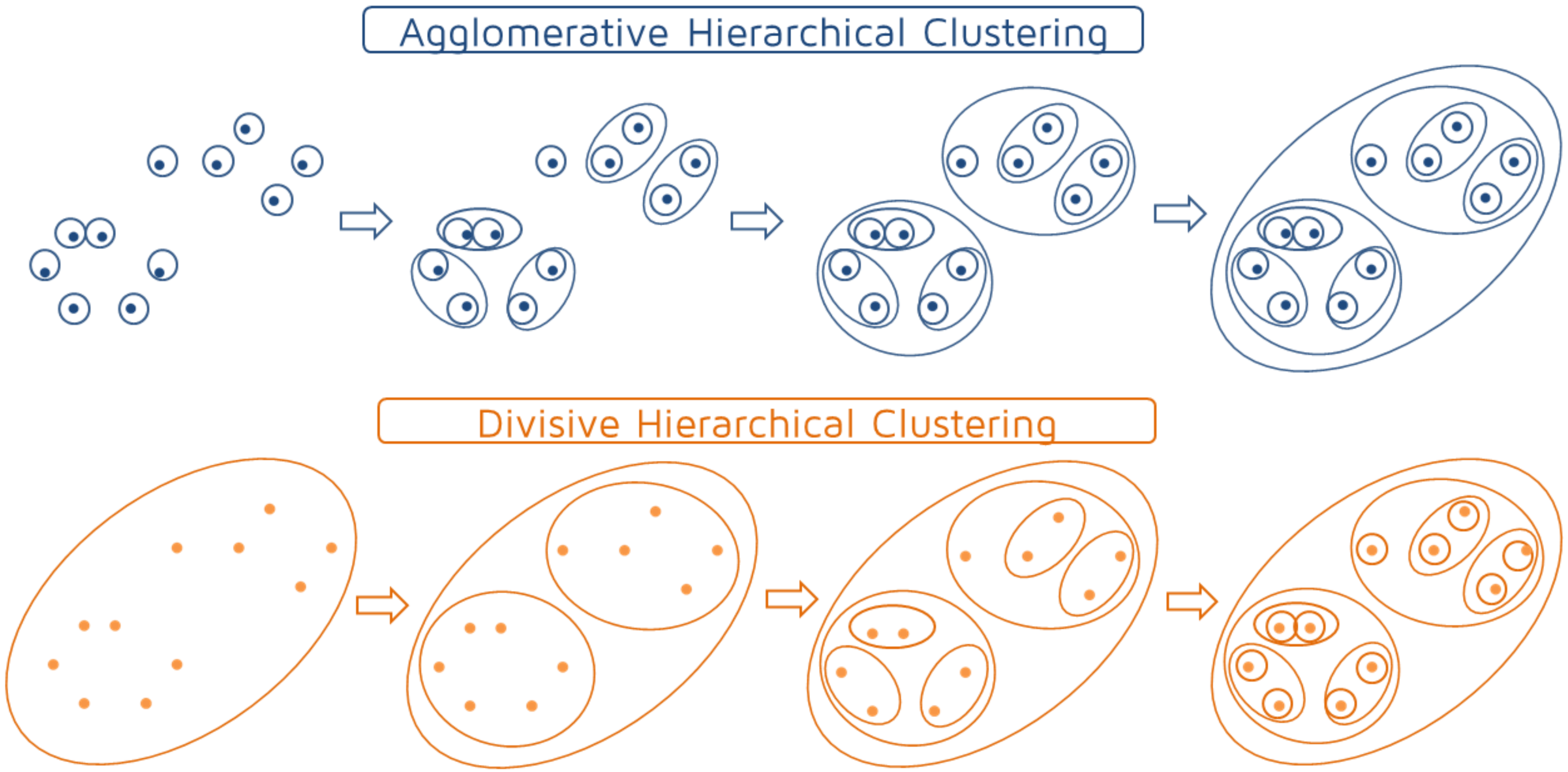
Thank you

- Article: https://arxiv.org/abs/1905.10566

- Pytorch implementation available online: https://github.com/PerretB/ultrametric-fitting

Agglomerative vs divisive approaches

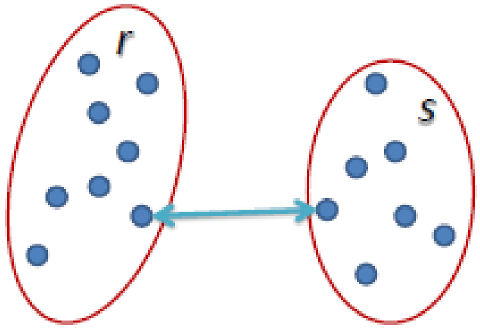
- There exist two large families of algorithms to build an ultrametric.



Linkage criteria in agglomerative clustering

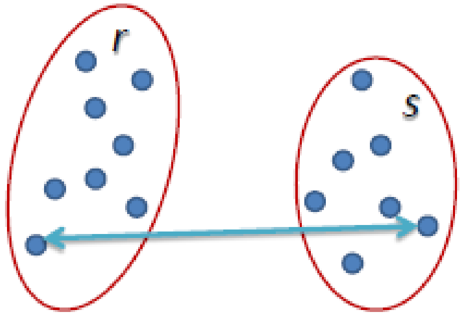
- Agglomerative methods are governed by the linkage criteria

Single linkage



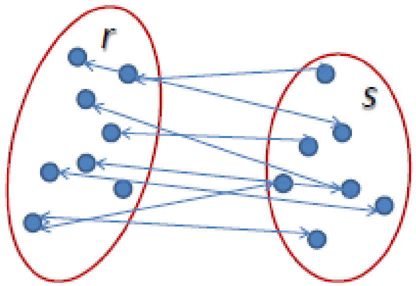
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Complete linkage



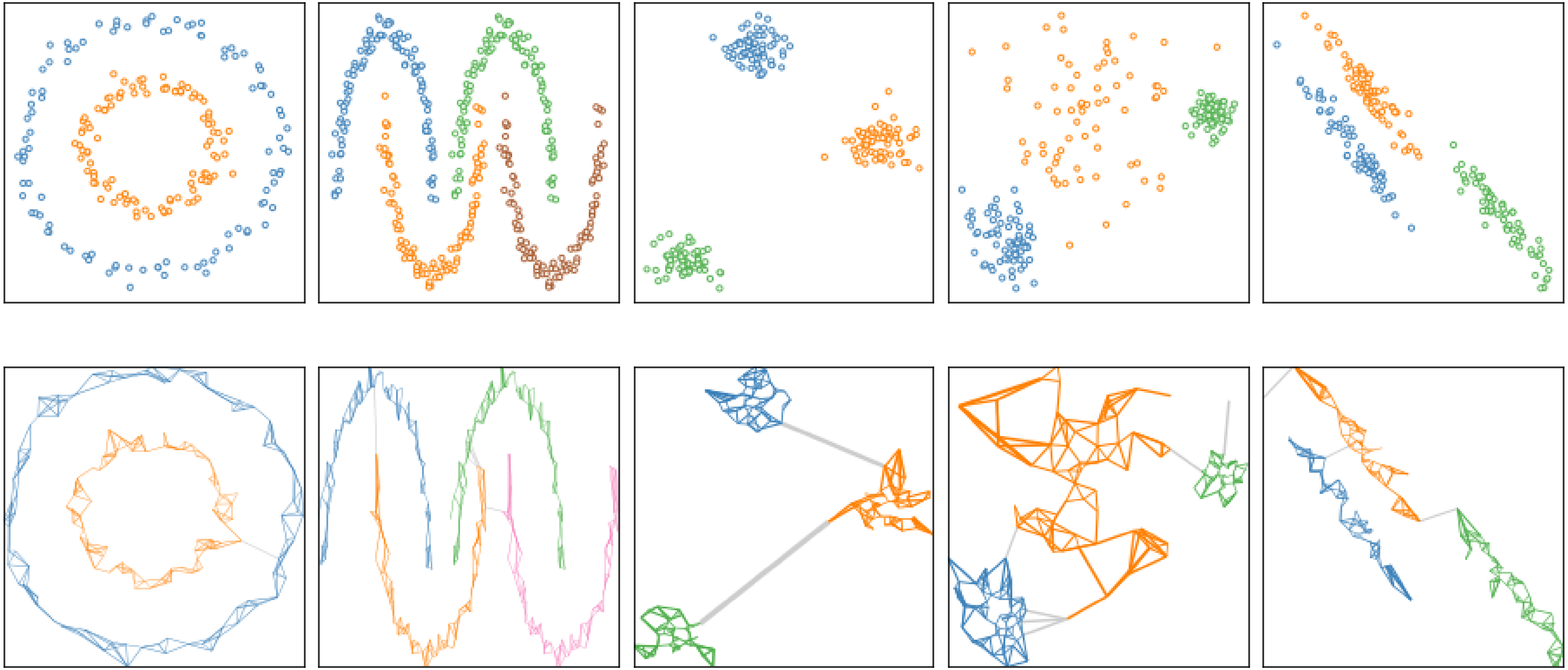
$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

Average linkage

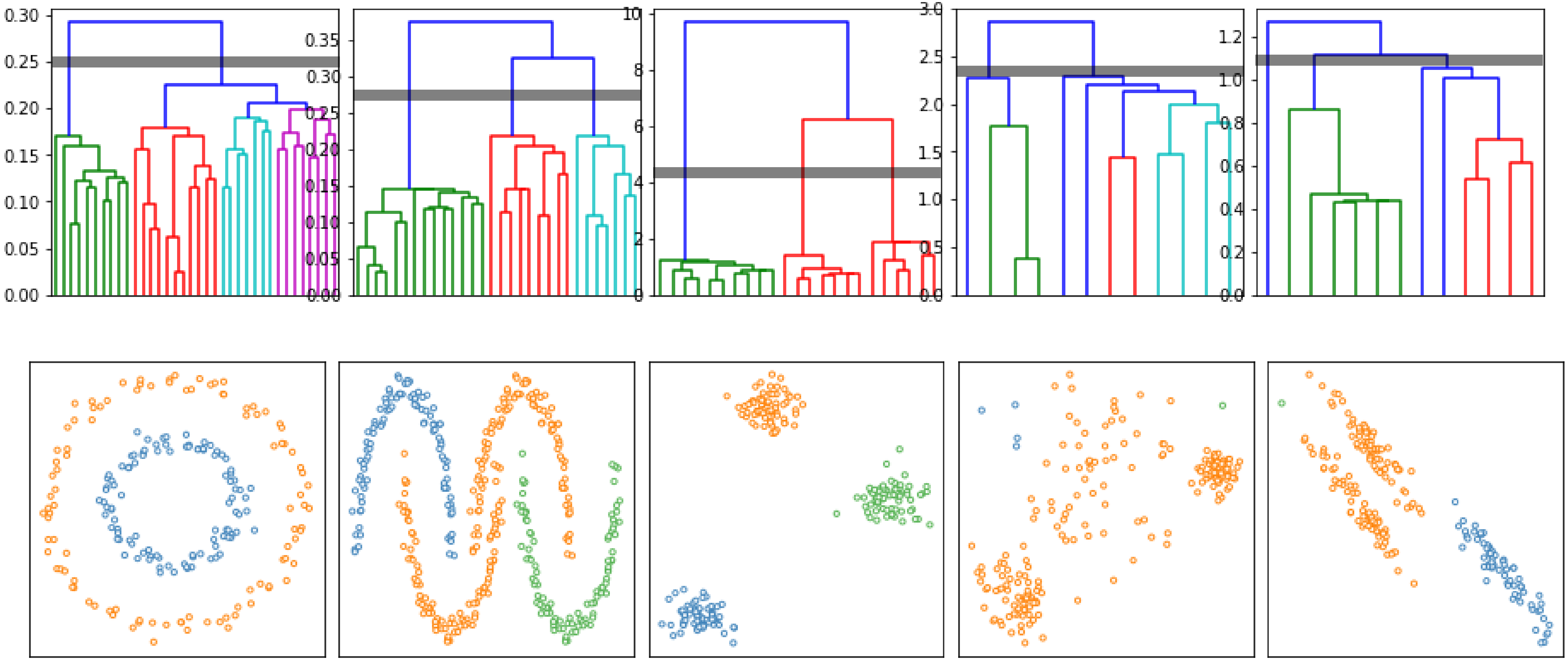


$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Example - Toy datasets



Example - Average-linkage clustering



Example – Single-linkage clustering