



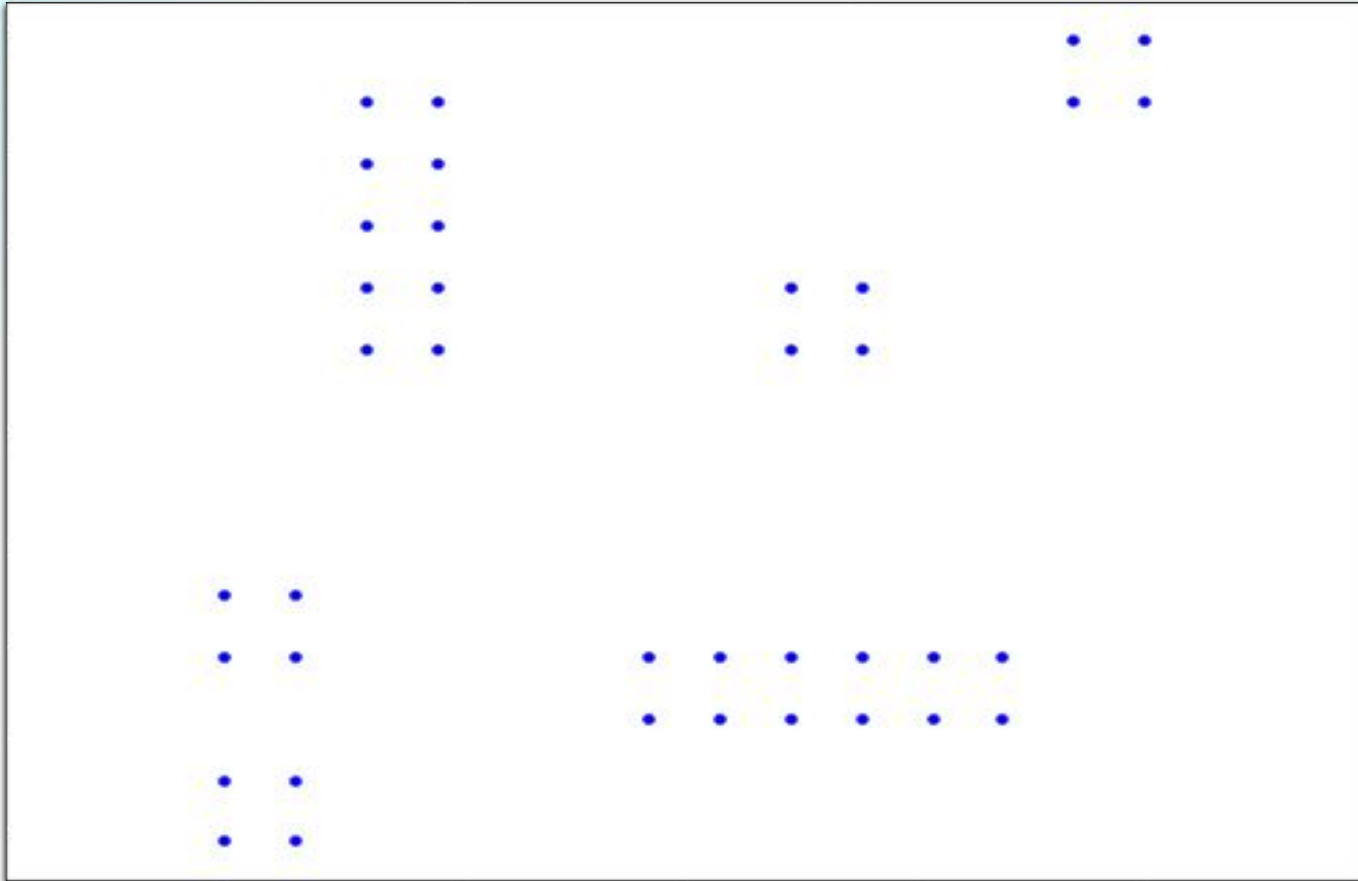
Dot Patterns Perceptual Grouping: RDG vs K-means

Di cosa parleremo?

- Percezione visiva di una scena
- Gestalt e percezione di Dot Patterns
- Cos'è il clustering?
- Algoritmo di clustering K-Means
- Algoritmo basato sul Reduced Delaunay Graph
- Confronto tra i due algoritmi

Come si può modellare la percezione dei Dot Patterns?

Quanti gruppi percepiamo?



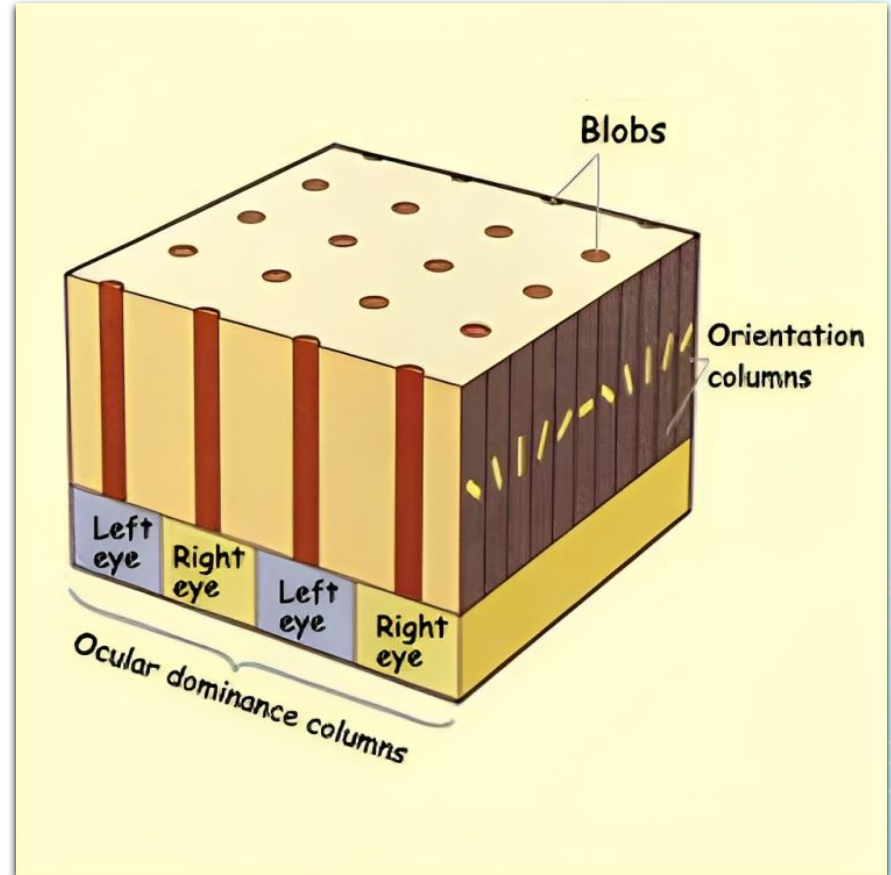
Location e Orientation Columns

Neuroni in una location column hanno **campi ricettivi molto vicini**.

All'interno di una location column sono presenti orientation columns con certe **orientazioni preferite**.

Le orientation columns sono raggruppate in **pinwheel** alternati a **blob**.

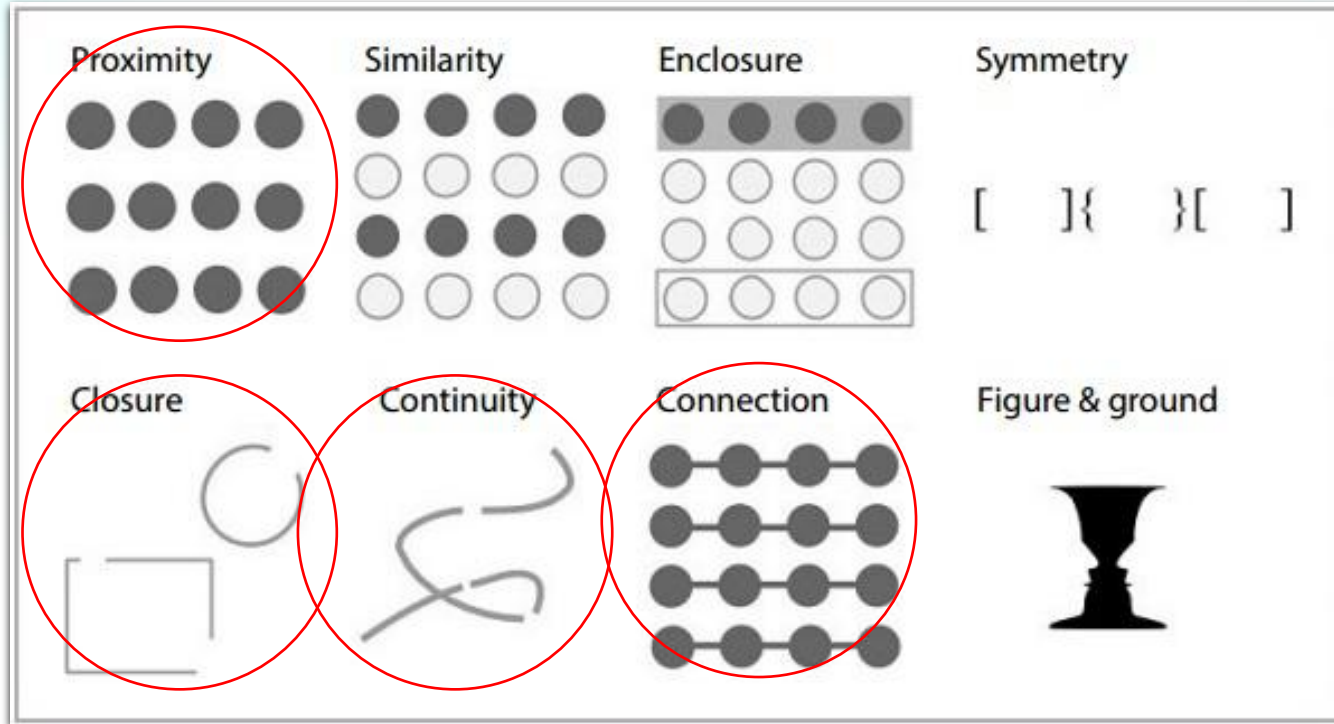
Le **ocular dominance columns** ricevono input da un occhio o dall'altro mediante il **Lateral Geniculate Nucleus**.



Leggi della Gestalt

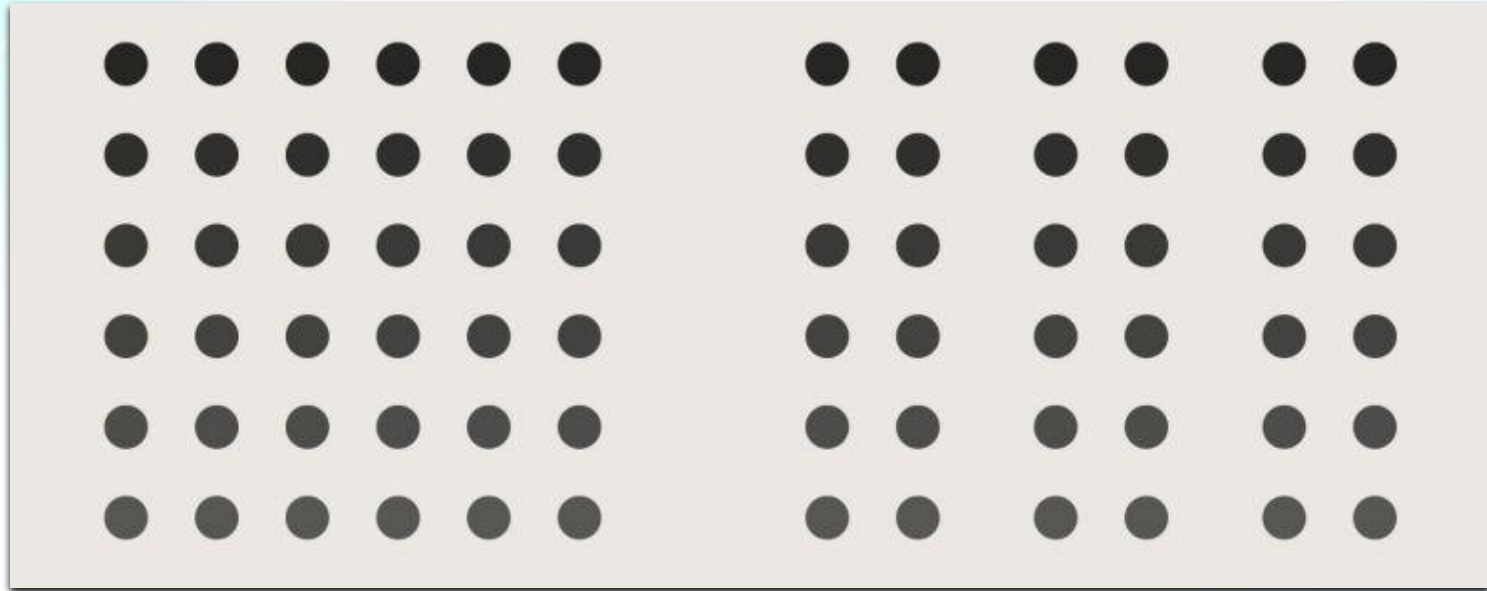
La psicologia della Gestalt concerne la **forma** e la **rappresentazione** nel contesto del Sistema Visivo Umano.

Elementi in un gruppo possono avere proprietà che emergono da **relazioni reciproche**.



Legge della Gestalt: prossimità

L'occhio umano tende a **raggruppare** gli elementi **vicini**, separandoli da quelli più lontani.



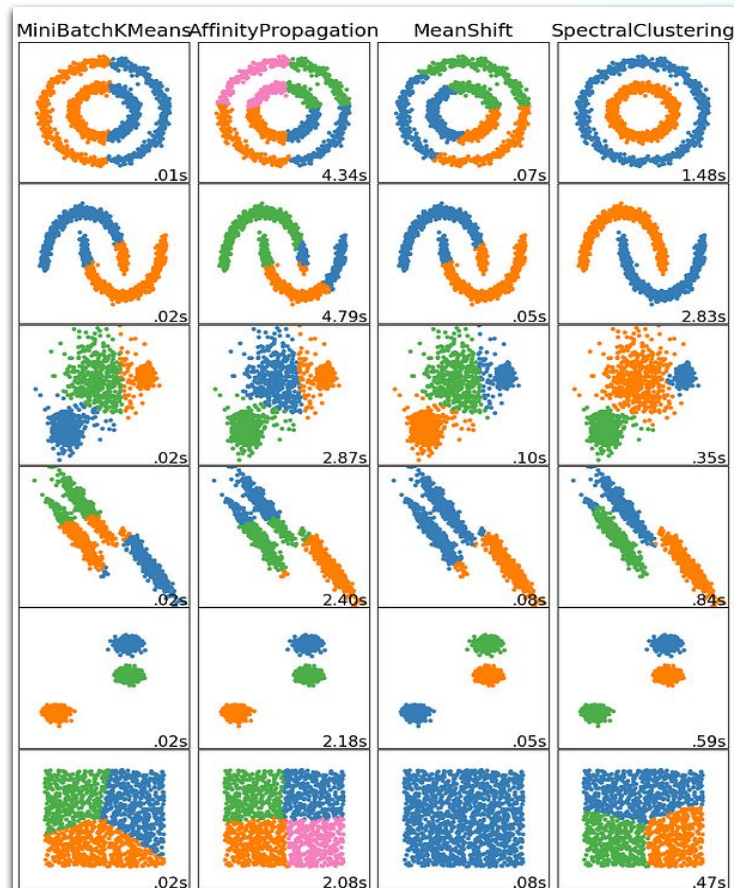
Vogliamo riprodurlo **algoritmicamente**!

Un modo per modellare la Gestalt: Clustering

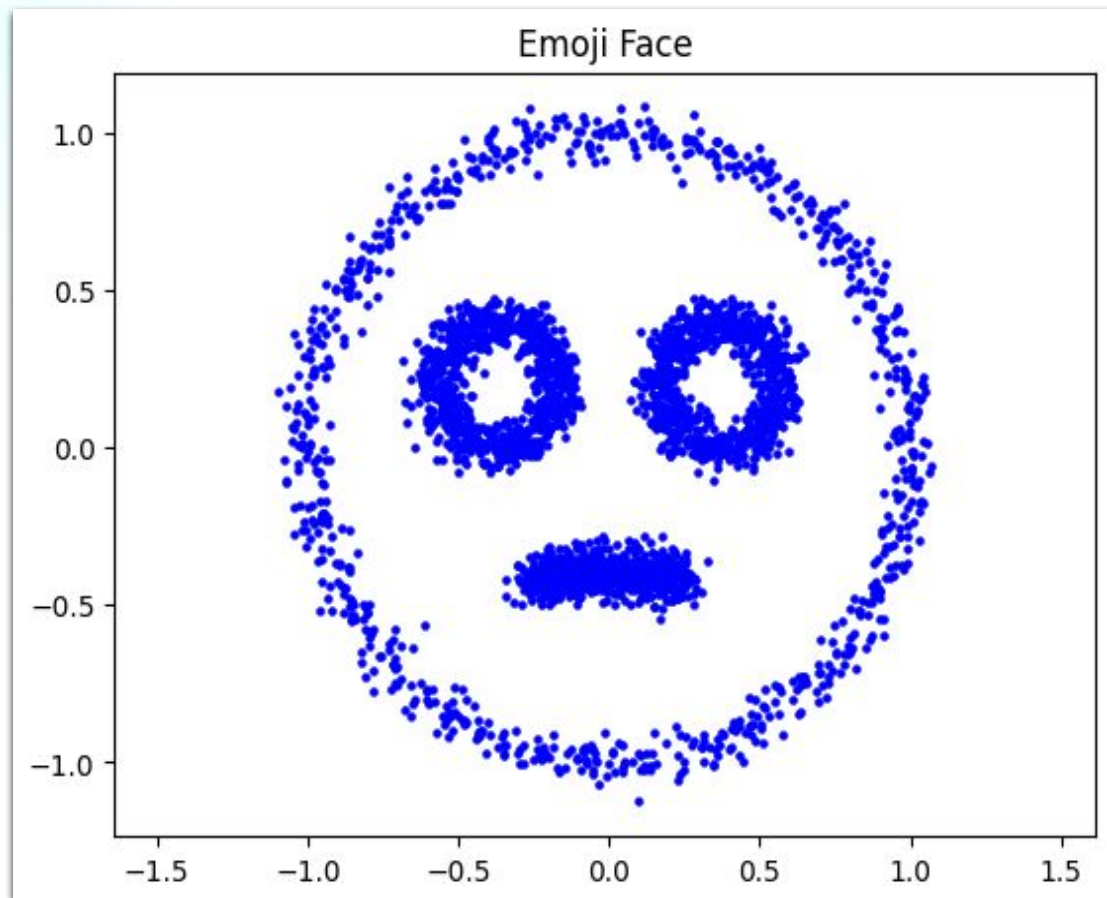
Il **clustering** è quell'insieme di tecniche non supervisionate volte alla selezione e **raggruppamento** di **elementi omogenei** in un insieme di dati.

Tipologie:

- **Clustering esclusivo** (o hard clustering): ogni elemento è assegnato ad uno e un solo cluster.
- **Clustering non-esclusivo** (o soft/fuzzy clustering): un elemento può appartenere a più cluster con gradi di appartenenza diversi.
- **Clustering partizionale** (o non gerarchico o k-clustering): l'appartenenza ad un gruppo viene data da una distanza da un punto rappresentativo. Esempio: **k-means**.
- **Clustering gerarchico**: presenta una gerarchia di partizioni caratterizzate da un numero (de)crescente di gruppi.



Dot Patterns



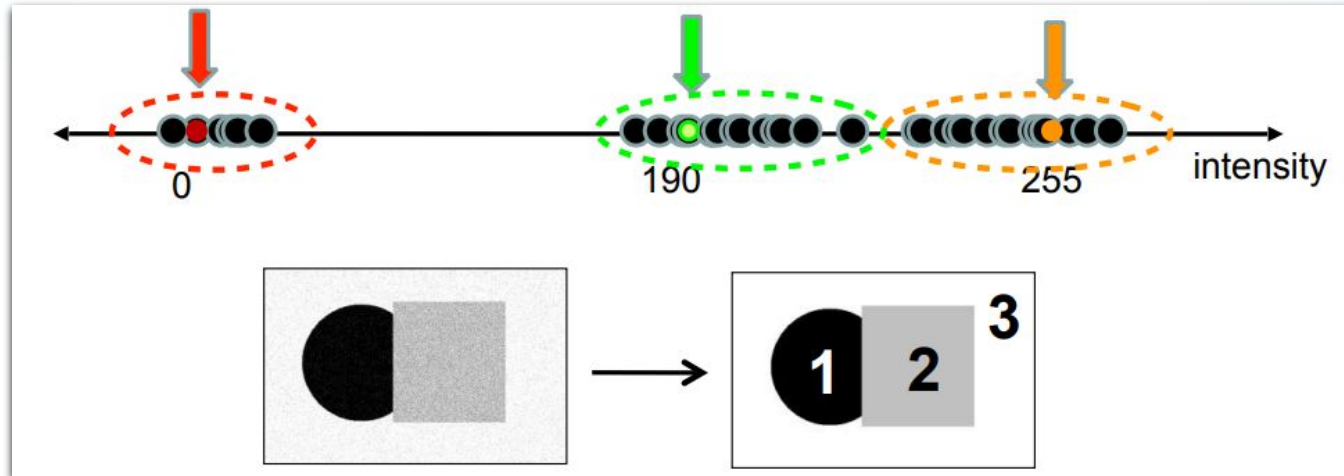
K-means clustering

Una prima idea è usare l'algoritmo **k-means**.

Identificazione dei gruppi → scelta di centroidi rappresentanti che minimizzino una certa

funzione di costo: $\sum_{j=1}^k E(C_j)$

Ad esempio la **Sum of Square Distances (SSD):** $\sum_{cluster\ i} \sum_{punti\ p\ nel\ cluster\ i} ||p - c_i||^2$



Problema dell'uovo e la gallina con i gruppi e i centroidi corrispondenti!

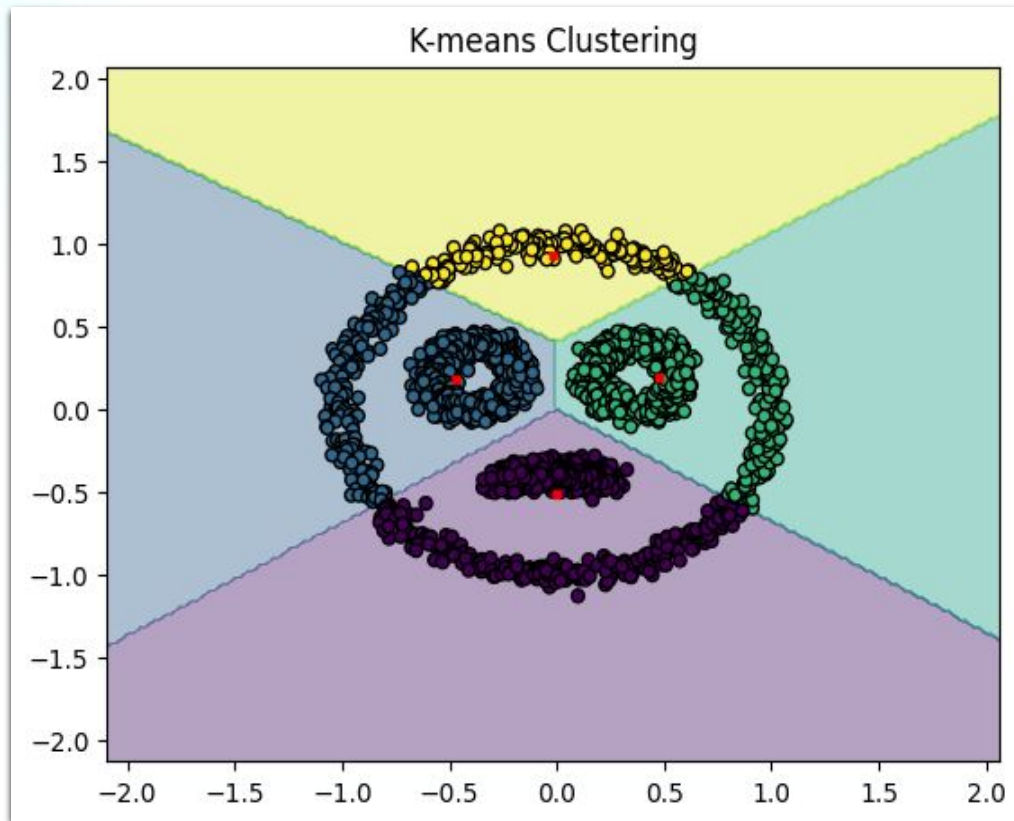
K-means clustering

Input: un array di punti nel piano caratterizzati da una coppia di coordinate $[x,y]$ e il numero k di cluster.

Output ed effetti (tipicamente): le coordinate dei centroidi per ogni cluster e assegnazione di ogni punto in input ad un cluster.

1. Inizializzare **casualmente** k centroidi c_1, \dots, c_k
2. Dati i centroidi trovare i punti per ogni cluster:
 - a) Per ogni punto p trovare il c_i più vicino
 - b) Mettere p nel cluster i
3. Dati i punti trovati in ogni cluster, trovare c_i e settare c_i come media dei punti del cluster i
4. Se c_i è cambiato rispetto al corrente tornare 2. altrimenti terminare

K-means clustering



Vogliamo modellare meglio la **percezione**!

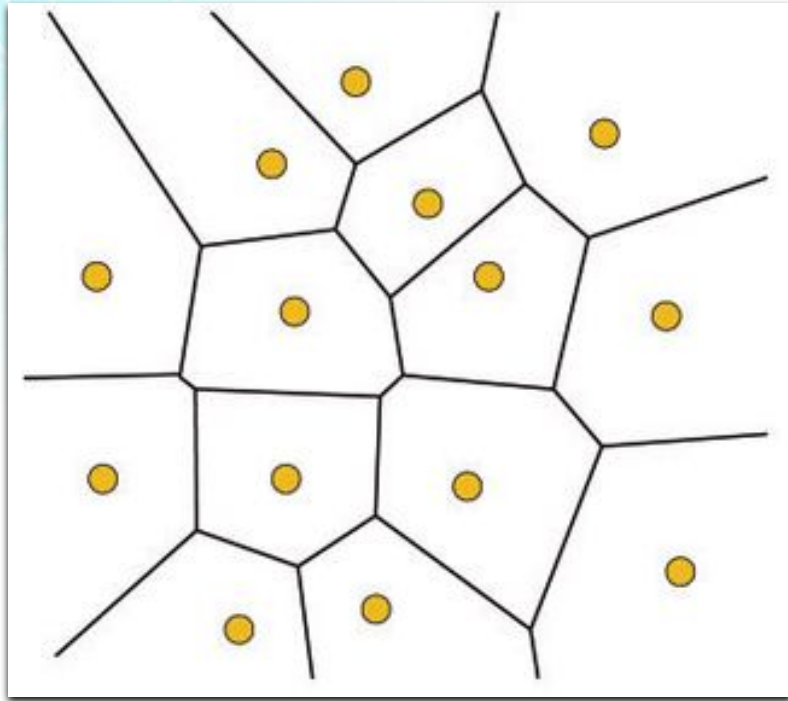
Diagramma di Voronoi e Triangolazione di Delaunay

Dato un insieme $S = \{p_1, \dots, p_N\}$ di **N punti** nel piano, **partizioniamo il piano in celle** C_1, \dots, C_N in modo tale che i punti appartenenti alla cella C_j , associata al punto $p_j \in S$, siano più vicini a p_j che a qualsiasi altro punto $p_k \in S, k \neq j$:

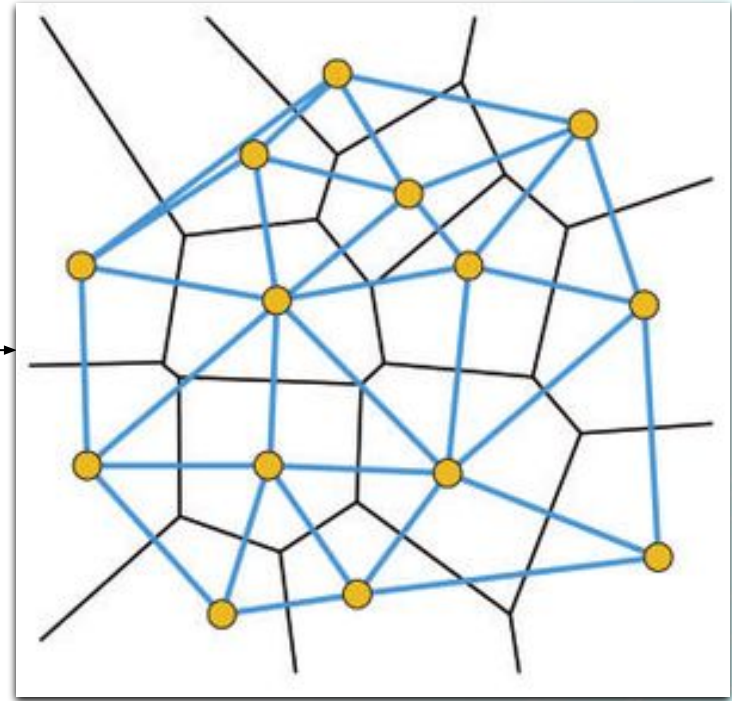
$$q \in C_j \Leftrightarrow d(q, p_j) \leq d(q, p_k) \quad \forall q, \forall p_j, p_k \in S$$

Il duale del Diagramma o Tassellazione di Voronoi, chiamato Grafo o Triangolazione di Delaunay, si ottiene **connettendo tutte le coppie di punti** di S le cui celle nel diagramma di Voronoi **condividono un confine**.

Diagramma di Voronoi

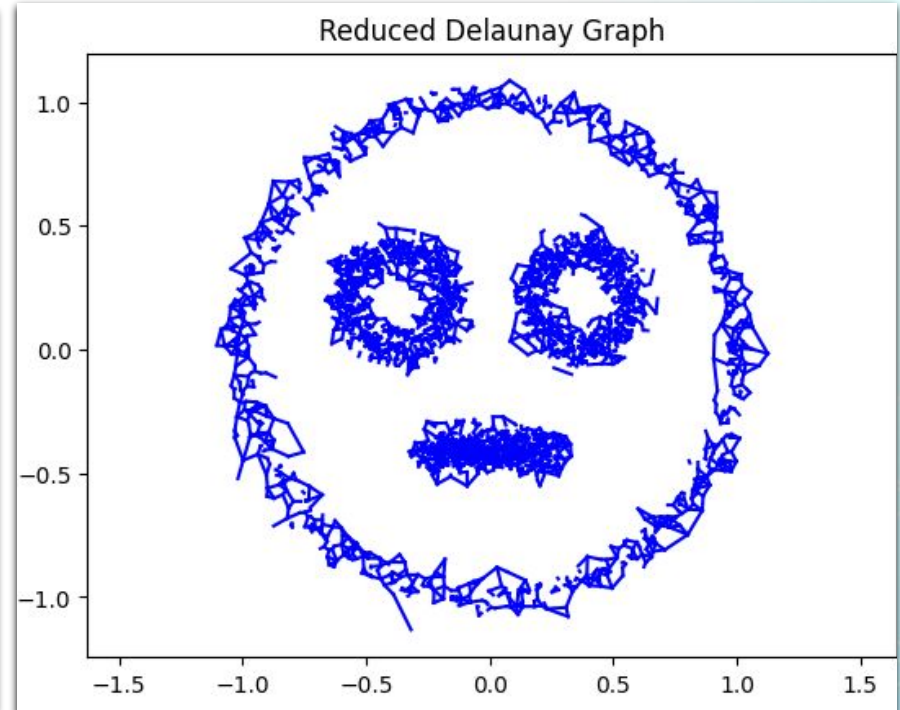
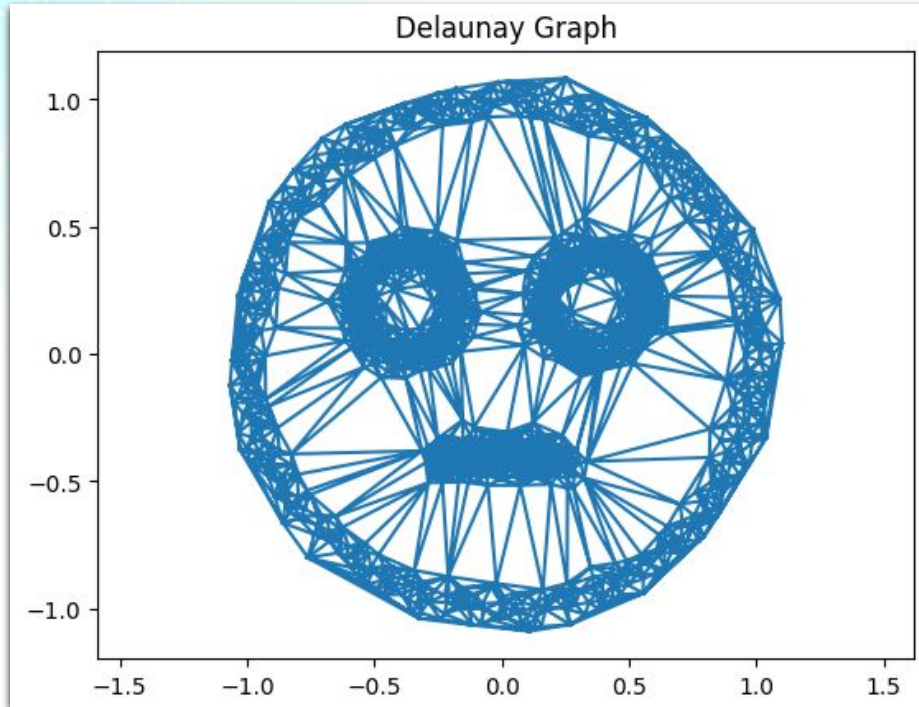


Triangolazione di Delaunay



Reduced Delaunay Graph

Idea: tenere **solo alcuni edge** in modo da evidenziare le zone prossimali.



Reduced Delaunay Graph

Per scegliere quali edge pq da rimuovere, calcoliamo una **distanza normalizzata**:

$$\xi(p, q) = \frac{d(p, q)}{\min_{x \in S} \{d(p, x)\}}; \quad \xi(q, p) = \frac{d(q, p)}{\min_{x \in S} \{d(q, x)\}}$$

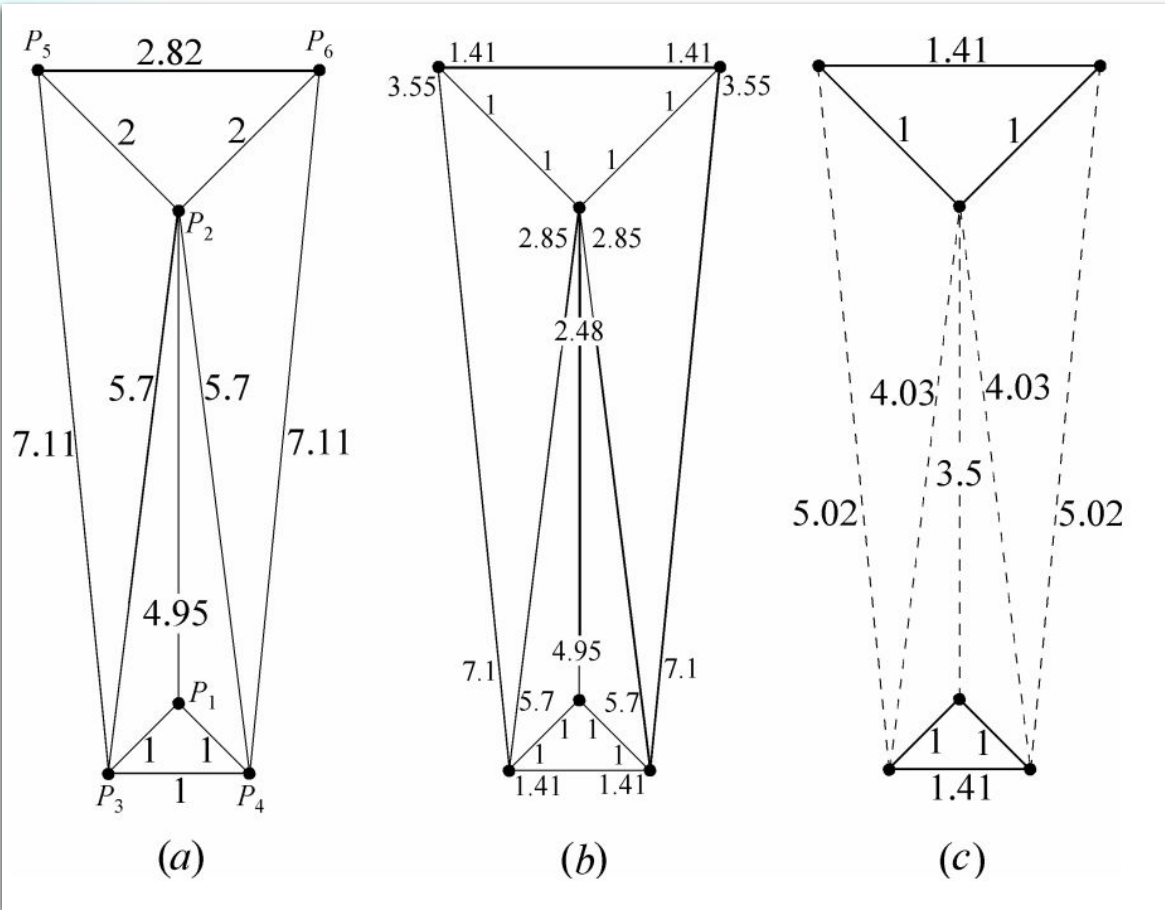
Assegniamo così due rapporti $r_1(e) = \xi(p, q)$, $r_2(e) = \xi(q, p)$ ad ogni edge del grafo. Dunque, li riduciamo ad una sola quantità, la loro **media geometrica**:

$$r(e) = \sqrt{r_1(e) \cdot r_2(e)}$$

E rimuoviamo dal grafo ogni edge per cui $r(e)$ è maggiore di un certo **threshold** r_T :

$$V_{RDG} = V_{DG}; \quad E_{RDG} = \{e \in E_{DG} \mid r(e) \leq r_T\}$$

Reduced Delaunay Graph



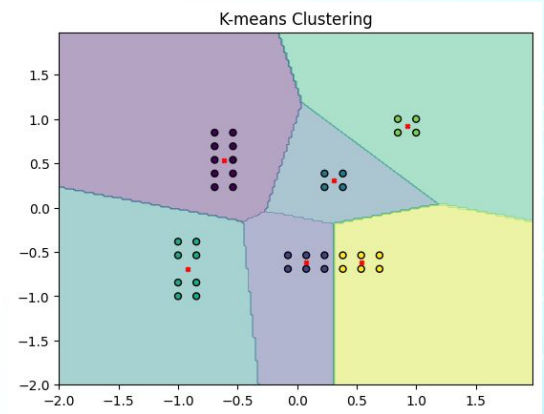
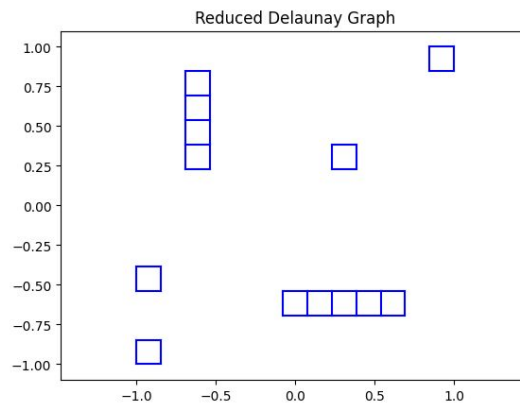
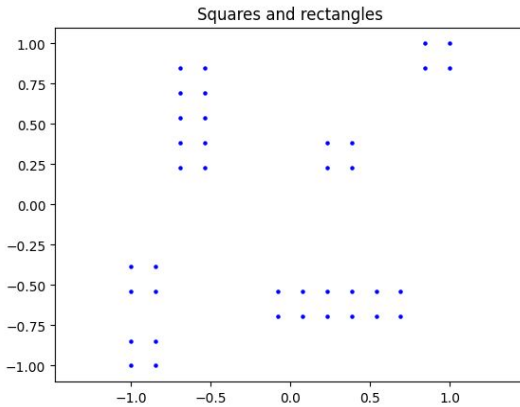
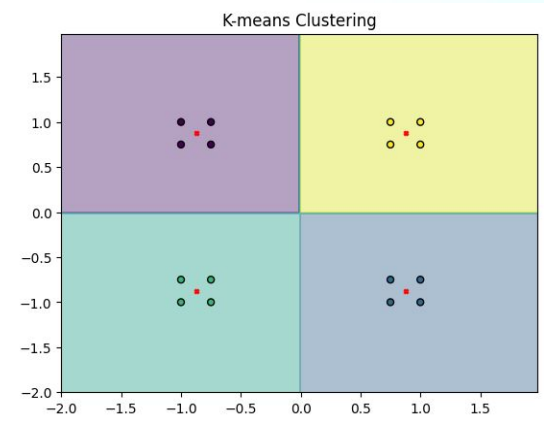
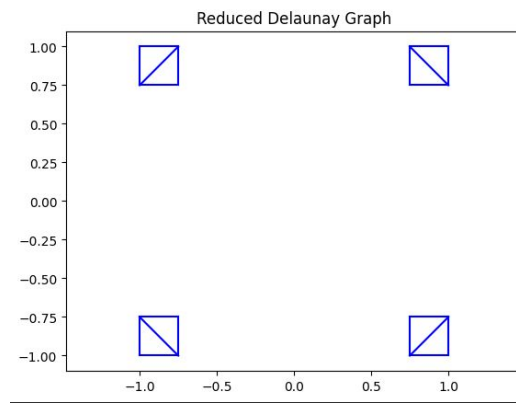
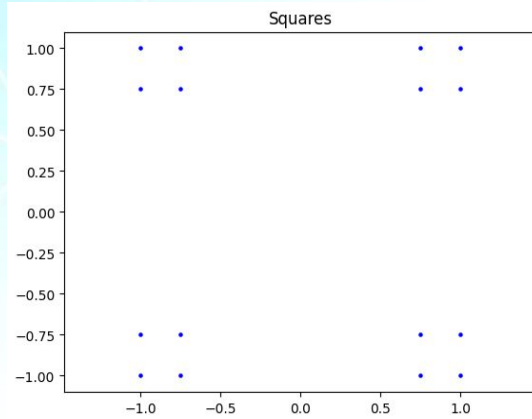
RDG: Algoritmo

Input: un array di punti nel piano caratterizzati da una coppia di coordinate $[x,y]$.

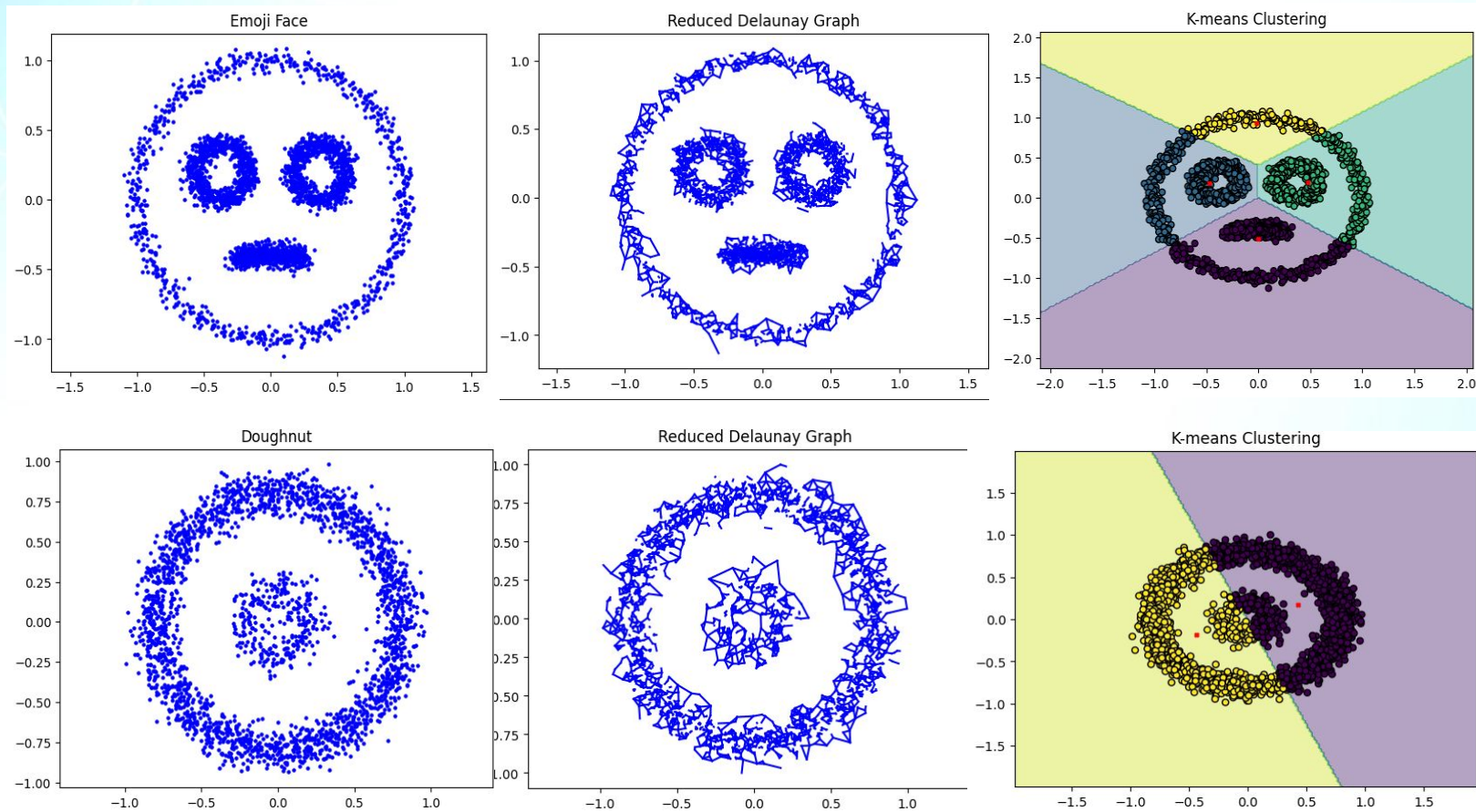
Output: un array di punti ridotto che, collegati con edge, formano l'RDG.

1. Calcolare la **triangolazione di Delaunay** sui punti dati in input
2. Calcolare le **distanze** tra i vicini per ogni punto
3. **Normalizzare** distanze con il minimo delle distanze tra i vicini
4. Calcolare la **media geometrica** tra i due rapporti trovati per ogni edge, e associare loro il risultato
5. **Thresholding:** eliminare gli edge il cui valore è maggiore di un certo threshold

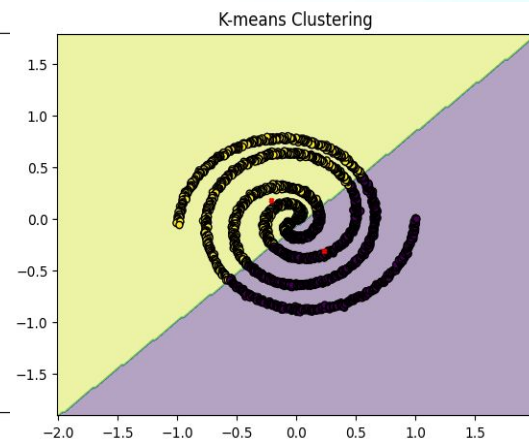
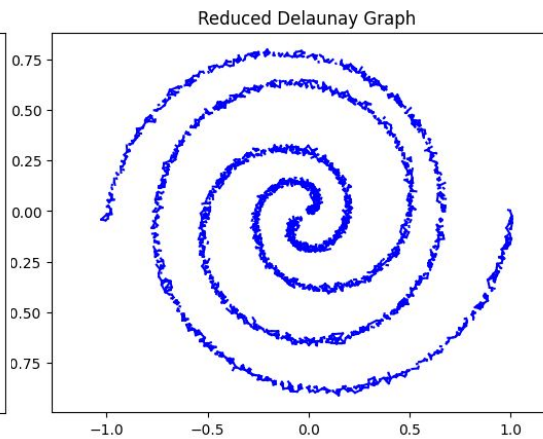
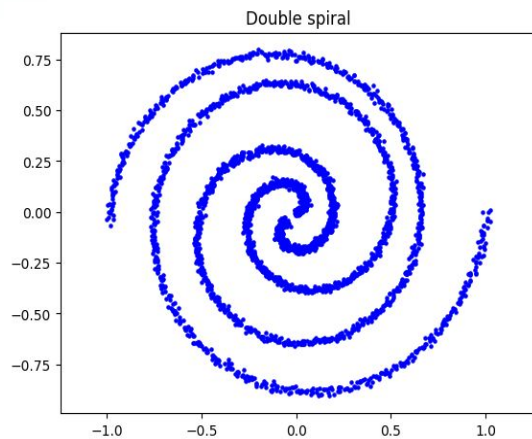
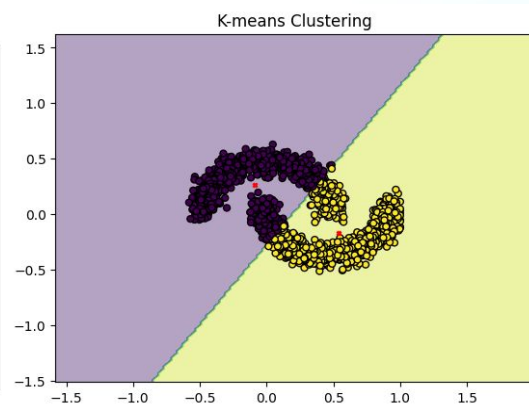
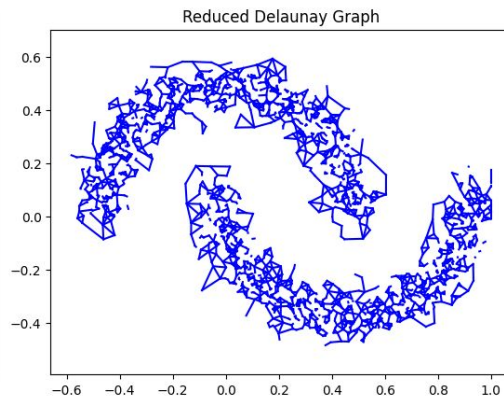
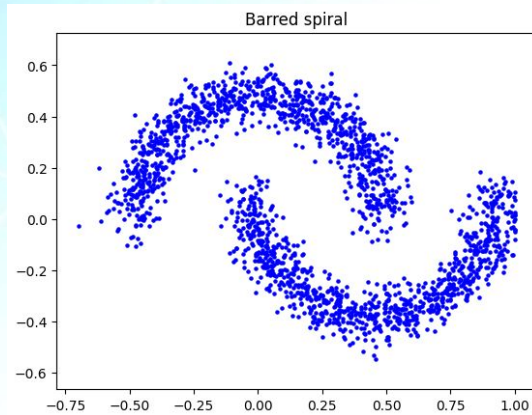
K-means vs RDG



K-means vs RDG



K-means vs RDG



K-means vs RDG

Somiglianze:

- Entrambi sono algoritmi **non supervisionati**
- La **tassellazione** del piano può far parte delle loro operazioni o output
- Entrambi funzionano bene con **nuvole di punti** separate

Differenze:

- RDG usa un **grafo** per considerare le distanze tra i punti
- RDG tecnicamente **non sceglie rappresentanti** per i gruppi trovati
- K-means può avere elementi di **casualità**

K-means vs RDG

Pro del k-means:

- **Semplice** da implementare
- Computazionalmente (abbastanza) **efficiente**
- **Converge sempre** ad un **minimo locale** per ogni cluster

Contro del k-means:

- Impostare k è difficile
- Sensibile ai centri iniziali **casuali** e al **rumore**
- Adatto principalmente a **cluster sferici**
- Non modella bene la percezione umana

Pro di RDG:

- Modella bene la percezione umana
- Può trovare gruppi anche in **forme complesse**
- Robusto ad **outlier lontani**

Contro di RDG:

- Impostare il threshold è difficile (specialmente a scale diverse)
- Computazionalmente **pesante**

K-means vs RDG: complessità

K-means: $O(nkt)$, con n numero di punti, t numero di iterazioni per convergere ad un minimo locale, k numero di cluster scelto.

RDG: $O(n\log(n))$, con n numero di punti.

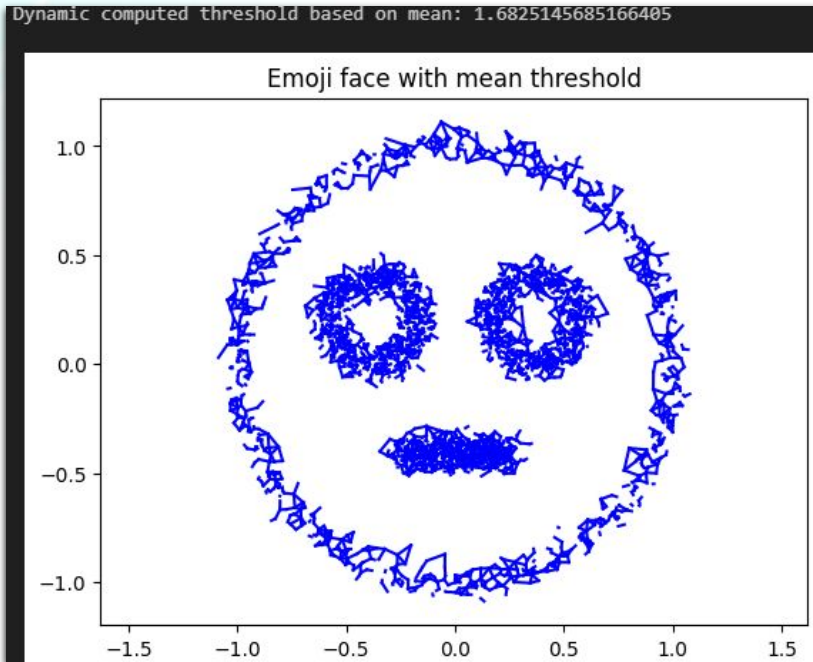
Complessità dedotta empiricamente.



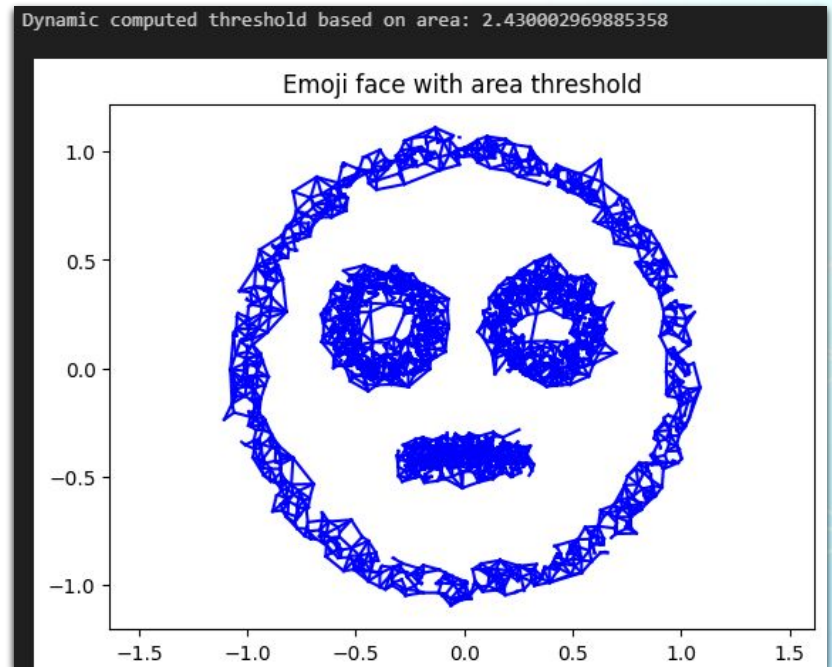
Come si potrebbe migliorare?

Una cosa a cui abbiamo pensato è l'impostazione di un **threshold dinamico**.

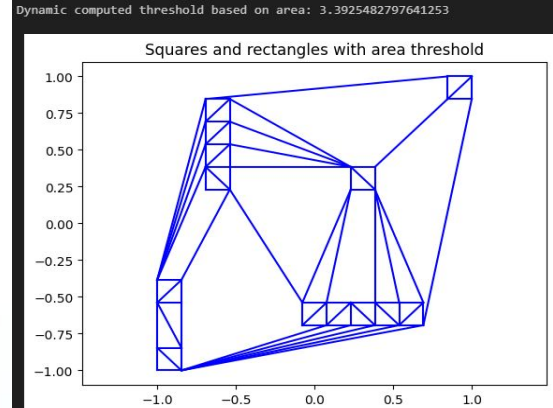
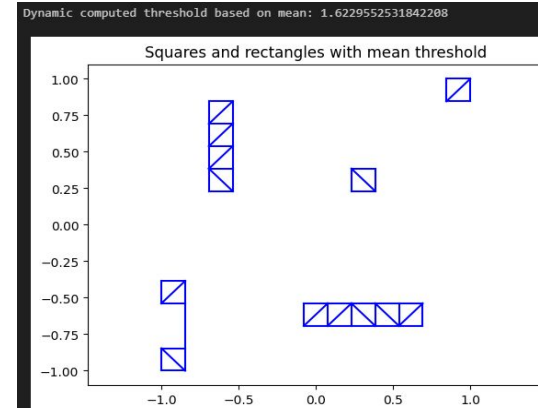
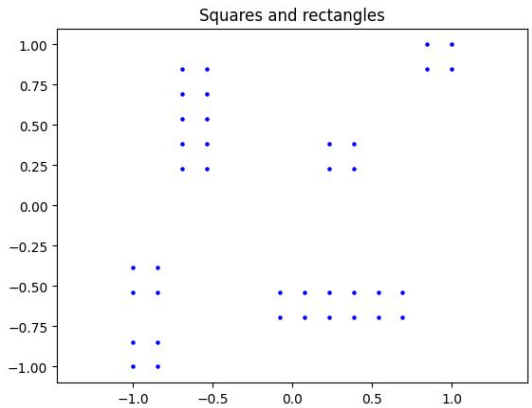
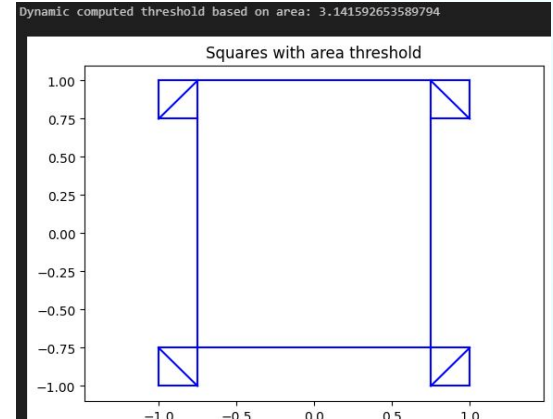
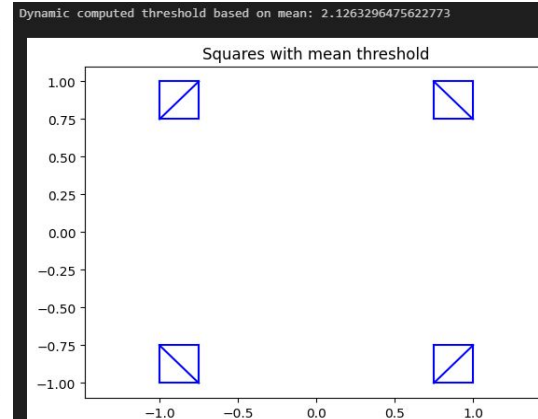
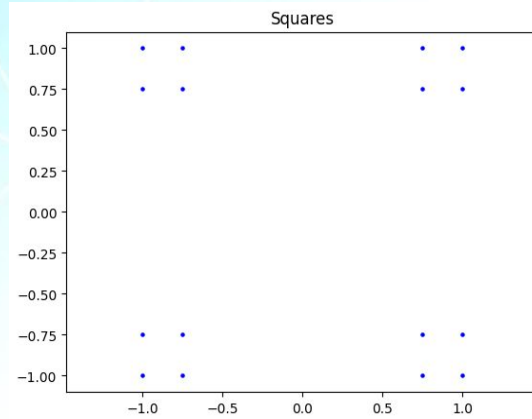
In base alla **media dei valori** associati ad ogni edge.



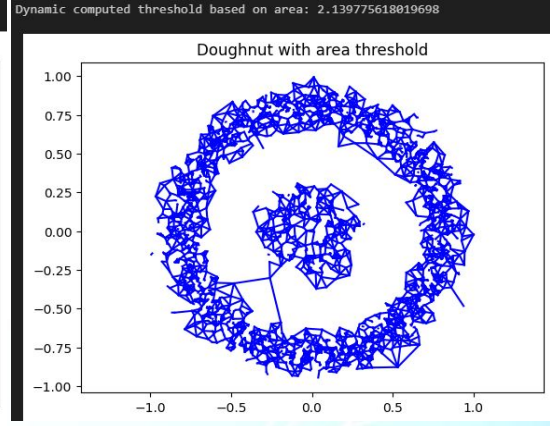
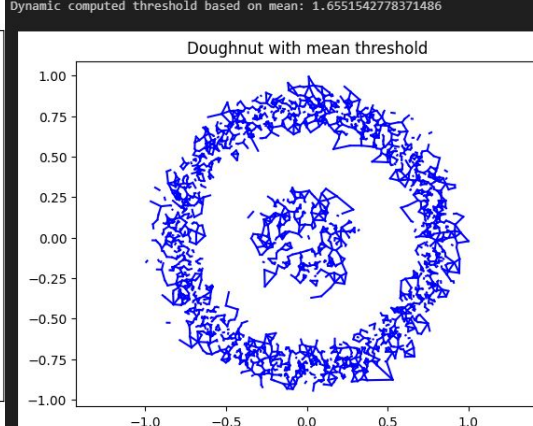
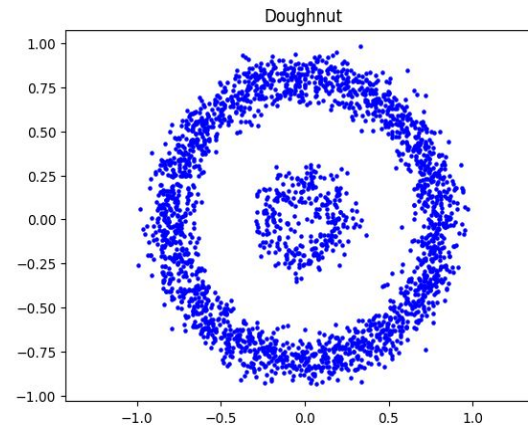
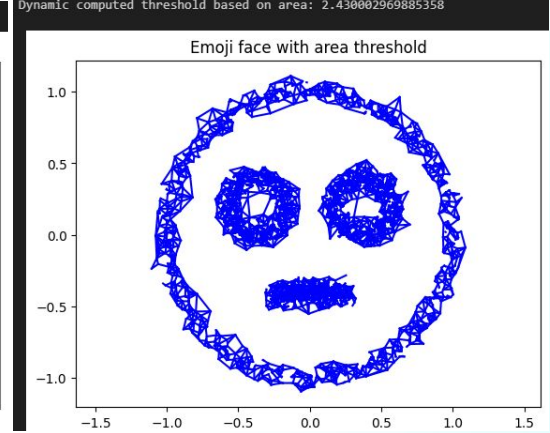
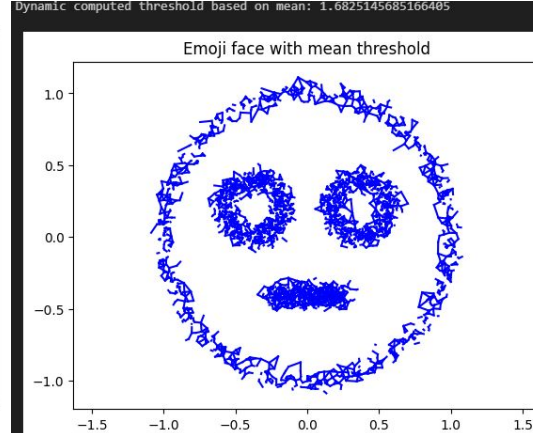
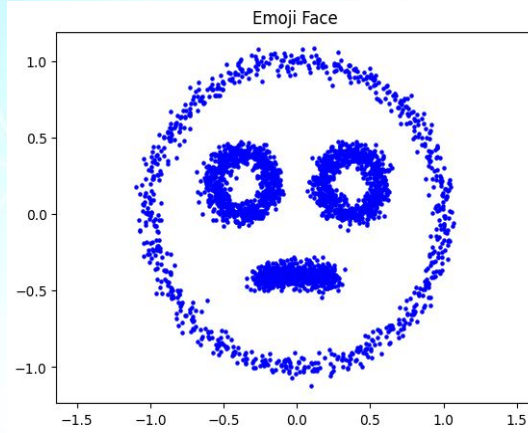
In base all'area del **minimum bounding circle**.



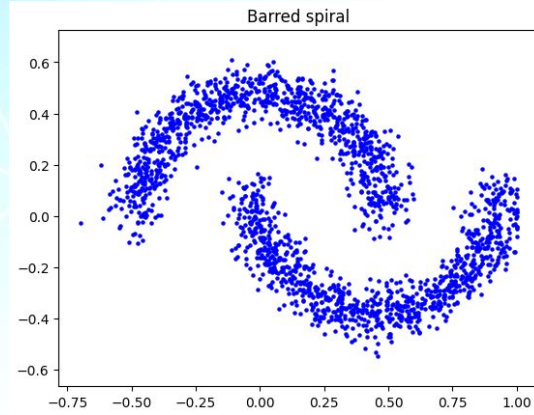
Mean vs Area threshold



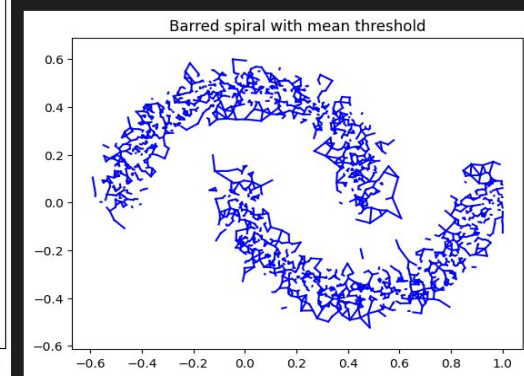
Mean vs Area threshold



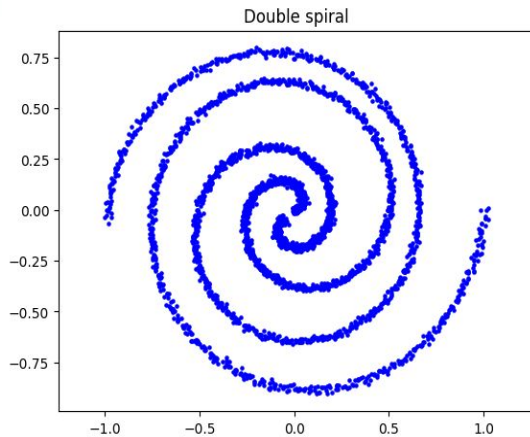
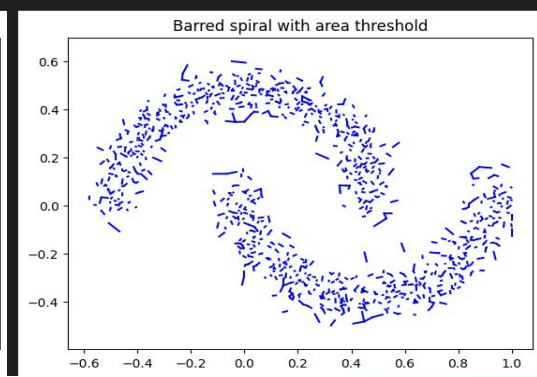
Mean vs Area threshold



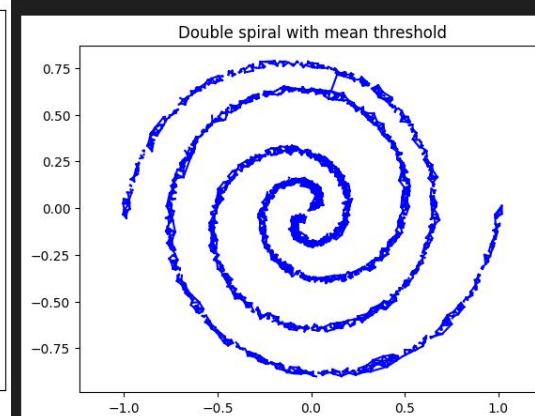
Dynamic computed threshold based on mean: 1.599048482130292



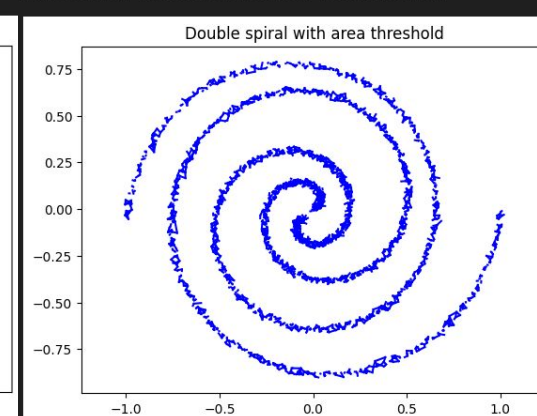
Dynamic computed threshold based on area: 1.1128946424794033



Dynamic computed threshold based on mean: 2.307183789444543



Dynamic computed threshold based on area: 1.7774135526858257



Fonti esterne

- <https://pressbooks.umn.edu/sensationandperception/chapter/columns-and-hypercolumns-in-v1>
- <https://www.analytixlabs.co.in/blog/types-of-clustering-algorithms>
- https://www.cs.rug.nl/~petkov/publications/2005LNCS3704_grouping_dots.pdf