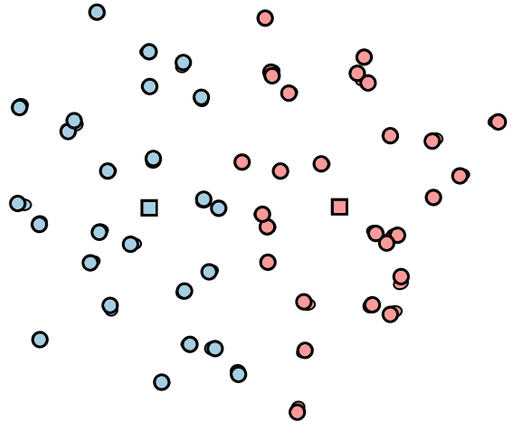
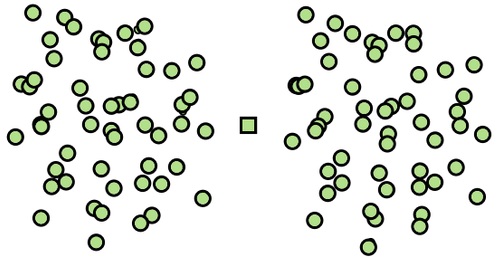
UNIVERSITÄT TRIER

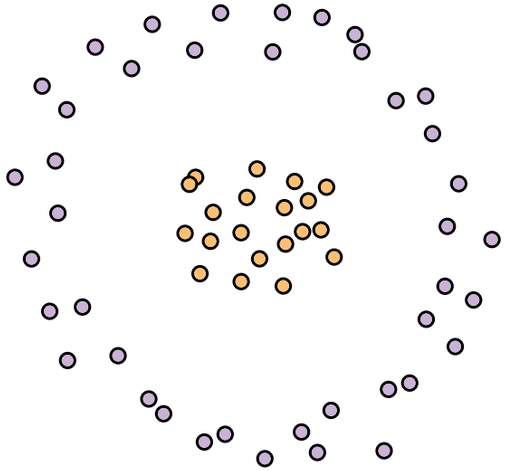
Algorithmen für geographische Informationssysteme

13. Vorlesung
Clustering



Teil I:
Problemstellung





Philipp Kindermann

Wintersemester 2024/25

Clustering

Clustering ist klassischerweise das Problem, eine Partition eines Datensatzes zu finden, sodass Elemente in derselben Zelle („Cluster“) gemäß einem gegebenen Distanzkriterium nahe beieinander liegen, während Elemente in verschiedenen Mengen weit voneinander entfernt sind.

Fundamentales Problem im Data Mining, aber nicht eindeutig definiert.

- Was willst du clustern?
- Was willst du mit den Daten tun?
- Distanzen:
  - Euklidisch?
  - metrisch?
- Wie viele Cluster?
- Wie kann ein Cluster aussehen?

Zielfunktionen

1. „Minimale Anforderungen an Domänenwissen zur Bestimmung der Eingabeparameter, da geeignete Werte oft nicht im Voraus bekannt sind, wenn mit großen Datenbanken gearbeitet wird.“

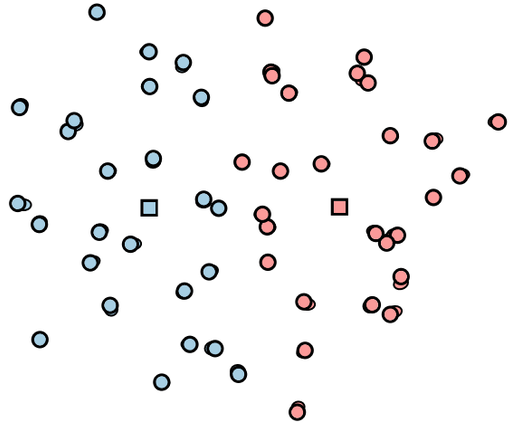
2. „Entdeckung von Clustern mit beliebiger Form, da die Form von Clustern in räumlichen Datenbanken kugelförmig, langgezogen, linear, verlängert etc. sein kann.“

3. „Gute Effizienz bei großen Datenbanken, d.h. bei Datenbanken mit deutlich mehr als nur ein paar tausend Objekten.“

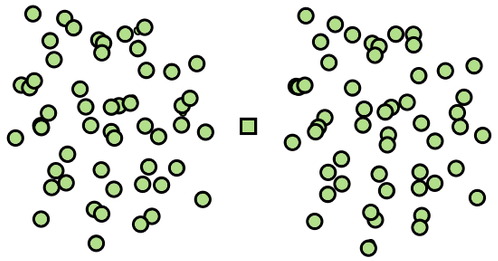
UNIVERSITÄT TRIER

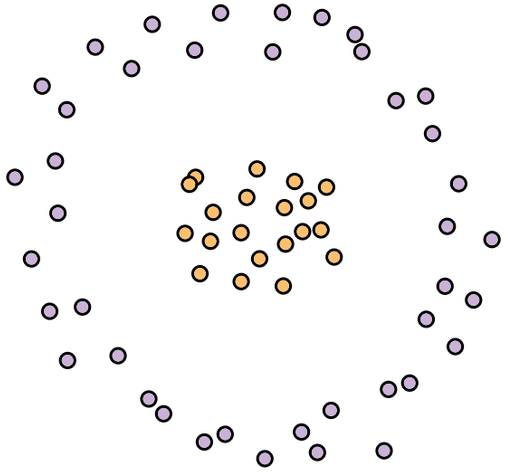
Algorithmen für geographische Informationssysteme

13. Vorlesung
Clustering



Teil II:
k-Means





Philipp Kindermann

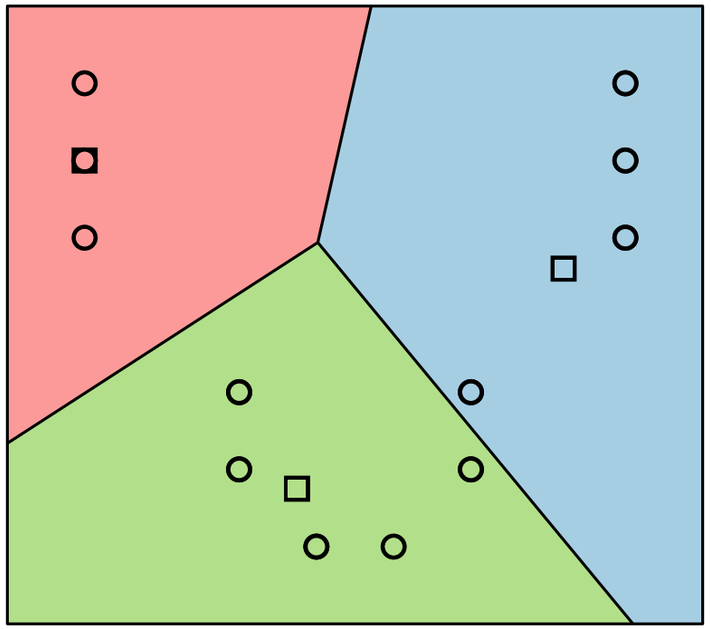
Wintersemester 2024/25

k-Means

Iterative Heuristik.

1. Wähle k zufällige Punkte als Clusterzentren
2. Weiße Punkte nähesten Clusterzentren zu
3. Verschiebe Clusterzentren auf Schwerpunkt des Clusters

Wiederhole 3. und 4. bis sich nichts mehr verändert

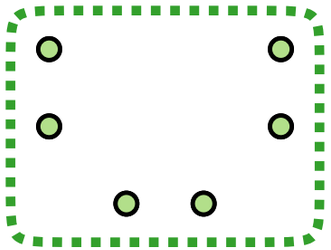


k-Means

Iterative Heuristik.

1. Wähle k zufällige Punkte als Clusterzentren
2. Weiße Punkte nähesten Clusterzentren zu
3. Verschiebe Clusterzentren auf Schwerpunkt des Clusters

Wiederhole 3. und 4. bis sich nichts mehr verändert



Laufzeit.

1. $\mathcal{O}(k)$
2. $\mathcal{O}(kn)$ pro Iteration
3. $\mathcal{O}(n)$ pro Iteration

k-Means: Beispiele

Segmentierung

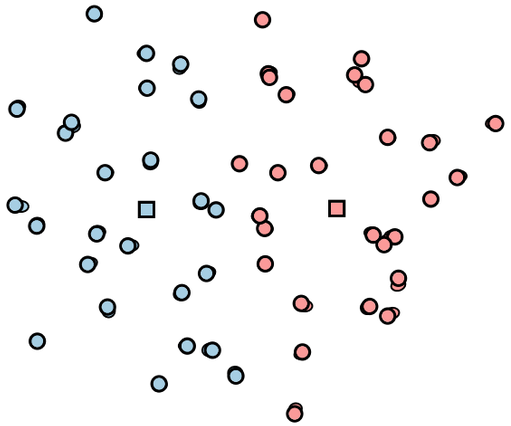
$k = 2$ | $k = 3$ | $k = 10$ | Original

Vektorquantisierung

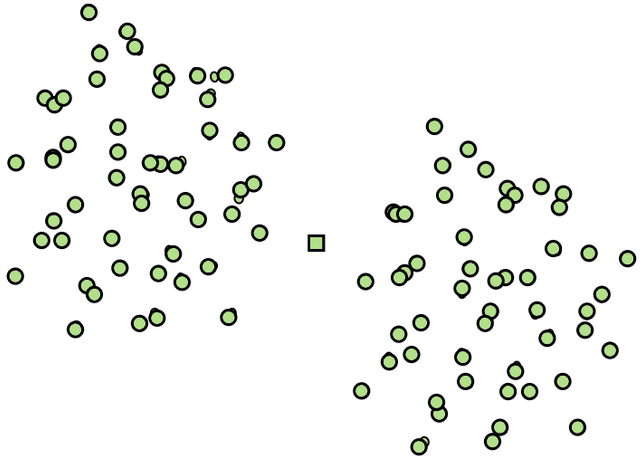
0,5 Bits / Pixel | 1,9 Bits / Pixel

8 Bits / Pixel

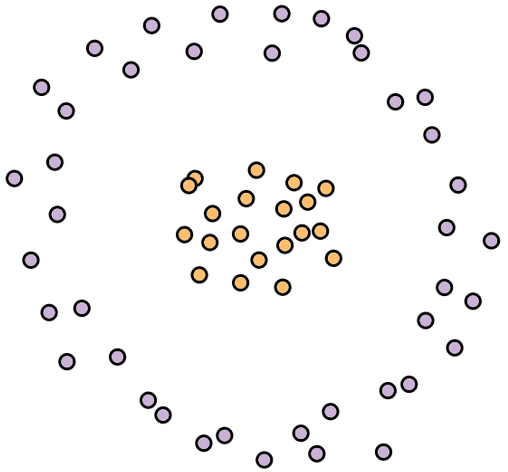
k-Means: Probleme



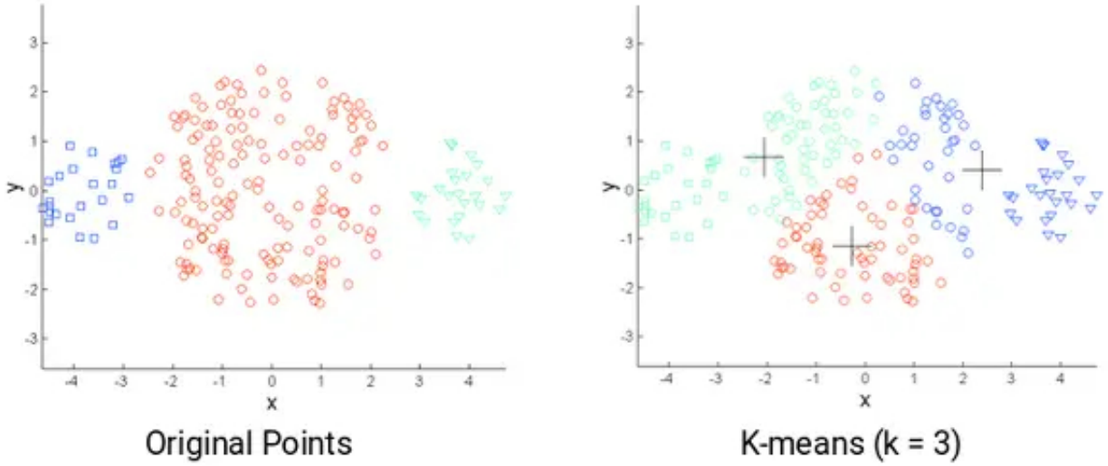
1 Cluster wäre besser



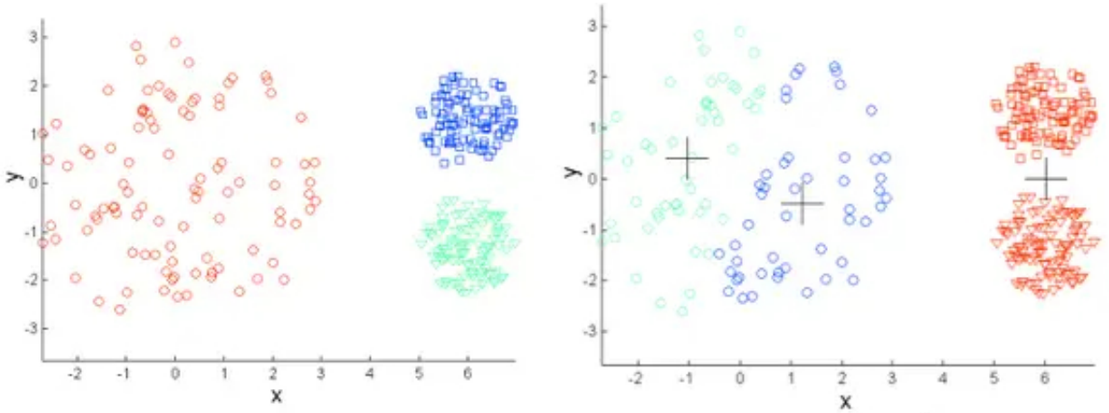
2 Cluster wären besser



kann *k*-Means nicht lösen



Unterschiedliche Clustergrößen

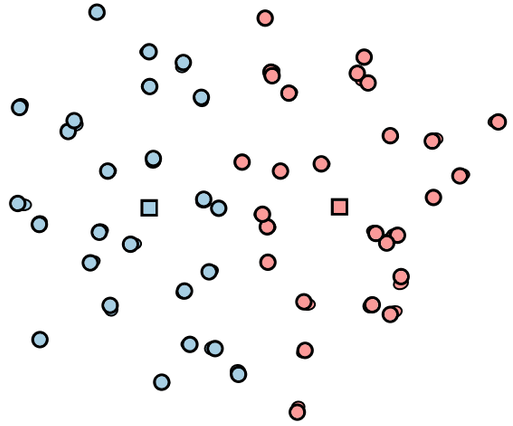


Unterschiedliche Punktdichte

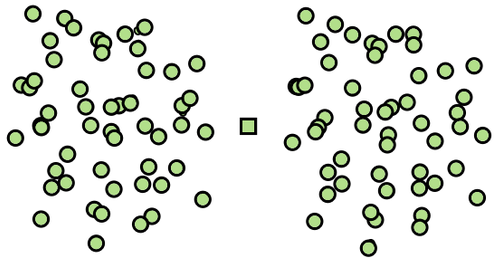
UNIVERSITÄT TRIER

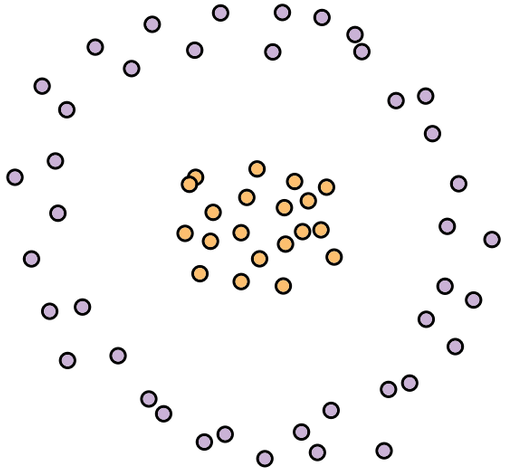
Algorithmen für geographische Informationssysteme

13. Vorlesung
Clustering



Teil III:
DBSCAN





Philipp Kindermann

Wintersemester 2024/25

DBSCAN

Gegeben. Menge X von (Daten)punkten, Distanzfunktion $d(\cdot,\cdot)$, max. Distanz ε und min. Anzahl k

Die **ε-Nachbarschaft** eines Punkts $p \in X$ ist $N_\varepsilon(p) = \{\, q \in X \mid d(p,q) \leq \varepsilon \,\}$.

Ein Punkt $p \in P$ heißt **Kern(punkt)** wenn $|N_\varepsilon(p)| \geq k$.

Ein Punkt $p \in P$ ist **direkt dichte-erreichbar** von einem Punkt q wenn:

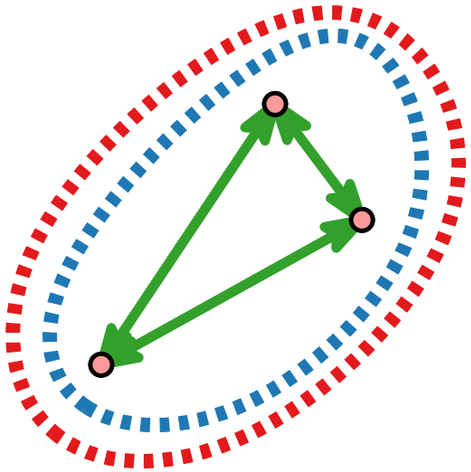
- $p \in N_\varepsilon(q)$
- $|N_\varepsilon(q)| \geq k$ (q ist ein Kern)

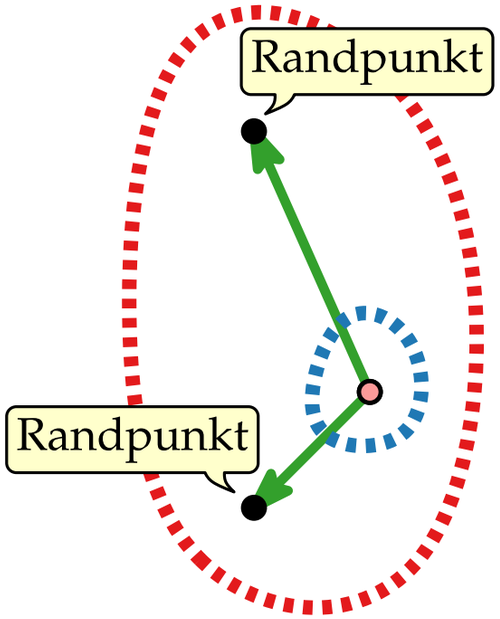
nicht symmetrisch!

Ein Punkt $p \in P$ ist **dichte-erreichbar** von einem Punkt q
$\Leftrightarrow$ es gibt einen Pfad von direkt dichte-erreichbaren Punkten von q nach p.

Zwei Punkte $p, q \in P$ sind **dichte-verbunden**
$\Leftrightarrow$ es gibt einen Kern r, so dass sowohl p als auch q von r dichte-erreichbar sind.

DBSCAN: Beispiel





Laufzeit.

Naiver Algorithmus läuft in $\mathcal{O}(n^2)$ Zeit.

> "Since the Eps-neighborhoods are expected to be small compared to the size of the whole data space, the average run time complexity of a single region query is $\mathcal{O}(\log n)$. [...] Thus, the average run time complexity of DBSCAN is $\mathcal{O}(n \log n)$."

Rauschen/Ausreißer

$k = 3$

Distanz ε

Kerne

direkt dichte-erreichbar

DBSCAN Clustering

DBSCAN* Clustering

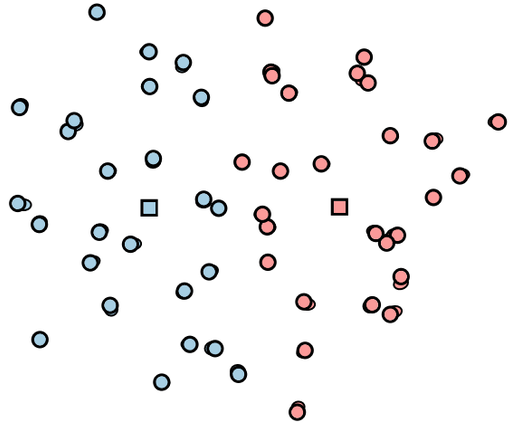
DBSCAN:
p und q sind im gleichen Cluster $\Leftrightarrow$ p und q sind dichte-verbunden

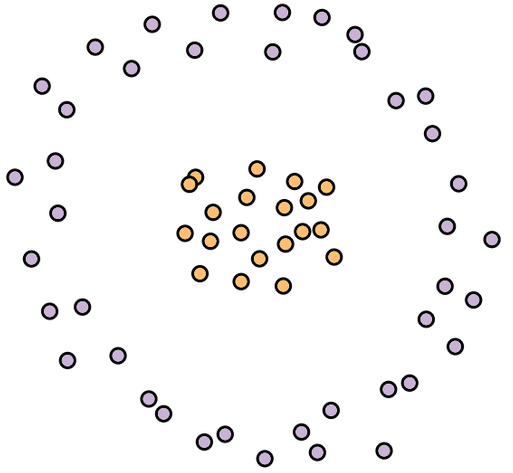
DBSCAN*:
(und Kerne)

UNIVERSITÄT TRIER

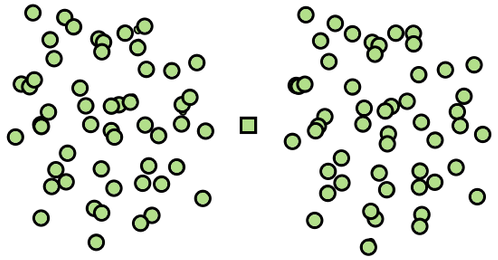
Algorithmen für geographische Informationssysteme

13. Vorlesung
Clustering





Teil IV:
DBSCAN: Laufzeit



Philipp Kindermann

Wintersemester 2024/25

De Berg, Gunawan, Roeloffzen (2017)

Annahme: ε frei, k feste Konstante, Euklidische Abstände

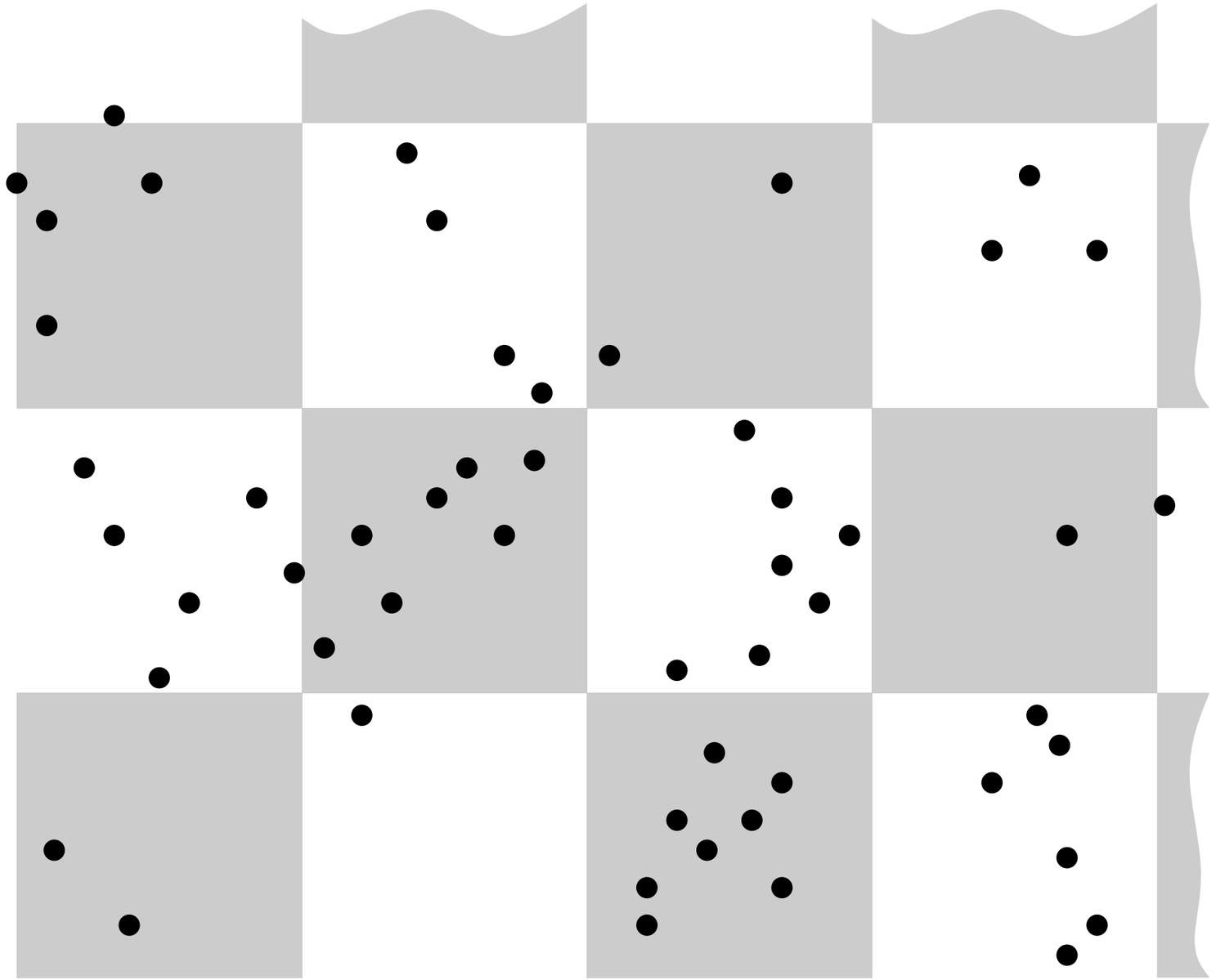
	2D	dD	
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil + 1}+\gamma})$	$\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$	

Box Graph $\mathcal{G}_{\text{box}}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



Gitterbasierter Ansatz.

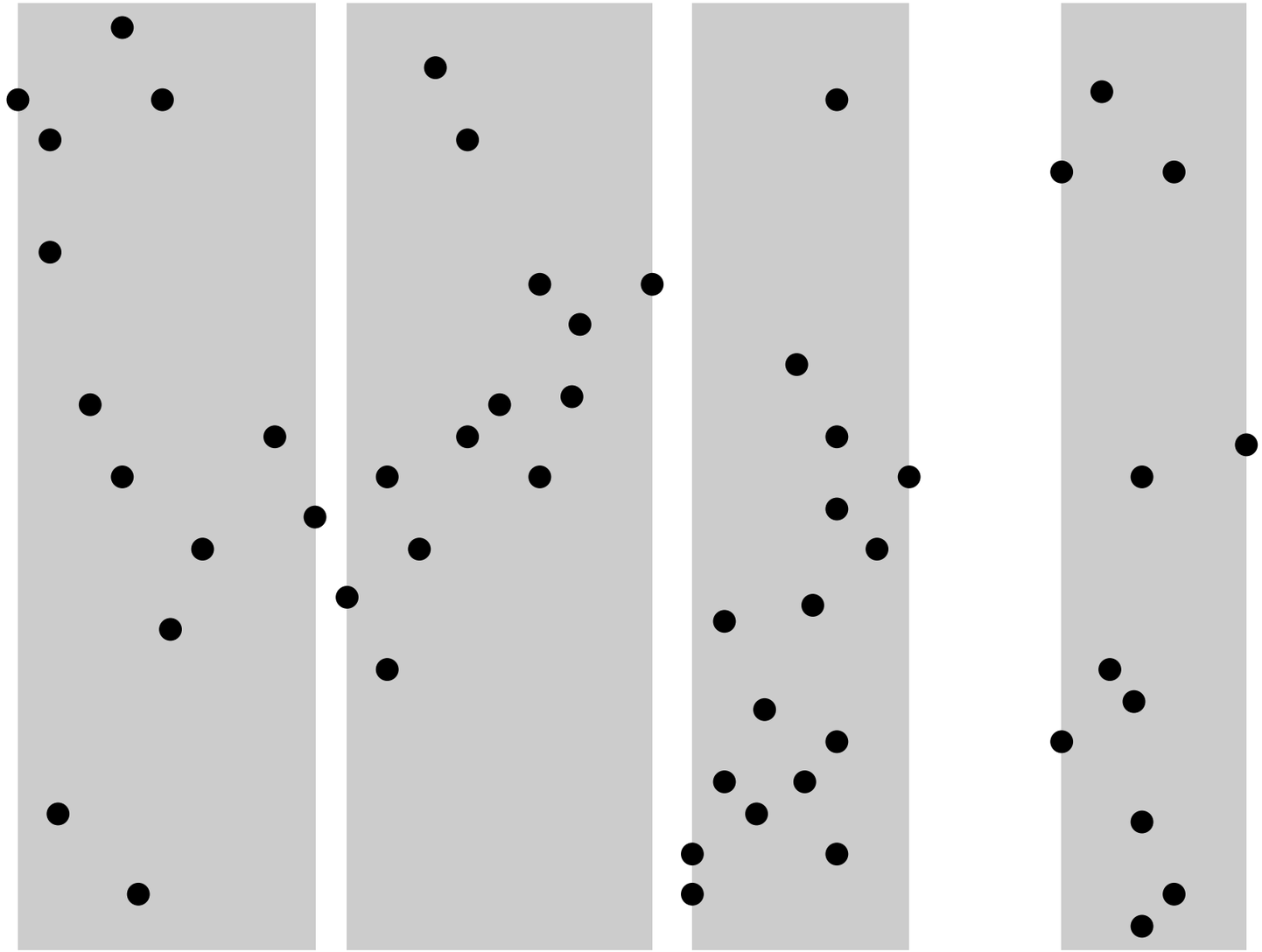
- Erstelle Gitter mit Zellengröße $\varepsilon/\sqrt{2}$.
- Zusammenhang in Gitterzellen?
- ... zwischen Gitterzellen?
- Nicht klar, wie wir die Laufzeit abhängig von n berechnen können, ohne die Verteilung der Punkte zu kennen.
- sei flexibler...

Box Graph $\mathcal{G}_{\text{box}}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



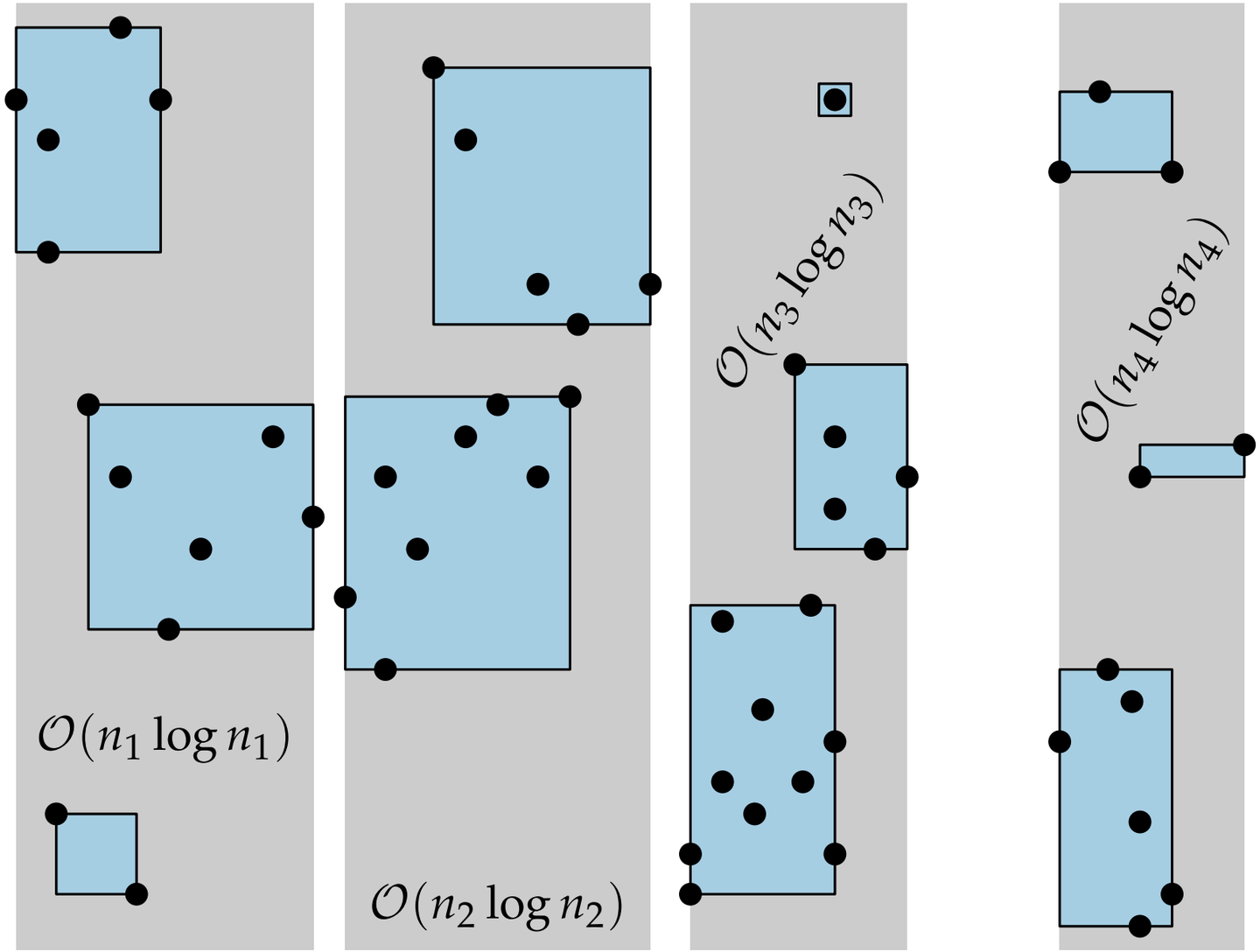
1. Erstelle Boxen.

- Füge Punkte hinzu, solange Streifenbreite $\leq \varepsilon/\sqrt{2}$.

SORTIERT

Box Graph $\mathcal{G}_{\text{box}}$

$k = 4$



1. Erstelle Boxen.

- Füge Punkte hinzu, solange Streifenbreite $\leq \varepsilon/\sqrt{2}$.
- Füge Punkte hinzu, solange Streifenhöhe $\leq \varepsilon/\sqrt{2}$.

Laufzeit.

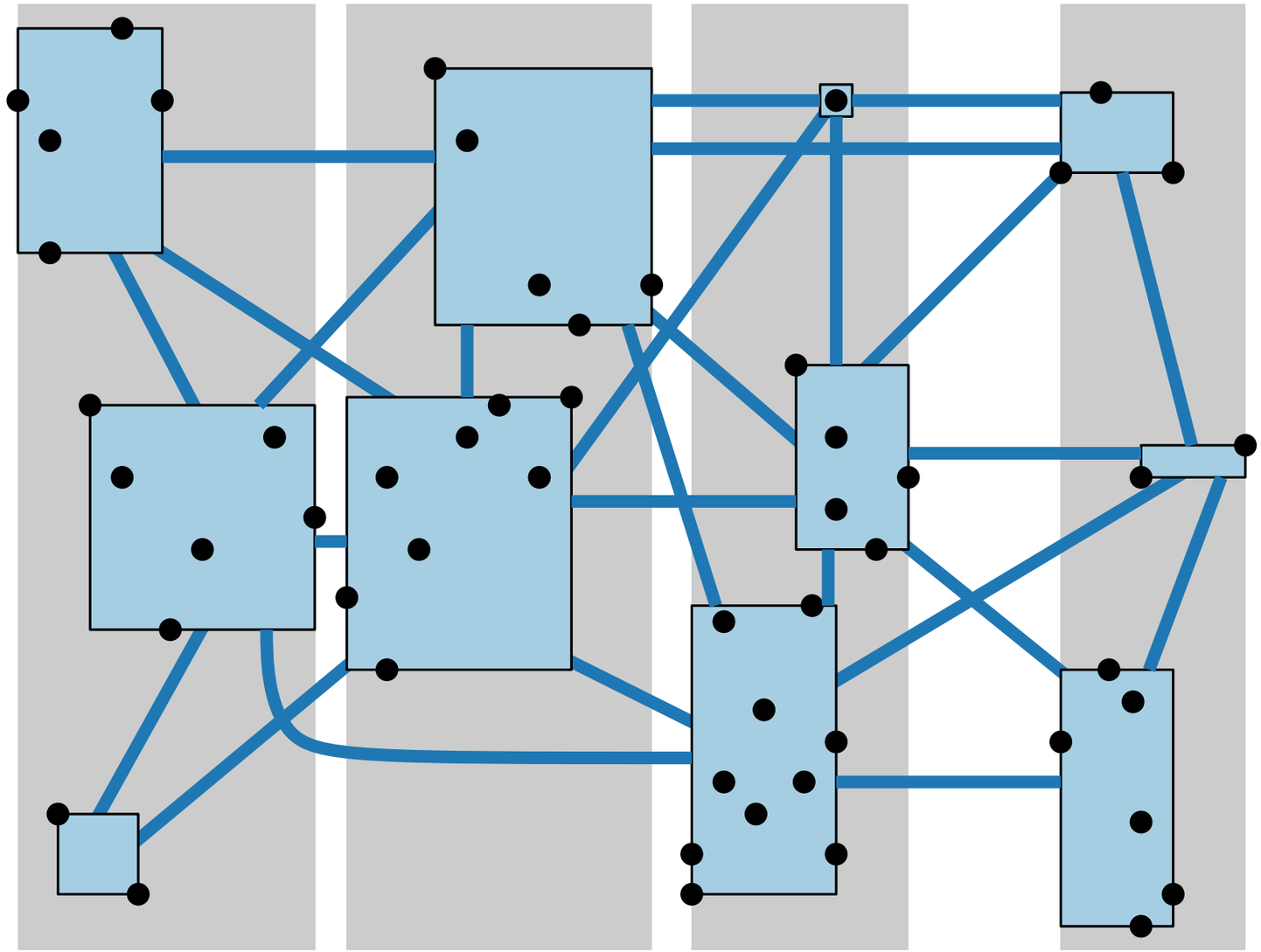
- Nach x sortieren: $\mathcal{O}(n \log n)$
- Pro Streifen nach y sortieren: $\sum_j \mathcal{O}(n_j \log n_j) = \mathcal{O}(n \log n)$
- Insgesamt: $\mathcal{O}(n \log n)$

Box Graph $\mathcal{G}_{\text{box}}$

$k = 4$

ε:

$\varepsilon/\sqrt{2}$:



Eigenschaften der Boxen.

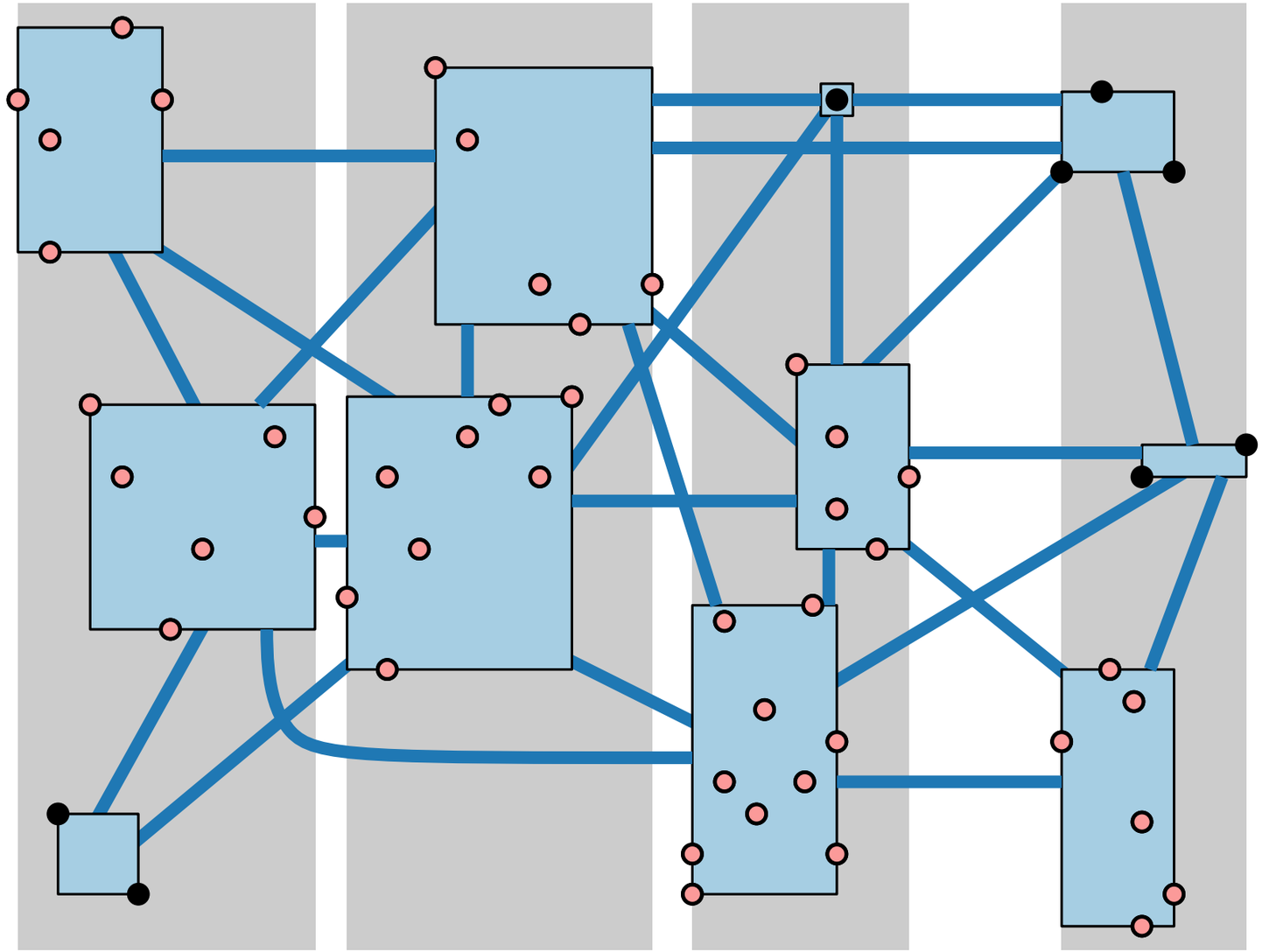
- Alle Punkte in einer Box... sind in ε-Nachbarschaft. (Boxbreite und -höhe $\leq \varepsilon/\sqrt{2}$)
- In Boxen mit min. k Punkten... sind alle Punkte Kerne.
- In Boxen mit $< k$ Punkten... kann es auch Kerne geben!

Eigenschaften von Boxpaaren.

- Verbinde Boxen mit Distanz $\leq \varepsilon$.
- Nicht-Nachbarn in $\mathcal{G}_{\text{box}}$: Keine der Punkte liegen in ε-Nachbarschaft.
- Wie viele Nachbarn kann eine Box haben? **22** $\in \mathcal{O}(1)$

Box Graph $\mathcal{G}_{\text{box}}$

$k = 4$ $\quad \varepsilon$: $\quad \varepsilon/\sqrt{2}$:



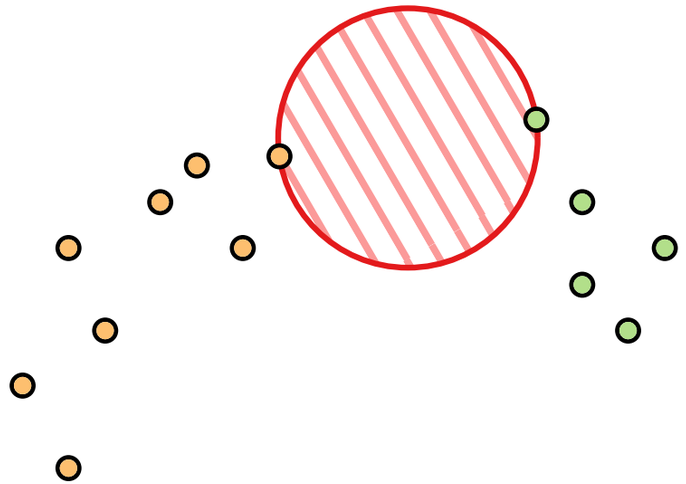
2. Finde alle Kerne

- Haben bereits alle Kerne in „schweren“ Boxen
- Für alle “leichten“ Boxen:
  - für alle Nachbarn:
    ... teste alle Paare.

Laufzeit?

- Andere Box ist auch leicht:
  $\mathcal{O}(k^2) = \mathcal{O}(1)$
- Andere Box ist schwer:
  Lasse schwere Box bezahlen:
  Punkte in schweren Boxen werden $\leq 22k$ mal überprüft
  (!!) ➡ $\mathcal{O}(kn) = \mathcal{O}(n)$

Box Graph $\mathcal{G}_{\text{box}}$

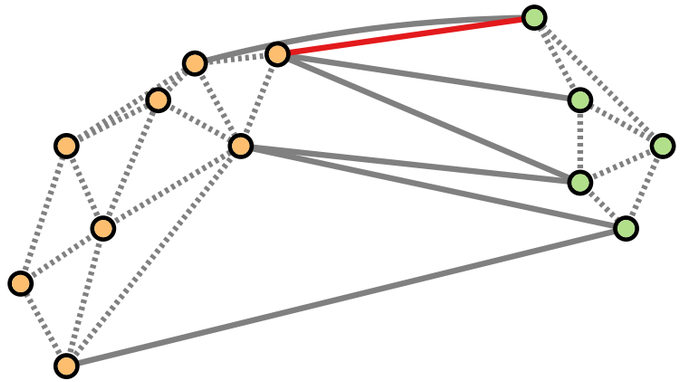


Paare von schweren Boxen.

- Das sind alles Kerne.
- Sind sie im **gleichen** Cluster?

BichromatischesNähestesPaar

Box Graph $\mathcal{G}_{\text{box}}$



Paare von schweren Boxen.

- Das sind alles Kerne.
- Sind sie im **gleichen** Cluster?

BichromatischesNächstesPaar

- Delaunay Triangulierung (DT) enthält diese Kante
- DT hat n Kanten, berechenbar in $\mathcal{O}(n \log n)$ Zeit
- Kanten in $\mathcal{G}_{\text{box}}$ bezahlen: Kante ij bezahlt $c_{ij}(n_i + n_j) \log(n_i + n_j)$ mal.

Insgesamt $\mathcal{O}(n \log n)$

weil $\sum_{ij \text{ ist Kante}} n_i \leq 22 n_i$.

Ergebnis

Annahme: ε frei, k feste Konstante, Euklidische Abstände

	2D	dD	
DBSCAN	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{2-\frac{2}{\lceil d/2 \rceil + 1}+\gamma})$	$\gamma > 0$
HDBSCAN	$\mathcal{O}(n \log n)$ expected	$\times$	

1. Erstelle $\mathcal{G}_{\text{box}}$
2. Finde Kerne
3. Verbinde Cluster

(4. Weise Randpunkte zu.)

BichromatischesNächstesPaar statt Delaunay Triangulierung.

(Agarwal, Edelsbrunner, Schwarzkopf, 1991)