

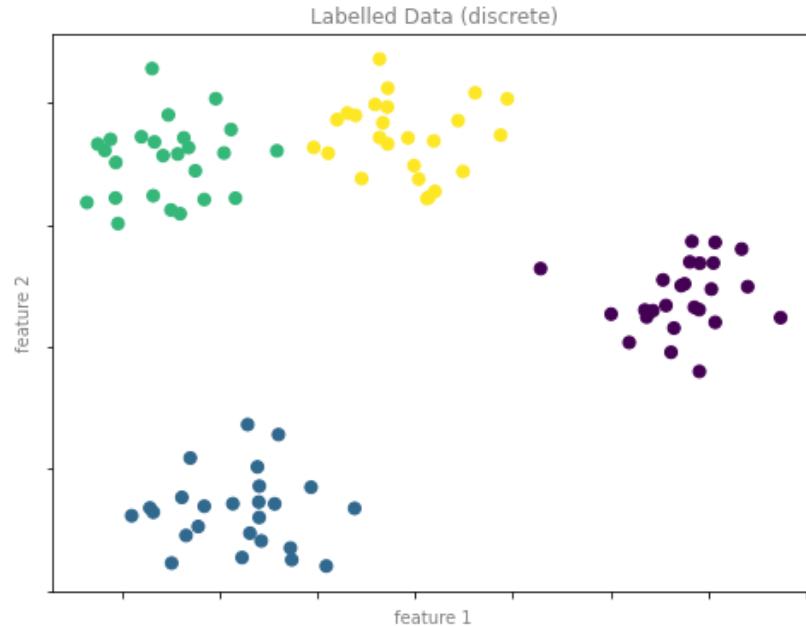
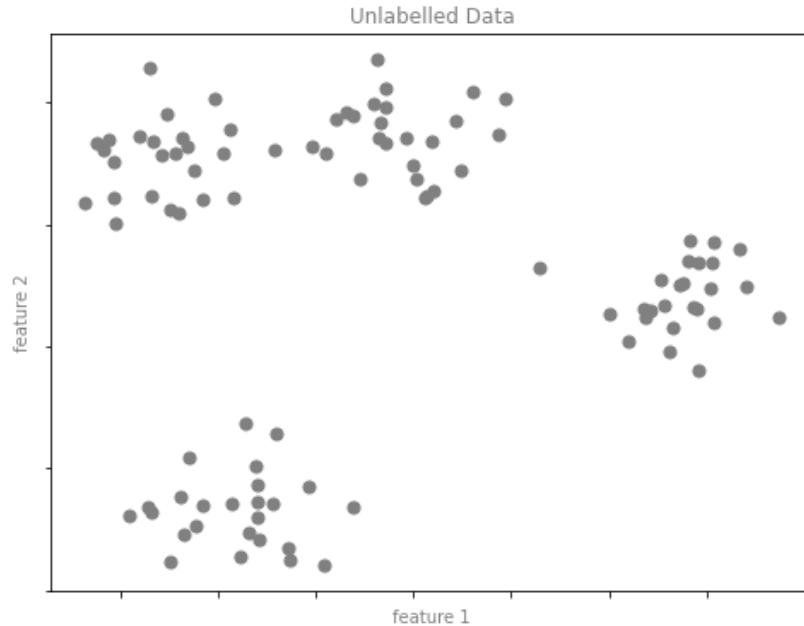
Modeling

Unsupervised Learning: Clustering

Agenda

- Clustering im Allgemeinen
- k-means Clustering: generell und in `sklearn`
- Expectation-Maximization Algorithmus
- k-means im Detail
- Nachteile des EM-Algorithmus
- Validierung von Clustering
- Variationen k-means
- Dichtebasiertes Clustering: DBSCAN

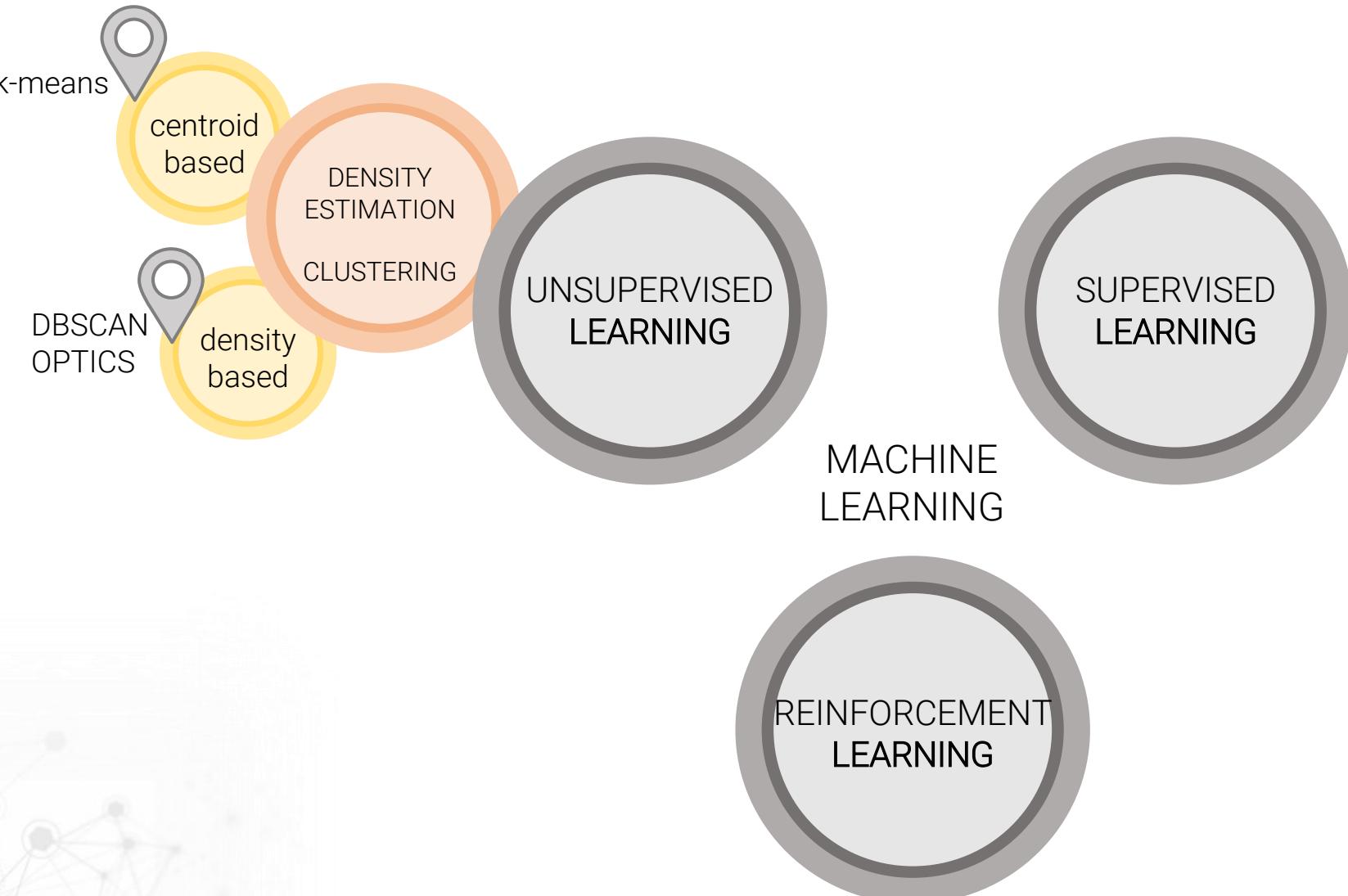
Was sehen Sie?



So what?

- Ihr visueller Apparat entdeckt automatisch und ohne Mühe Gruppierungen bzw. Struktur in Daten
- Wie bringen wir das Algorithmen bei?¹

SVM: wo befinden wir uns?

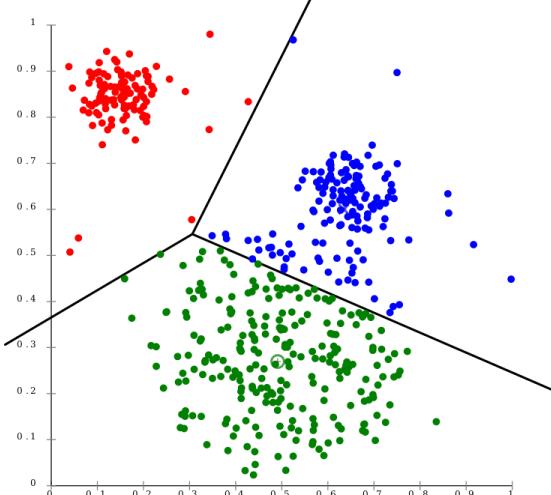


Clustering im Allgemeinen

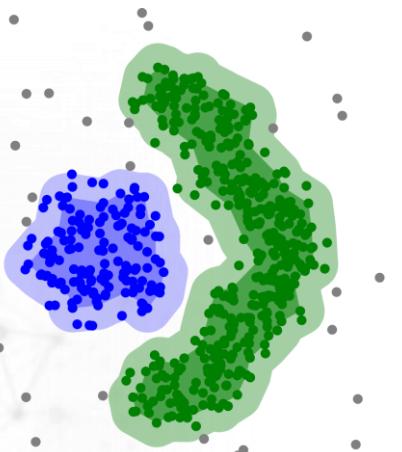


Was denken Sie?

Partitionierend



Dichtebasiert



Wie würden Sie mit eigenen Worten beschreiben, was ein Clustering bewirkt bzw. durchführt?



Was denken Sie?

Was heißt hier optimal?

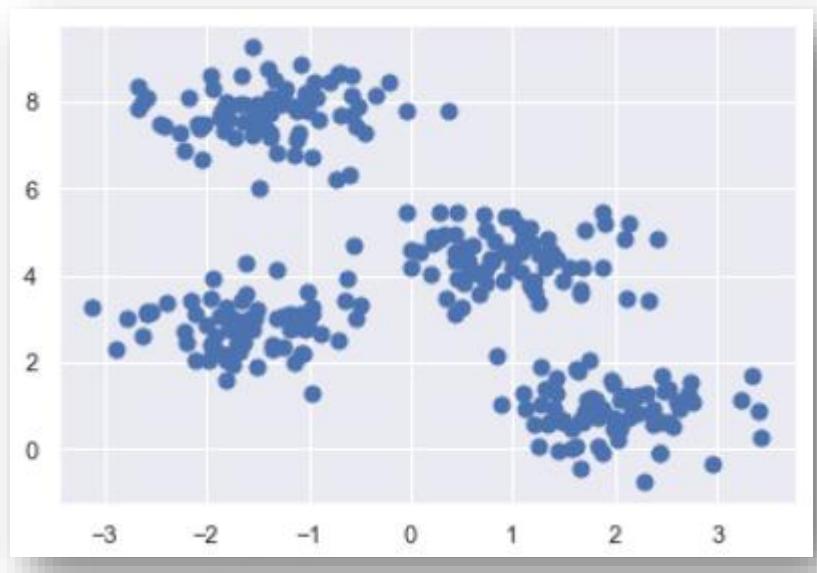
- Clustering Algorithmen versuchen aus den Eigenschaften der Daten eine **optimale Unterteilung in Gruppen** zu erreichen → diskrete Labels
- Es gibt **verschiedene Arten des Clusterings**:
 - Partitionierendes Clustering (z.B. k-means und Varianten)
 - Dichtebasierter Clustering (z.B. DBSCAN, OPTICS)
 - Hierarchisches Clustering
 - Gitterbasierte Verfahren

k-means: im Allgemeinen



Was denken Sie?

Wie wird ein Cluster definiert?



- Wir beginnen mit dem *k-means* Algorithmus
- Der *k-means* Algorithmus sucht nach einer **vorabbestimmten Anzahl** an Clustern in einem **nicht gelabelten** Datensatz
- **Heuristik** des *k-means* Algorithmus:
 - Der Cluster-Mittelpunkt (== **Centroid**) ist das arithmetische Mittel aller Punkte des jeweiligen Clusters
 - Jeder Datenpunkt ist **näher** an seinem eigenen Centroid als an anderen



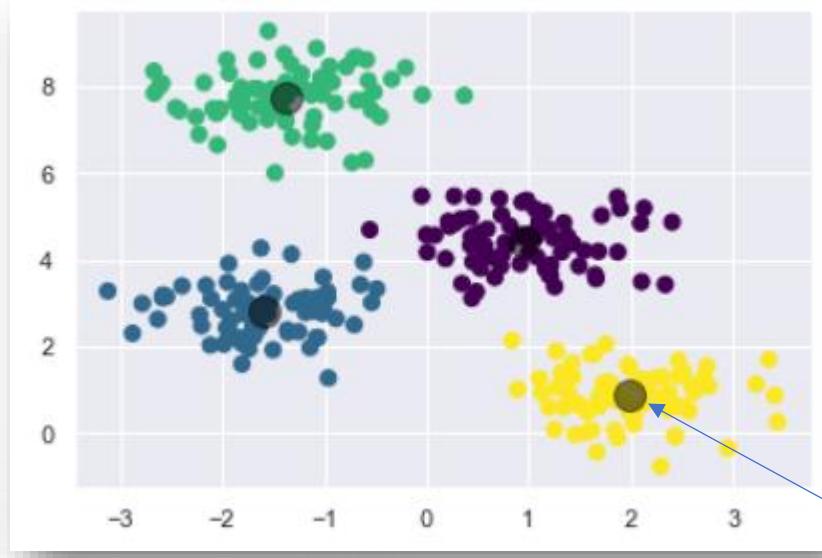
So what?

Unsere Leitfrage: Wie macht der *k-means* Algorithmus das, was wir „mit dem Auge“ mühelos können?

```
1 X, y_true = make_blobs(n_samples=300, centers=4,  
2                         cluster_std=0.60, random_state=0)  
3 plt.scatter(X[:, 0], X[:, 1], s=50);
```

k-means: sklearn

```
1 from sklearn.cluster import KMeans  
2 kmeans = KMeans(n_clusters=4)  
3 kmeans.fit(X)  
4 y_kmeans = kmeans.predict(X)
```



Centroid

```
1 plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')  
2 centers = kmeans.cluster_centers_  
3 plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```

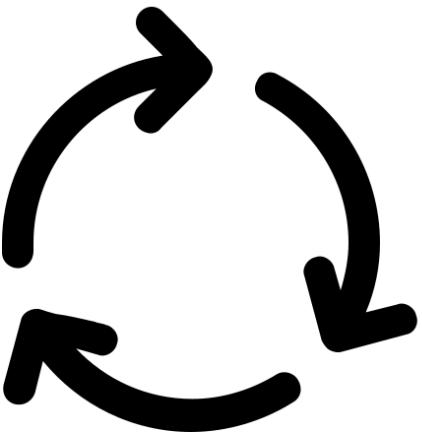
- In sklearn weist der k-means Algorithmus die gleiche Struktur wie andere Modelle auf
- Als wichtiger Zusatz bei der Instanziierung des Modells müssen wir die **Anzahl an Clustern** angeben
- Die **Zugehörigkeit** bzw. **Labels** der Datenpunkte (auch neuer bzw. ungesehener Datenpunkte) zu den jeweiligen Centroids erhalten wir über die `.predict()`-Methode
- Die erlernten Centroids stecken im `.cluster_centers_-Attribut` des Modells



Was denken Sie?

Wie findet der Algorithmus die Centroids ohne alle möglichen Kombinationen explizit zu evaluieren?

Einschub: Expectation-Maximization



- Die Heuristik der vorletzten Folie wird umgesetzt – ein „Rezept“ entsteht → die allgemeine Form dieses „Rezeptes“ ist der sog. **Expectation-Maximization-Algorithmus**
- Der EM-Algorithmus findet in vielen Machine Learning Modellen Anwendung
- Es handelt sich um einen **iterativen** Algorithmus
- **Generelle Idee:**
 1. Initialisiere eine **zufällige** Konfiguration des Modells
 2. Führe folgende zwei Schritte abwechselnd bis **Konvergenz** aus:
 - i. **E(xpectation)-Schritt:** Zuordnung der Daten zu den einzelnen Teilen des Modells
 - ii. **M(aximization)-Schritt:** Parameter an die neueste Zuordnung anpassen
- Konvergenz: findet keine **wesentliche** Verbesserung mehr statt → Abbruch

Einschub: Expectation-Maximization



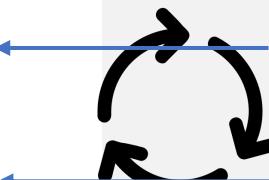
Was denken Sie?

Was sind die analogen Schritte beim k-means?

Initialisiere Centroids zufällig

Weise Datenpunkte zum nächsten Centroid zu

Berechne die neuen Centroids als arithmetische Mittel der neuen Datenpunkte



So what?

- *Expectation*: bedeutet beim k-means also „unsere Erwartung“ updaten, zu welchem Cluster die Datenpunkte gehören
- *Maximization*: ein Gütemaß wird maximiert (in unserem Fall durch die Berechnung des Mittelwertes)

- Die Heuristik der vorletzten Folie wird umgesetzt – ein „Rezept“ entsteht → die allgemeine Form dieses „Rezeptes“ ist der sog. *Expectation-Maximization-Algorithmus*

- Der EM-Algorithmus findet in vielen Machine Learning Modellen Anwendung

- Es handelt sich um einen **iterativen** Algorithmus

- **Generelle Idee:**

1. Initialisiere eine **zufällige** Konfiguration des Modells
2. Führe folgende zwei Schritte abwechselnd *bis* Konvergenz aus:

- i. **E(xpectation)-Schritt**: Zuordnung der Daten zu den einzelnen Teilen des Modells
- ii. **M(aximization)-Schritt**: Parameter an die neueste Zuordnung anpassen

- Konvergenz: findet keine **wesentliche** Verbesserung mehr statt → Abbruch

k-means: Input und Output



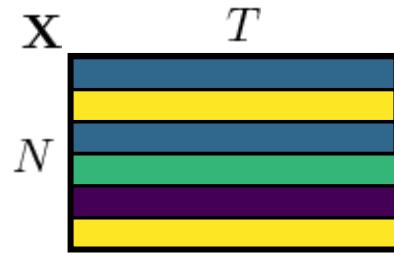
\mathbf{X} : Data matrix with dimensions $N \times T$

N : number of observations

T : number of features (e.g. \bar{x}, σ^2, \dots)

k : number of clusters to be extracted

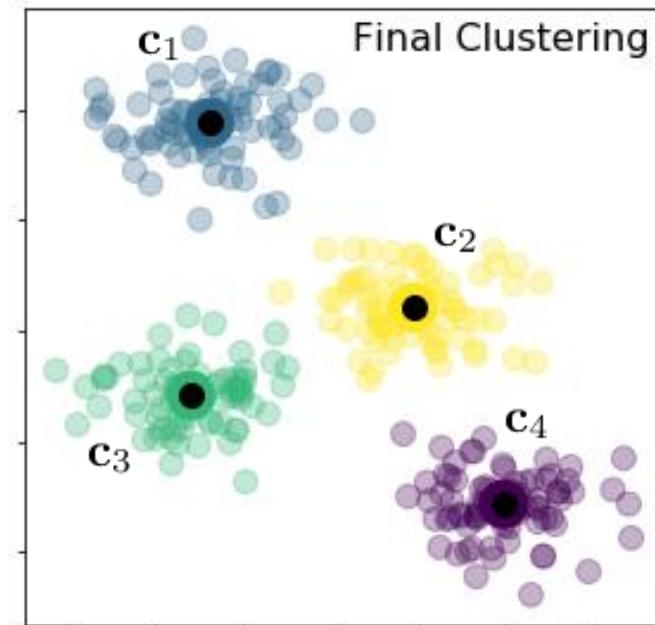
Input



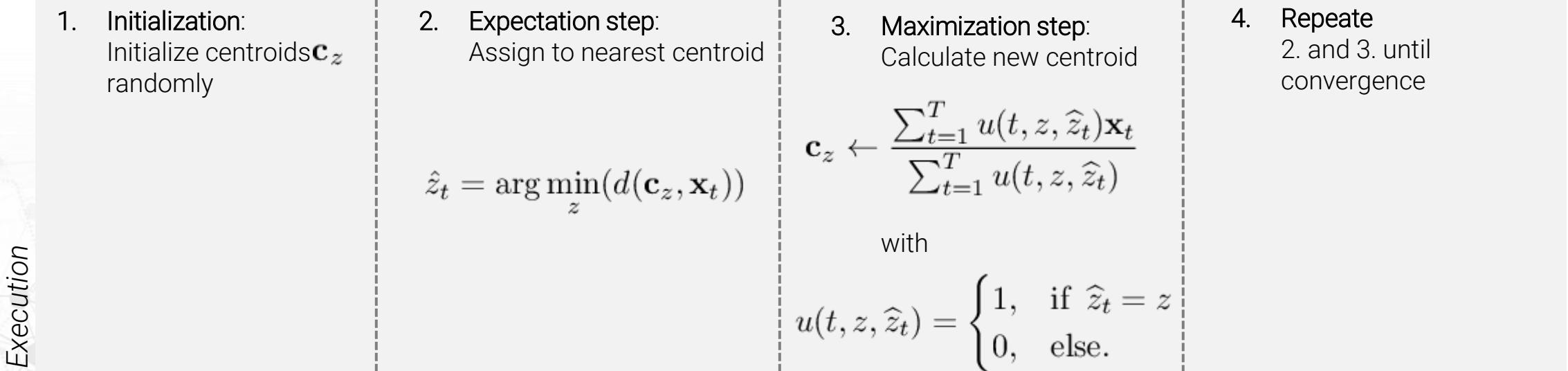
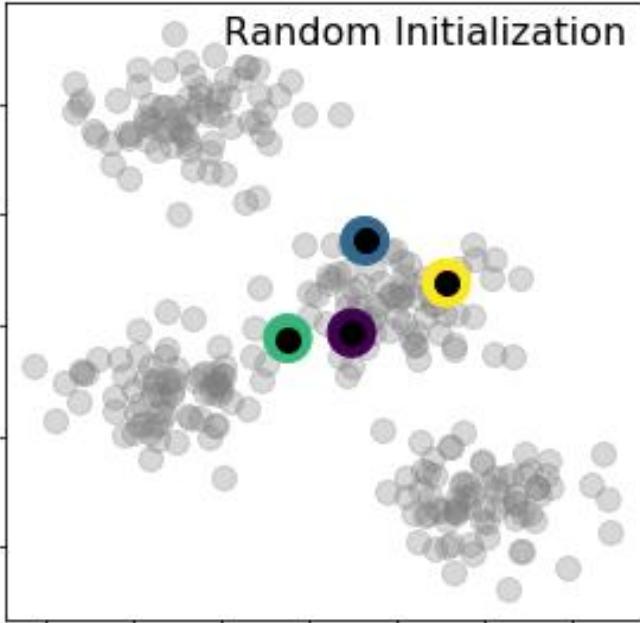
\mathbf{c}_z : cluster centroids with $z = 1, \dots, k$

\hat{z}_t : assigned cluster index of data point \mathbf{x}_t

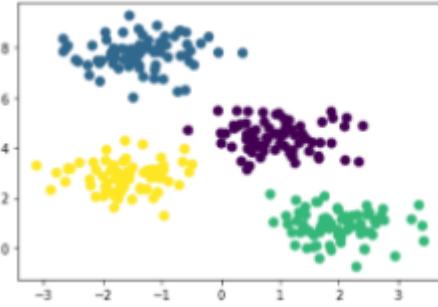
Output



k-means: Ausführung



Beispiel: k-means from Scratch



Der k-means Algorithmus ist so anschaulich, dass wir diesen Algorithmus "from Scratch" - also im Detail - nachprogrammieren können. Unsere Aufgabe in diesem Beispiel ist es also, den k-means Algorithmus Schritt für Schritt zu implementieren. Schreiben Sie also eine Funktion, die als Eingabeargumente die Daten und die Anzahl zu extrahierender Cluster aufnimmt - und uns die Centroids und Labels der Daten zurückgibt.