clustering

1 クラスタリング

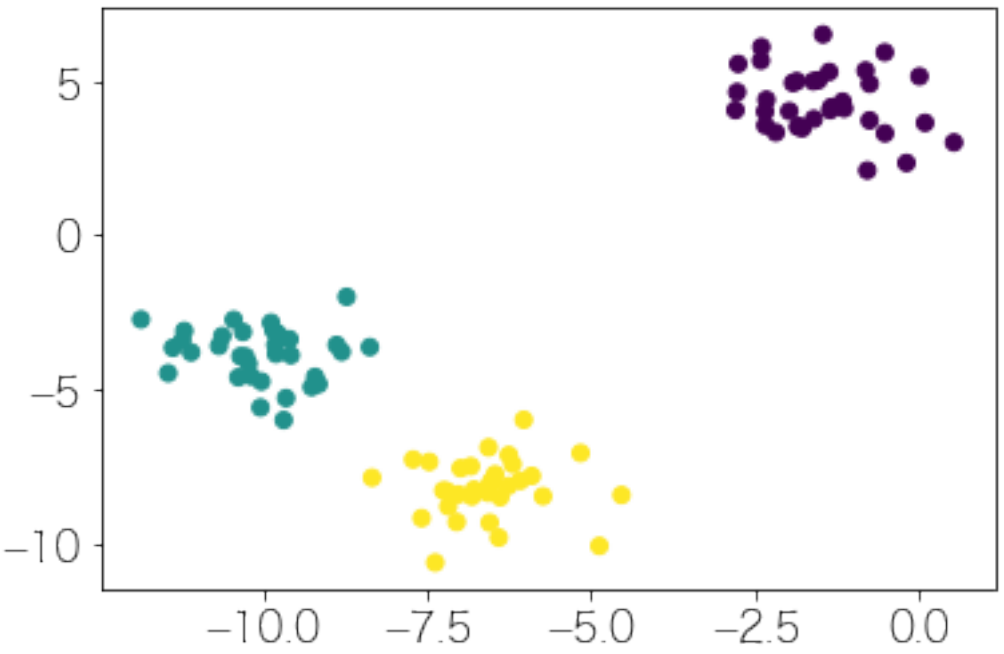
- 教師なし学習の一つ
  - 多数のアイテムが与えられた際に、それをいくつかのクラスタに分割する
  - 正解はない

1.1 K-means

- 代表的なクラスタリングアルゴリズム
- 2 つの動作を収束するまで繰り返す
  - 仮のクラスタの重心 (セントロイド) を求める
  - 最も近い重心に属するように仮のクラスタを再構成する
- 事前にクラスタ数を指定する必要がある

[47]:
```
# 3つのクラスタからなるデータセットを作る
X, y = make_blobs(random_state=1)
plt.scatter(X[:, 0], X[:, 1], c=y)
```

[47]:
```
<matplotlib.collections.PathCollection at 0x7f9358c72490>
```



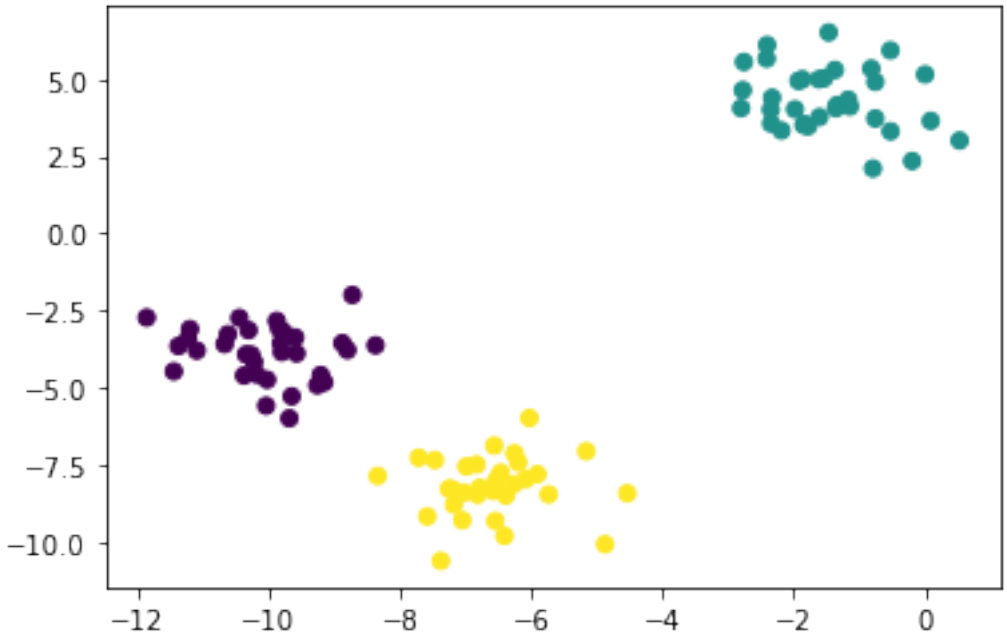
[ ]:

[48]:
```
# K-means による クラスタリング
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
# 結果の確認
kmeans.labels_
```

[48]:
```
array([0, 1, 1, 1, 2, 2, 2, 1, 0, 0, 1, 1, 2, 0, 2, 2, 2, 0, 1, 1, 2, 1,
       2, 0, 1, 2, 2, 0, 0, 2, 0, 0, 2, 0, 1, 2, 1, 1, 1, 2, 2, 1, 0, 1,
       1, 2, 0, 0, 0, 0, 1, 2, 2, 2, 0, 2, 1, 1, 0, 0, 1, 2, 2, 1, 1, 2,
       0, 2, 0, 1, 1, 1, 2, 0, 0, 1, 2, 2, 0, 1, 0, 1, 1, 2, 0, 0, 0, 0,
       1, 0, 2, 0, 0, 1, 1, 2, 2, 0, 2, 0], dtype=int32)
```

[5]:
```
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_)
```

[5]:
```
<matplotlib.collections.PathCollection at 0x7f9350b854f0>
```

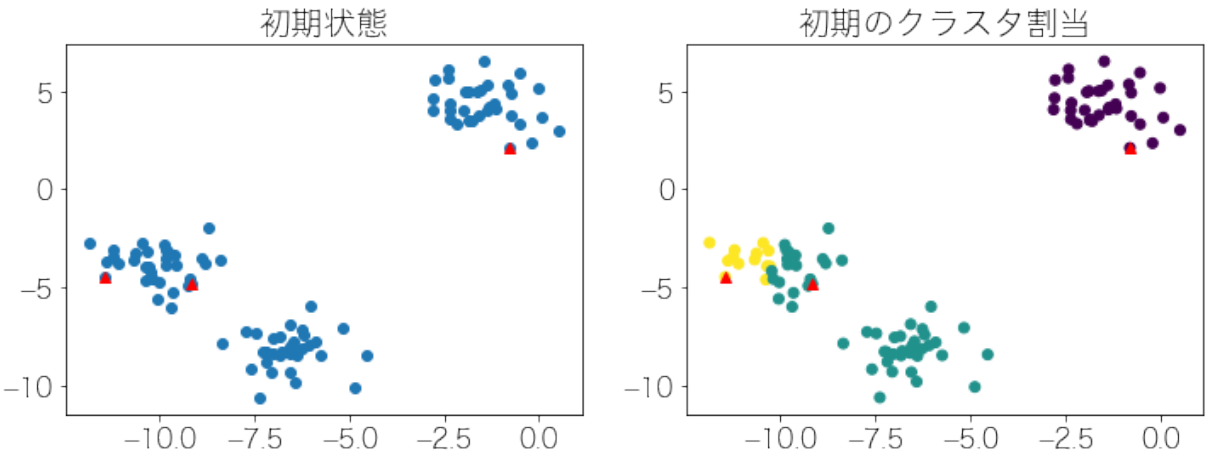


1.1.1 K-means アルゴリズムの説明

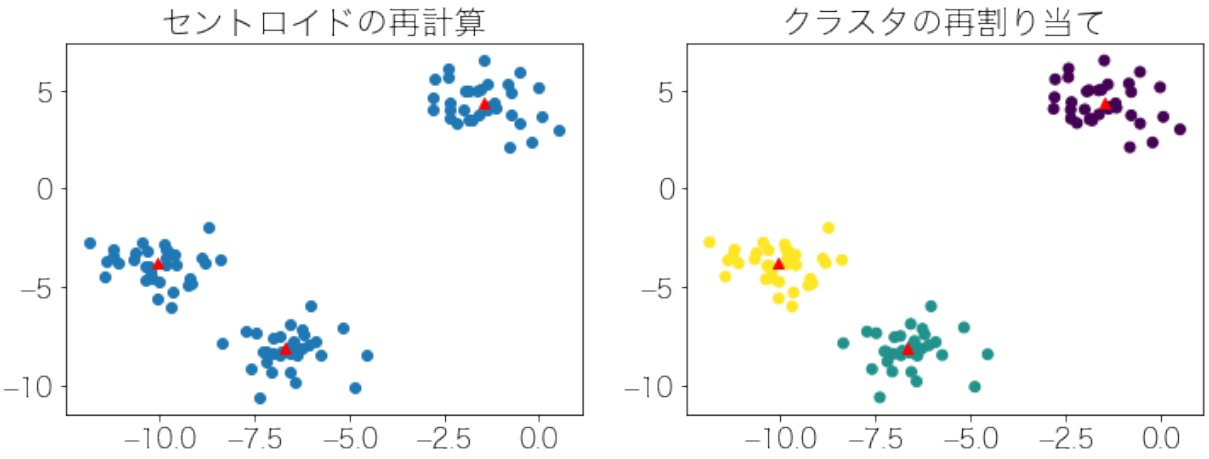
1. 初期のセントロイドを適当に決める
   - ここではデータの最初の 3 要素を抜き出しているが、ランダムにしたほうがいい
2. 個々のデータについて、最寄りのセントロイドを調べ、そのセントロイドが代表するクラスタに属することにする。
3. 個々のクラスタに対して、重心点を計算しその点を新たなセントロイドとする。
4. 新しいセントロイドと、一つ前のセントロイドが変化しているか

- 変化してなければ収束したと判断して終了
- 変化していれば、2 に戻る

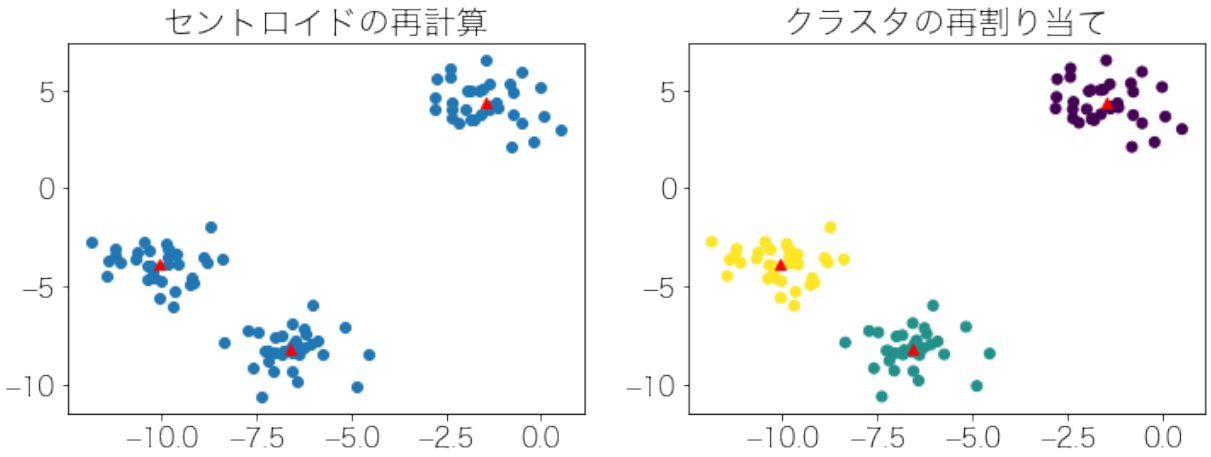
[36]: '初期状態。赤三角が適当に決めた初期セントロイド。これを用いてクラスタ割当を行う。'



[38]: '1 ステップ目'



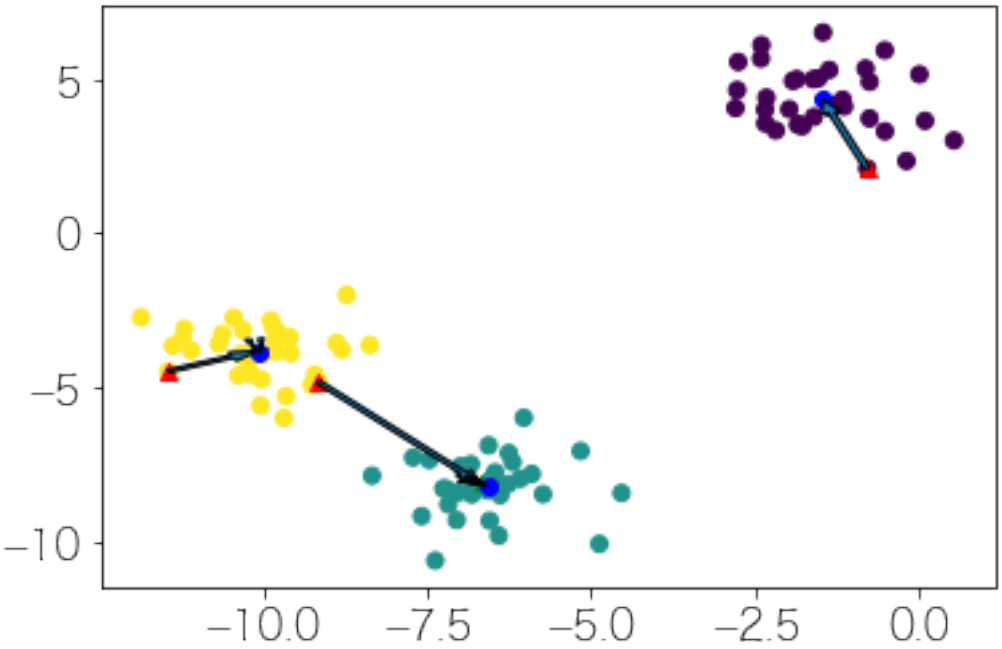
[39]: '2 ステップ目'



1.1.2 セントロイドの動き

- 矢印のように動く
- セントロイドの初期値が近くても、自然に分離していく

[41]: 'セントロイドの動き'



1.2 クラスタリングの評価指標

- 一般論としてクラスタリングの評価は非常に困難
  - 「妥当な」クラスタはいくらでもあり得る
  - 例: 顔写真をクラスタリングする。男性と女性でクラスタ分類されるのと、メガネのあるなしでクラスタ分類す

るのとはどちらが良いクラスタリングか?

- 正解データがあるとき
  - ARI: 調整ランド指数
  - NMI: 正規化相互情報量
- シルエットスコア
  - クラスタの凝集度を評価
  - 0-1 の値を取る

1.3 K-means の問題点

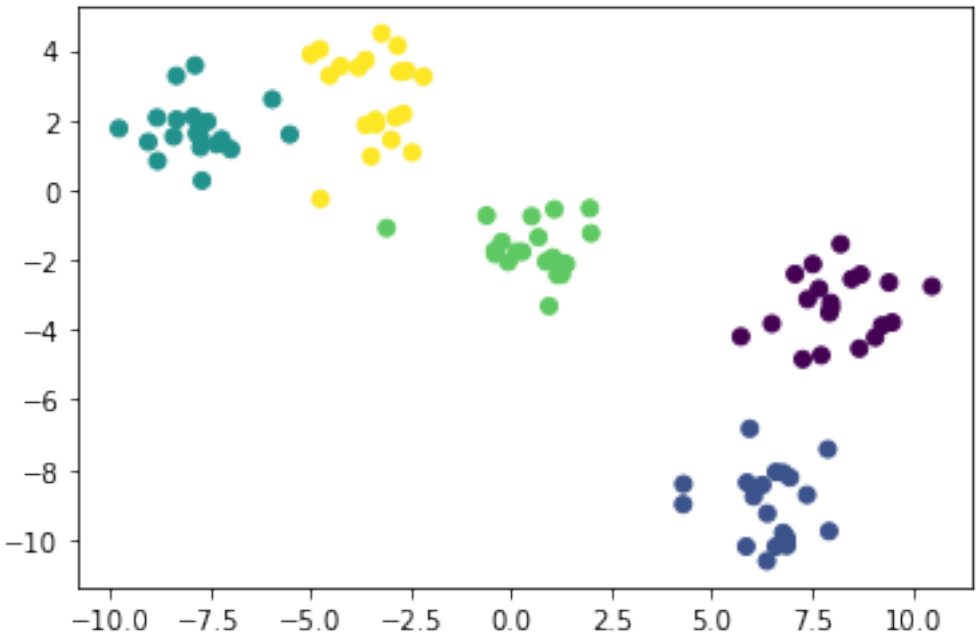
- 事前にクラスタの数を指定しなければならない
  - 一般にはクラスタ数は事前にはわからない
- 凸形状の分布でないとうまく分割できない

1.3.1 クラスタ数の決定

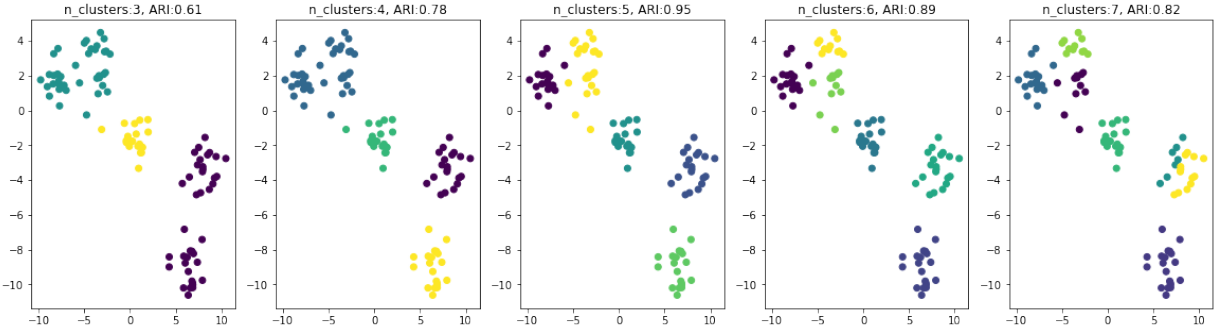
- シルエットスコアを用いる
  - 個々の点について、以下のシルエット係数を計算し、すべての点の平均をとったもの
  - シルエット係数: クラスタがまとまっていて、他のクラスタから離れていると高くなる。$\frac{b-a}{max(a,b)}$: その点が含まれているクラスタへの距離を a, それ以外で最も近いクラスタへの距離を b
- エルボー法
  - クラスタ数を変化させてクラスタリングを複数回行う
  - 出来上がったクラスタの凝集度を、クラスタセンターとの平均二乗誤差で評価
  - 評価値が急に変動した、肘 (エルボー) のような形状になっている点のクラスタ数を採用する

[15]:
```
# 5クラスタのデータセットを作る
X, y = make_blobs(random_state=6, centers=5)
plt.scatter(X[:,0], X[:,1], c=y)
```

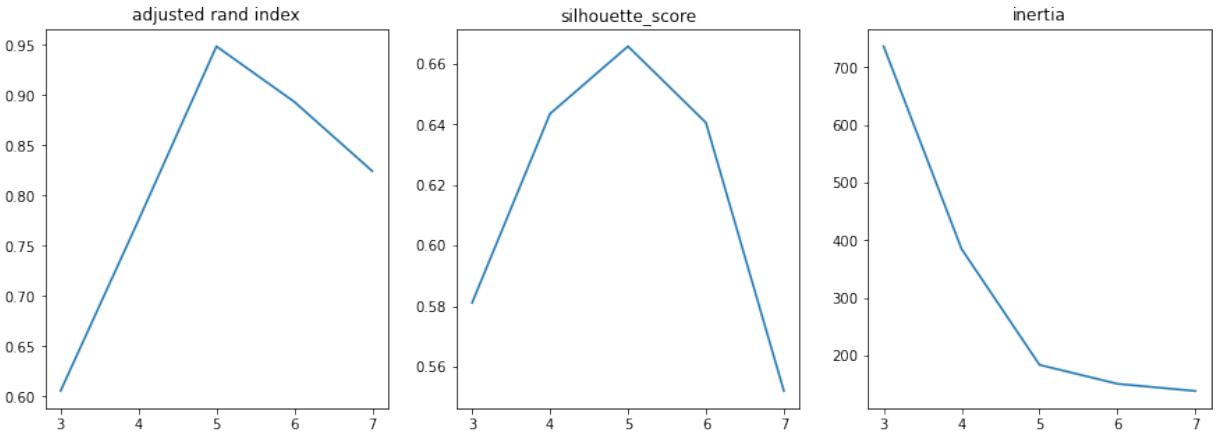
[15]:
```
<matplotlib.collections.PathCollection at 0x7f933821f130>
```



- クラスタ数を 3-7 に変えた場合の結果と ARI 値
  - 注意: ARI は正解がないと算出できないのでクラスタ数の決定には使えない



[24]:
```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].plot(range(3, 8), aris)
axes[0].set_title("adjusted rand index")
axes[1].plot(range(3, 8), sils)
axes[1].set_title("silhouette_score")
axes[2].plot(range(3, 8), inrs)
axes[2].set_title("inertia")
None
```

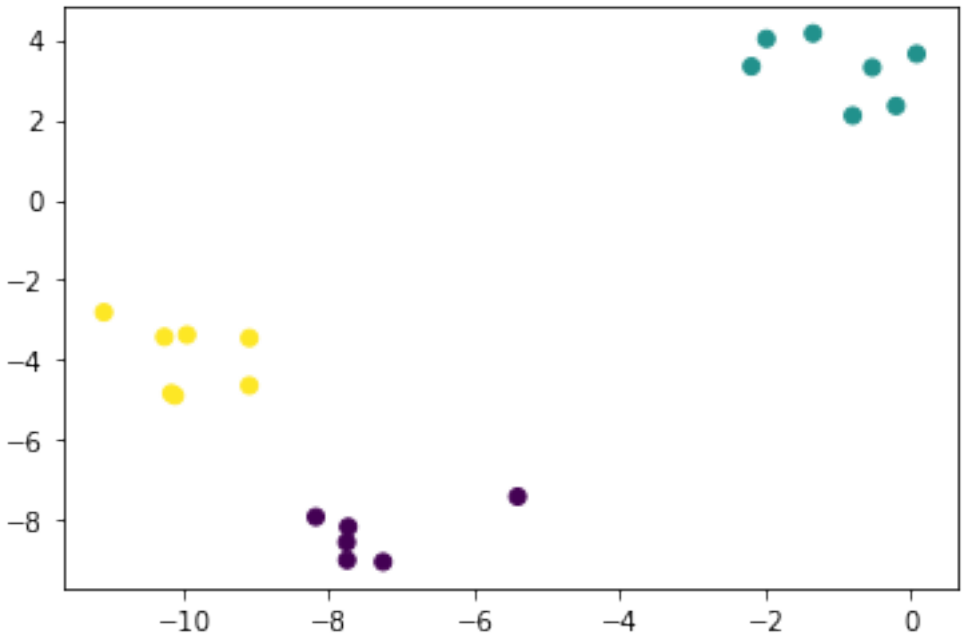


1.4 凝集型クラスタリング

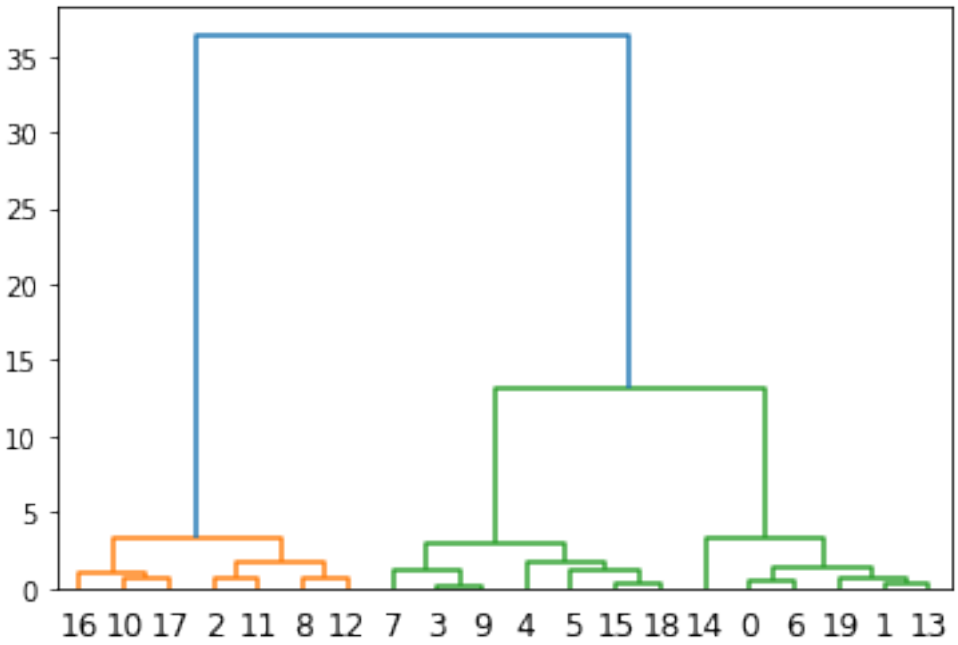
- 1 点 1 クラスタという状況から初めて徐々にクラスタを大きくしていく
- 最も近接している 2 つのクラスタを融合して、クラスタの数を減らしていく
- デンドログラムを見ると適切なクラスタ数がわかる

[26]:
```
X, y = make_blobs(random_state=1, n_samples=20)
agg = AgglomerativeClustering(n_clusters=3)
agg.fit_predict(X)
plt.scatter(X[:,0], X[:,1], c=agg.labels_)
```

[26]:
```
<matplotlib.collections.PathCollection at 0x7f93410de550>
```



[19]:
```
# デンドログラムの描画
from scipy.cluster.hierarchy import dendrogram, ward
linkage_array = ward(X)
dendrogram(linkage_array)
None
```

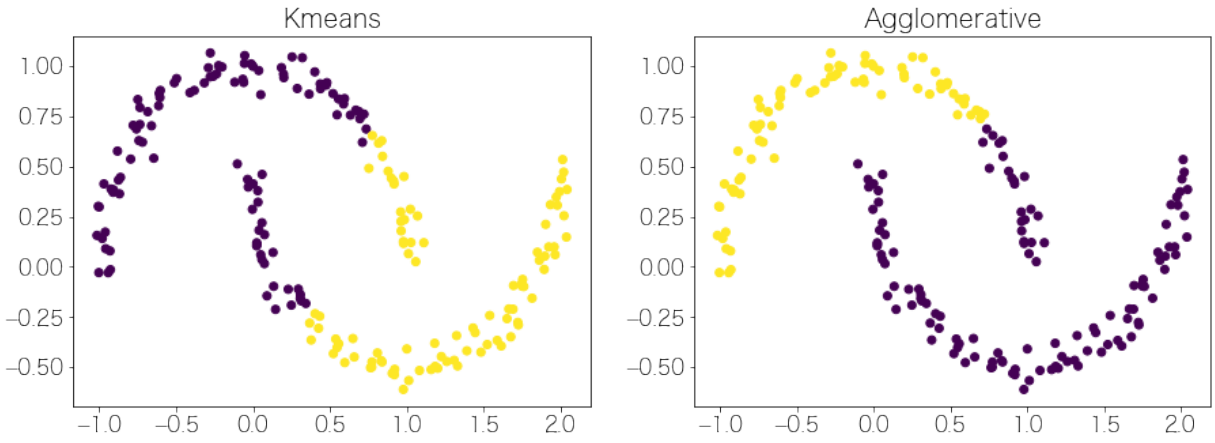


- 高さは統合するクラスタ間の距離を表す
- 距離が大きく離れたものを統合する部分で統合を停止すれば、適切なクラスタが得られる。
- この場合、クラスタ数は 3, もしくは 2 が適当

1.5 K-means でも凝集法でも分類できない例

- 凸でない形状のクラスタはうまくクラスタリングできない
- 本質的に凸の形状に空間を分割しようとするため

[46]: '左は Kmeans, 右は凝集型クラスタリング: いずれもうまく分類できていない '



1.5.1 DBSCAN

- 密度に基づくノイズあり空間クラスタリング
- 密度の高い領域がクラスタを構成していて、クラスタ間には密度が低い領域が存在するという過程
- 密度の高いコアサンプルを探し、近傍点を加える形でクラスタを作っていく
- どのクラスタにも属さないノイズ点が存在しうる
- パラメータの調整が必要
  - min_samples コアサンプルとなるために必要な近傍点数
  - eps 近傍を決める距離

[22]:
```
dbscan = DBSCAN(min_samples=5, eps=0.2)
dbscan.fit_predict(X)
plt.scatter(X[:,0], X[:,1], c=dbscan.labels_)
"DBSCAN による分類"
```

[22]:
```
<matplotlib.collections.PathCollection at 0x7f933011bf70>
```